Population Iterative Annealing Algorithm with Improved Inver-over Operator for TSP

Zhi Yuan[1] and Yuanping Zhang[2, *]

[1]South China Institute of Software Engineering, Guangzhou University, Guangzhou, 510990, P. R. China
[2]School of Computer and Educational Software, Guangzhou University, 510006, P. R. China
[*]Corresponding author

Abstract—For improving optimizing ability of Inver-over operator in solving TSP, we improved the Inver-over operator and employing annealing skill. First, adopted two strategies to improve Inver-Over operator, one is drawing more optimal edges into the population as earlier mainly through the "2e-switch-1p-shift-opt", and another is avoiding nonfunctional iterations considering the similarity of the population, these improvements increased the converging speed evidently. Second, employed annealing skill in the improved population algorithm, by designing the temperature function with the characteristics of the population, the temperature of the annealing process presented self-adaptive and two-stage decreasing characteristic, the temperature can achieve peak and then decrease periodically, and the peak was decreased gradually. The individuals in the population except the best one can jump present local area with large probability and search in new area. The experiment results showed that the proposed algorithm overcomes GT algorithm in the con-verging speed and final tour obviously.

Keywords-TSP; population intelligent algorithm; inver-over operator; simulated annealing algorithm; iterative annealing

I. INTRODUCTION

An instance of the traveling salesman problem (TSP) consists of a set of cities and the pairwise distances between these cities. The goal is to find the shortest tour that visits every city exactly once and returns to the starting city in the end. TSP is one of the most studied optimization problems, the research on TSP can be used in communication control, network routing and designing of LSIC, etc..

TSP belongs to the class of NP-hard problems that are computationally intensive to solve [1]. Many researchers have tried to solve it with population intelligent algorithm. The GT algorithm, proposed by Guo et al[2], is a population intelligent algorithm for solving TSP specially. The kernel of GT is Inver-over operator and it is easy to program and efficient for small-scale instances. However, for large-scale instances, the operator converges with low speed and it is apt to be trapped in a local optimum.

We proposed an algorithm, in which Inver-Over operator and simulated annealing algorithm are combined. In the proposed algorithm, we first modify the Inver-over operator with two strategies: one is drawing more optimal edges into the population as earlier, and another is avoiding nonfunctional iterations. For implementing the strategy one, the better tour produced by the inverse operations is kept promptly and "2e-switch-1p-shift-opt" is executed on a tour chosen randomly from the population after one population iteration. For implementing the second strategy, at least two inverse operations are required in one tour iteration and a tour is mutated after one population iteration. The improved GT algorithm (IGT) improves converging speed greatly at the early period.

Then, we employed the annealing skill in IGT to design population iterative annealing algorithm (PIA). By designing the temperature function with the characteristics of the population, the temperature of the annealing process presented self-adaptive and two-stage decreasing characteristic, the temperature can achieve peak and then decrease periodically, and the peak was decreased gradually. The individuals in the population except the best one can jump present local area with large probability and search in new area. The PIA algorithm can base on the characteristics of the TSP and population to control the temperature automatically, so to avoid testing temperature parameters repeatedly.

Finally, we test the PIA algorithm on TSPLIB for comparing with GT algorithm. The results showed that PIA possesses better converging speed and global optimizing ability. For the instances with city number fewer than 280, PIA achieves optimal solutions in 30 seconds every time. Under the circumstances of getting the optimal solution, the average time of PIA is less than 20% of GT algorithm. For large scale instances, PIA maintains optimizing ability when the evolution of GT keeps halted.

II. RELATED WORK

A. Inver-over Operator

Many population intelligent algorithms were employed to solve TSP, including the genetic algorithm[3,4], particle swarm optimization[5], ant colony optimization[6,7], etc.. Each kind of population intelligent algorithms possesses coincident arithmetic procedure, the differences of the effects are decided mainly by the operators.

The Inver-over operator includes the crossover operation and mutated operation implicitly. It has been proven that the performance of Inver-over operator overcomes the genetic operators of the order cross (OX), circle cross (CX) and edge recombination crossover (ER), etc.[8].

