Proceedings of the International Conference on Industrial Engineering and Operations Management
Bangkok, Thailand, March 5-7, 2019

A New Metaheuristics for Solving Traveling Salesman Problem: Partial Comparison Optimization

Antono Adhi
Industrial Engineering Department
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia,
Industrial Engineering Department
Universitas Stikubank
Semarang, Indonesia
antonoadhi@edu.unisbank.ac.id

Budi Santosa, Nurhadi Siswanto
Industrial Engineering Department
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
budi_s@ie.its.ac.id, siswanto@ie.its.ac.id

Abstract

This research article proposes new metaheuristics method to solve Traveling Salesman Problem (TSP). This method is called Partial Comparison Optimization (PCO). TSP is defined as a problem where a salesman must visit all cities where each city is only visited once, and must start from and return to the origin city. The goal of solving this problem is to determine the route with minimum total distance or cost. TSP was first formulated in 1930 and it is one of the most intensively studied problems in optimization. Variants and various application of TSP have been developed and solved to accomodate industrial problems. TSP is an NP-hard combinatorial optimization problem. It means TSP can be solved in polynomial time. Exact methods are hard to solve big size TSP problem. The process of the exact method needs longer computational time to solve the problem. The limitation of exact method in dealing with complex TSP only can be solved by metaheuristics. PCO is powerful metaheuristic to solve combinatorial problems such as TSP. To test the performance of PCO, it was used to solve some TSPLIB instances. In this research PCO gave good optimum solution that almost close to the optimal solution of every TSPLIB instance.

Keywords
Traveling Salesman Problem, Metaheuristics, Partial Comparison Optimization

1. Introduction

Traveling Salesman Problem (TSP) is problem of a salesman to search shortest traveling distance for visiting every city and back to the origin city (Yan et al. 2017). Salesman has to search sequence of cities from origin city and back to the origin city. For the different permutation of cities sequence, a salesman will obtain different traveling distance. It become to the salesman to search shortest traveling. Cities in TSP are located in Euclidean plane and the distance between two cities is ordinary Euclidean distance (Wang et al. 2017). In a various objection, shortest traveling distance could be defined as minimum traveling cost.

TSP can be simply defined as a complete weighted graph $G = (V, E, d)$ from the perspective of graph theory (Wang, 2017). $V = \{1, 2, \ldots, n\}$ is a set of vertices (cities). $E = \{(i, j)|(i, j) \in V \times V\}$ is a set of edges, and d is a function assigning a weight (distance) d_{ij} to every edge (i, j) (Wang et al. 2017).

It is one the most widely and famous studied combinatorial problems (Ozden et al. 2017). It has several applications in its purest formulation, such as transportation, logistics, and planning. In the development, TSP is

applicated in manufacture of printed circuit board (Wang et al. 2017). it also appears as a sub-problem in many areas, including image processing, DNA sequence encoding , data clustering, protein function prediction, and so forth (Wang et al. 2017) . Constraints also be added in many applications, like limited resources or time windows may be imposed, such as vehicle routing problems and traveling purchaser problems (Wang et al. 2017). Nowadays, diversified applications require large-scale TSPs to be solved efficiently with acceptable optimum result (Wang et al. 2017).

Solution methods of TSP in any studies can be classified as Exact Algorithms, Heuristics, or Meta-Heuristics. Exact algorithms are guaranteed to obtain an optimal solution in a bounded number of steps. The dynamic programming algorithm and the branch-and-bound algorithm are some well known algorithms in this class. They are good for solving instances up to 60 cities. Enumeration algorithm is only good for solving small instances up to 10 cities. The limitation of exact algorithms is happened because TSP is NP-hard (Garey and Johnson 1979, Wang, 2017). Hence, for large-scale TSP instances, exact methods find optimal tours with a dramatic increase of execution time (Wang et al. 2017). The approaches that are able to generate near-optimal tours in a reasonable time are heuristics (Wang et al. 2017). Heuristics are methods which have statistical or empirical guarantee to find good solutions, but have no mathematical proofs of their effectiveness yet (Wang et al. 2017). Metaheuristics approaches as for example evolutionary algorithms (EA), tabu search, simulated annealing or ant colony systems are also widely used for the TSP (Wang et al. 2017).

This study proposed novel metahuristics method called Partial Comparison Optimization (PCO) to solve TSP. PCO has the same basic principle in processing with NEH algorithm (Nawaz et al. 1983) but PCO has better performance since it gives wider processing by iteration process that cannot be performed by NEH in order to achieve optimum solution.