In the Inver-over operator, the traversal order of the city is used to encode the solution, for example, one tour of 6 cities is encoded as $S=(2,3,4,1,5,6)$. The operator is described as following.

```
Randomly initialize the population P
    While termination-condition is not satisfied
    For each individual Si ∈P
        S' = Si
        Select randomly a city c from S'
        Loop
            If rand(0..1) < pr   //pr=0.02 generally
                Select c' from the remained cities in S'
            Else
                Select randomly an individual from P
                Assign to c' the next city of c in the individual
            If c' is near to c in S'
                Exit loop
            Inverse the section from the next city of c to c' in S'
            c =c'
        End loop
    End for
    If eval(S') <eval(Si)   //eval(*) is the length function
        Si = S'
    End while
```

Inver-over operator executes the population iterations until the termination-condition is satisfied. During every population iteration, each tour in the population is iterated. In each tour iteration, a copy of tour S is created as S', then inverse operation is executed several times on S' by mutating (when $rand<pr$) or crossing over (when $rand \geq pr$). Lastly, S is replaced with S' if S' is better.

Inver-over operator is easy to program. It achieves better effect for the instances with scales fewer than 100 cities. But for large scale instances, it is difficult to jump out the local optimum. Using heuristic methods (e.g., nearest insertion, farthest insertion, neighborhood method) [9] can improve the optimization speed of Inver-over operator at early stage, but there is no practical aid for searching the global optimal solution.

B. Application of Annealing Algorithm in TSP

The simulated annealing algorithm was proposed by Kirkpatrick et al in 1983 [10]. A typical implementation of the simulated annealing algorithm is shown as following, in which, Rs is the solution space and $r(T)$ is a function to reduce the temperature T.

```
Choose an initial solution S
Set T=T0
Repeat until system is frozen
    Do the following cycle l times
        Make a perturbation in S and generate S'∈Rs
        Set ΔC=C(S')-C(S)
        If  ΔC≤0 then S= S'
        If ΔC>0 then set S= S' with probability exp(-ΔC/T)
        Set T=r(T)
Show S
```

The simulated annealing algorithm has been applied to TSP and gained much development. For example, Yang applied deterministic annealing algorithm to the travelling salesman problem [11], Baranwal applied deterministic annealing algorithm for approximating the solutions to the multiple TSP and other variants on the TSP [12].

With the development of the population intelligent algorithm, some algorithms combining with population intelligence and annealing algorithm are produced for solving TSP. Eswarawaka, Lan and Yao proposed their individual algorithm combining with simulated annealing algorithm and genetic algorithm [13-15], Honjo, Cheng proposed algorithms combining with simulated annealing and particle swarm optimization [16,17], Guzman proposed a framework for the parallel solution of combinatorial problems implementing tabu search and simulated annealing algorithms[18], Wang proposed a multi-agent simulated annealing algorithm [19].

The same strategy for these combined algorithms is: using population intelligent for iterative optimizing and using annealing algorithm to jump out the local optimum. A key problem needed to solve in the combined algorithms is how to set and control the temperature. At present, the most frequently used method is repeatedly testing the initial temperature T0 and the arguments in function r(T) for each TSP instance, it produces difficulty for the application.

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

III. POPULATION ITERATIVE ANNEALING ALGORITHM WITH IMPROVED INVER-OVER OPERATOR

A. Improve Inver-over Operator

First modified strategy is drawing more optimal edges (edges in the optimal tour) into the population as earlier. It is based on two intuitions: (1) it should be the high probability event that the shorter tours include more optimal edges than the longer tours; (2) drawing more optimal edges into the population as earlier can improve the converging speed of the algorithm. The reliabilities of these two intuitions are validated by experiments in Section 4.

For drawing more optimal edges into the population as earlier, the Inver-Over operator is modified as: (1) Generate initial solution by the neighborhood method, the size of the neighborhood is set as 6. (2) For each inverse operation to S', compute the incrementing value of the length of tour, if the incrementing value is less than 0, replace S with S' at once. (3) After each population iteration, choose one tour randomly and execute a local optimization called "2e-switch-1p-shift-opt".

The "2e-switch-1p-shift-opt" means, the tour is traversed in order and for present city c_1, searching city c_2 in its

neighborhood (the neighborhood size is set as 6), trying to switch 2 edges as FIGURE 1. or move 1 point as FIGURE 2.
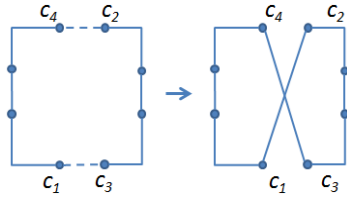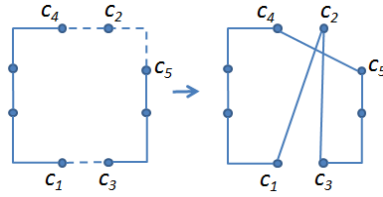

FIGURE I. 2EDGES-SWITCH

FIGURE II. 1-POINT-SHIFT

"2e-switch-1p-shift-opt" on a tour is implemented as following:

```
For i=1 to n
    Assign to c₁ the ith city in the tour
    For each c₂ in the neighborhood of c₁
        Assign c₃ the next city to c₁ in the tour
            IF (c₂== c₃) continue
        Assign to c4 the city next to c₂ in the tour
        Assign to c5 the city previous to c₂ in the tour
        Set Δeval_1= d(c₁, c₂) + d(c₃,c4) - d(c₁, c₃) - d(c₂,c4)
        Set Δeval_2= d(c₁, c₂) + d(c₂, c₃) + d(c4,c₅)
                - d(c₁, c₃) - d(c2,c₄)- d(c₄,c₅)
        If Δeval_1<0 and Δeval_1<Δeval_2
            Inverse the section from c₃ to c₂
        Else if Δeval_2<0 and Δeval_2<Δeval_1
            Move c₂ to the point of c₃
    End for
End for
```

The second strategy is to avoid the void iterations. This strategy is based on the quantity analysis of the population similarity. The population similarity reveals the ratio of same edges included in the tours in the population to the total edges.

The population similarity is computed as formula (1), in which, P is population, S_{best} is the present best individual in the population, n is the number of the cities, m is the number of individuals in the population, $same(S_1,S_2)$ is the number of same edges in the tours S_1 and S_2.

$$sim(P, S_{best}) = \frac{\sum_{i=1}^{m} same(S_i, S_{best})}{m \times n} \times 100\% \qquad (1)$$

The variation of the population similarity can be seen in Figure 2 when the Inver-Over operator is executed.
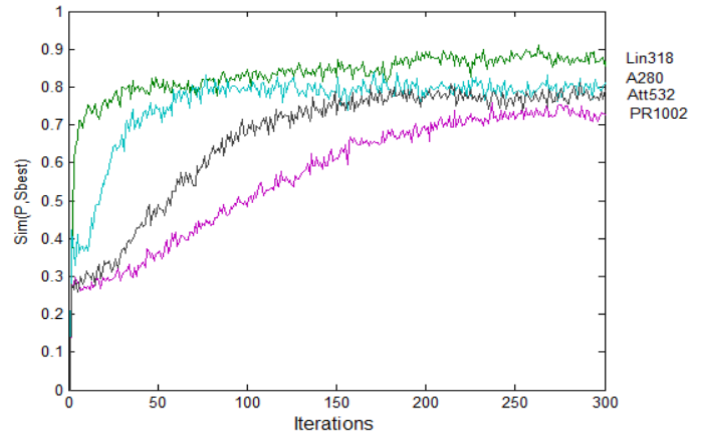

FIGURE III. VARIATION OF POPULATION SIMILARITY

The Figure 2 shows that when the algorithm is executed, the population similarity is becoming large gradually. When the population similarity achieves 0.8, the similarity of two tours is at least 0.6. The probability of r times hybridizing for the tours is less than 0.4r, so the tour iteration produce none practical operation with high probability.

For avoiding the void iterations, the modified methods are: (1) for each tour iteration, at least two inverse operations are executed; (2) after each population iteration, choose one tour randomly from the population except the best tour and execute mutating operation on it as following.

```
Select randomly city c₁ in the tour
Assign c₃ the next city to c₁ in the tour
Select randomly c₂ in the neighborhood of c₁
    IF (c₂≠c₃)
    Assign to c₄ the city next to c₂ in the tour
    Assign to c₅ the city previous to c₂ in the tour
    IF rand(0..1)<0.5
        Inverse the section from c₃ to c₂
    Else
        Move c₂ to the point of c₃
```

All the modified method is described in detail in Section 3, which is called IGT algorithm. The experiments in Section 4 show that IGT overcomes GT in speed greatly.

B. Self-Adaptive Population Iterative Annealing Method

In Inver-Over operator, after population was iterated one time, the individuals in the population will become better or maintain unchanged. The worsen tour will not be accepted. When the scale of the problem is larger, it will be trapped in local optimum easily. Using annealing algorithm, the worsen tour can be accepted for jumping out the local optimal solution, so to improve the ability to search the global optimal solution.

We use the operation contained in the Inver-over operator to produce new tour, none new operations is needed to attach.

For control the temperature. the characteristic of population is used to design the temperature function. The population temperature T is computed as following formula (2), where $eval(S_{best})$ is the length of present best solution, k the time of

iteration of the algorithm, n the number of cities and % the modulo operator.

$$T = \sqrt{eval(S_{best})} \times \frac{(k~\%~n)}{n} \qquad (2)$$

With the formula (2), T is changed periodically as n. During each period, the population experiences temperature rising and then annealing. The length of the present best solution descends continually as the algorithm executing. The peak value of the temperature of each period descends along to reveal bipolar annealing characteristic.

On the base of keeping the elite individual, the worsen tour is accepted according to the following:

$$\text{If}~~ S \neq S_{best} ~\text{and}~ rand(0..1) < \exp\left(-\frac{\Delta eval}{T}\right)$$

then $S = S'$ (3)

C. Proposed Algorithm

Combining the discussion in Sections 3.1 and 3.2, we propose the population iterative annealing algorithm (PIA) as following.

```
Randomly initialize the population P
While the terminal-condition is not satisfied
    Choose one tour and execute "2e-switch-1p-shift"
    Choose one tour and execute one mutated operation
    Compute the temperature T according the formula (2)
    For each individual Sᵢ in P
        S' = Si
        inver_time=0
        Choose one city c from S'
        Loop
          If rand(0..1)<pr  //pr is 0.02 generally
             Choose c' from remained cities in S'
          Else
             Choose another individual from P randomly
             Assign to c' the next city of c in chosen individual
          If c is near to c' in S' and  inver_time>1
             Accept worsen solution according to the formula (3)
             Exit loop
          Else
             inverse the section from the next city of c to c' in S'
             inver_time++
             If incrementing length of tour is less than 0
                  Si= S'
          c=c'
        End loop
    End for
End while
```

Several operations are added in the PIA algorithm on the base of Inver-Over operator. The costs of these operations in each iteration are: $O(n)$ for "2e-switch-1p-shift-opt", $O(1)$ for tour mutation, $O(1)$ for computing T, $O(m)$ for computing probability to accept worse solution. The added cost can be omitted almost comparing with the cost of the Inver-over operator.

IV. EXPERIMENTS

We test the performances of GT and PIA for the instances in TSPLIB. The size of the population is set as 40, the longest time 30 seconds, and each algorithm has been executed 100 times. The average results (avg.), probability converging to the optimal solutions (pro.), average time converging to the optimal solutions in second(time) have been compared. The results have been shown in the table 1.

The Table 1 shows that for the instances with city number fewer than 280, PIA achieves optimal solutions every time in 30 seconds, it is better obviously than GT algorithm. Under the circumstances of getting the optimal solution, the average time PIA needed is less than 20% of GT algorithm.

TABLE I. RESULTS OF GT AND PIA FOR THE TSP INSTANCES

Instance	optimal	GT			PIA		
		Avg.	*Pro.*	*Time*	*Avg.*	*Pro.*	*Time*
ATT48	33522	33534	0.8	2.2	33522	1.0	0.36
EIL51	426	426.36	0.74	9.2	426	1.0	0.55
KROD100	21294	21351	0.045	10.8	21294	1.0	1.3
EIL101	629	636.4	0.02	25.8	629	1.0	3.1
PR144	58537	58639	0.04	15.8	58537	1.0	4.4
A280	2579	2624.2	0.0	----	2579	1.0	4.2

The Table 1 reveals one phenomenon also: the running time of PIA achieving the optimal solution is not positively related to the number of cities (see the cases of EIL101, PR144 and A280). The phenomenon could be resulted for two reasons: (1) The effects of the city neighborhood size setting. With the different characteristic of the city distribution, the size may be different. Through the repeat testing, the size is set as 6. There is not efficient method to decide the size which is adaptive pervasively. (2) The instance of TSP may have many optimal solutions; it reduces the difficulty of searching the optimal solution. For the case A280, we have found many tours with minimized length 2579.

For further comparison of performances of GT, IGT and PIA, they are applied to PR1002 separately, the iteration time is set as 30000, and the converging curves of three algorithms were shown in the Figure 3.

The Figure 3A shows full converging curves of three algorithms, it indicates that the converging speed of the algorithms IGT and PIA overcome GT obviously. It is hard to see the difference of IGT and PIA in the Figure 3A. Enlarging the local area of Figure 3A to 3B, the converging curves from the generations 500 to 30000 show that, on the base of IGT, PIA has strong ability for searching global optimal solution.
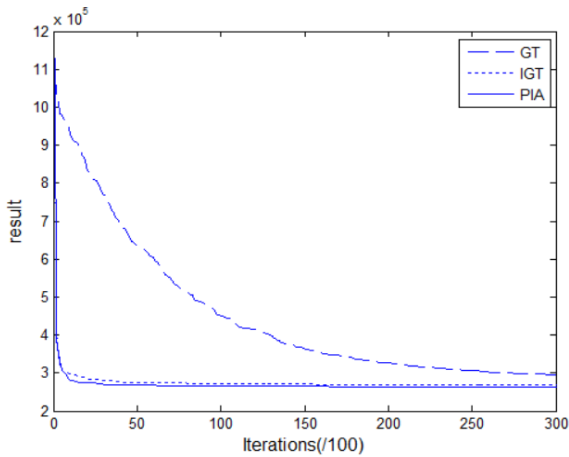
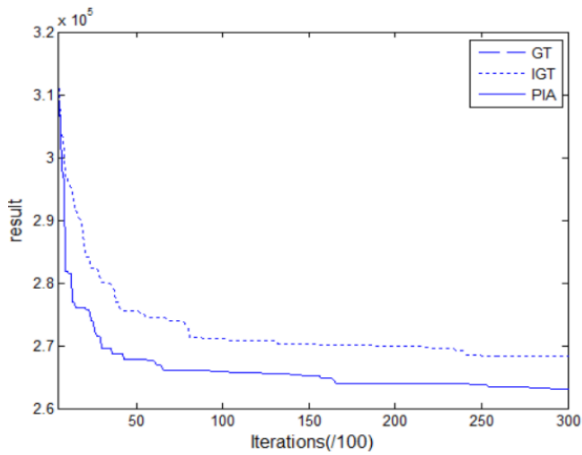FIGURE IV. CONVERGING CURVES OF THREE ALGORITHMS ON PR1002.

FIGURE V. LOCAL CONVERGING CURVES OF IGT AND PIA ON PR1002

V. CONCLUSION

We adopted two strategies to improve Inver-over operator, one is drawing more optimal edges into the population as earlier, and another is avoiding void iterations. The two strategies are realized in improved algorithm, and then, we employed the annealing skill in IGT to design population iterative annealing algorithm (PIA). By designing the temperature function with the characteristics of the population, the temperature of the annealing process presented self-adaptive and two-stage decreasing characteristic. During each period, the individuals in the population except the best one can jump present local area with large probability and search in new area. Comparing the new algorithm to GT, the converging speed and final solution are better obviously.

We have found that, "2e-switch-1p-shift-opt" takes a key role in the algorithm. Keeping the framework of our algorithm, if "2e-switch-1p-shift-opt" can be replaced by other operator with approximate time and better effect, the performance of the algorithm may be improved further.

REFERENCES

[1] T. Bektas, The multiple traveling salesman problem: an overview of formulations and solution procedures, Omega, Vol.34, no.3(2006), 209–219.

[2] T. Guo, Z. Michalewicz, Inver-over operator for the TSP, Lecture Notes in Computer Science, 1498(1998), 803-812.

[3] M. Gorges-Schleuter, Asparagos96 and the traveling salesman problem, Proceedings of the 1997 IEEE Internation Conference on Evolutionary Computation, (1997), 171-174.

[4] Y. Nagata, S. Kobayashi, Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem. In: Proc. 7th International Conference on Genetic Algorithms, (1997), 450-457.

[5] A. W. Mohemmed, N. C. Sahoo, T. K. Geok, Solving shortest path probelm using particle swarm optimization. Applied Soft Computing, 8(4)(2008), 1643-1653.

[6] J. H. Yang, X. H. Shi, M. Marchese, et al, An ant colony optmiaztion method for generalize TSP problem, Progress in Natural Science, 18(11)(2008), 1417-1422.

[7] Y. D. Zhang, L. N. Wu, S. H. Wang, et al, Improved ant colony algorithm base on membership cloud models, Computer Engineering and Applications, 47(14)(2011), 46-55.

[8] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, S. Dizdarevic, Genetic algorithms for the traveling salesman problem: A review of representations and operators. Artificial Intelligence Review, 13(2)(1999), 129-170.

[9] D. E. Rosenkrantz, R. E. Stearns, P. M. Lewis, An analysis of several heuristics for the traveling salesman problem, SIAM Journal on Computing, 6(1977), 563-581.

[10] [10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Optimization by simulated annealing, Science, Vol. 220, no.4598(1983), 671–680.

[11] G. W. Yang, W. M. Zheng, D. X. Wang, X. M. LI, An Algorithm for traveling Salesman Problem Using Deterministic Annealing, Journal of Software, 1 (1999), 57-59.

[12] M. Baranwal, B. Roehl, S. M. Salapaka, A deterministic annealing approach to the multiple traveling salesmen and related problems, http://arxiv.org/pdf/1604.04169.pdf, 4(2016).

[13] R. Eswarawaka, S. K. N. Mahammad, B. E. Reddy, Genetic annealing with efficient strategies to improve the performance for the NP-hard and routing problems, Journal of Experimental and Theoretical Artificial Intelligence, 27(2015), 779-788.

[14] S. N. Lan, W. G. Lin, Genetic algorithm optimization research based on simulated annealing, 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), (2016), 491-494.

[15] M. H. Yao, N. Wang, L. P. Zhao, Improved simulated annealing algorithm and genetic algorithm for TSP, Computer Engineering and Applications, 49(14)(2013), 60-65.

[16] M. Honjo, H. Iizuka, M. Yamamoto, Insertion-based Particle Swarm Optimization with local interaction, 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS), (2014), 755-760.

[17] B. C. Cheng, H. Y. Lu, X. P. Xu, W. Q. Shen, Improved local search-based chaotic discrete particle swarm optimization algorithm for solving traveling salesman problem, Journal of Computer Applications, 36(1)(2016), 138-142.

[18] L. G. Guzman, E. D. N. Ruiz, C. J. Ardila, A novel framework for the parallel solution of combinatorial problems implementing tabu search and simulated annealing algorithms, 6th International Conference on Computers, Communications and Control (ICCCC), 16 (2016), 259-263.

[19] C. Y. Wang, M. Lin, Y. W. Zhong, H. Zhang, Solving travelling salesman problem using multiagent simulated annealing algorithm with instance-based sampling, International Journal of Computing Science and Mathematics, Vol. 6, no. 4(2015), 336–353.