2. Problem Statement

According to Ismail (2011), when V is a set of cities, S is a subset of V, and c_{ij} is the cost of moving from city i to city j, integer linear programming formulation of TSP can be defined as:

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \quad (1)$$

$$\sum_{i \in V} x_{ij} = 1, i \in V \quad (2)$$

$$\sum_{j \in V} x_{ij} = 1, j \in V \quad (3)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \forall S \subset \mathrm{V}, S \neq \emptyset \quad (4)$$

$$x_{ij} \in \{0, 1\}, \forall i, j \in V \quad (5)$$

x_{ij} = 1 if the path goes from city i to city j and 0 otherwise.

3. Partial Comparison Optimization

Partial Comparison Optimization (PCO) is a new metaheuristics method. PCO is used to solve optimization problems by finding the best solution of the alternatives under consideration. PCO solve discrete optimization problems that combine the sequence of elements. The best solutions achieved from PCO are optimal solutions such as minimum cost or shortest distance. One of the problems that can be solved by PCO is TSP. Although PCO is a new method in metaheuristics, PCO in its development is able to provide better results than other metaheuristics methods.

Trapped in local optimum as a constraint of optimization problems that become the central point of completion of metaheuristics can be done well by PCO. Trapped in local optimum causes metaheuristics do not give its best solution. Regard to its nature as a combinatorial optimization problem solver, PCO has its own advantage in terms of searching time, since PCO does not need to convert processes from continuous to discrete. PCO is effective in finding the optimum value. PCO will not be trapped in local optimum and makes possible obtain optimum global value. This is because the directional randomization process in PCO allows searching within the area of combination values. By the randomization process, radical shift of the combination will be possible. Therefore it will provide possibly new results that may be optimum. However, this randomization process is still controlled by guidelines that only optimum result is selected. The time required to produce optimum value is relatively not too long compared to other metaheuristics methods. There are options that can be selected to get the optimum value in faster iterations. However, this option will cause longer processing time.

PCO has basic guidelines that become the principle of the optimum value searching process. The guidelines are:

1. Random Choosing (RC)
   A set of element J={1,2,3,..., j} is union of element I ={1,2,3,..., i} and element K={1,2,3, ..., k}. A set of element I contains unprocessed element partially. A set of element K is sequence of element that will be processed in partial scheduling. RC step is choosing process of one element I to enter K. Choosing process is done one by one until all of element I enter to element K. The choosing is done randomly. This random choosing has a purpose to widen area of optimum point. This choosing process is controlled and observed by PC step. The bound area of the random choosing is still in scope of permutation element J. This random choosing will give new possibility when the partial process is repeated in the next iteration.
2. Partial Comparison (PC)
   In PC step, only element in K will be processed partially from all of elements in J. The randomly chosen element I_i from a set of element I will find the most optimum position in a set of element K. Element I_i will try to take a position in before, between, and after from a set of element K. Each position will compare its fitness value. The position that performs the best fitness will be selected. This process was performed as in the NEH algorithm (Nawaz et al., 1983).
3. Changing Neighborhood (CN)
   Based on experiments, for a position took by a new element, it will change the possibility of fitness when the position of neighbors next to it is swapped. Element I_i, that take position n, has neighbor element K_x in position n-1 and element K_y in position n+1. CN process will swap position of K_x and K_y. This change gives the chance of changing the fitness level better. If swapping process gives better fitness, then this process will be chosen as a better position. The received consequence is the step of CN will prolong processing time. But it will give the chance of the total searching process optimum results faster. Based on experiments, the time generated by the process of CN for obtaining more optimum result is relatively faster compared than not using this step. CN can be done more than one level position around the placed element I_i. The changing process performs not only in position n-1 and n+1, but n-2 and n+2.
4. Looping Process (LP)
   LP step will repeat the searching of optimum value by iteration. NEH algorithm cannot perform this LP because identification in the first process gives a permanent sequence. Different with NEH algorithm, obtaining the optimum fitness value for each iteration is possible achieved since PCO has RC step. From LP step, it will be obtained global optimum by comparison local optimum. Global optimum from a set of element K is the optimum result of PCO calculation.
5. Stopping Iteration (SI)
   PCO usually achieve optimum solution in early iteration. In the beginning, PC and CN are processed in probability 1 until effective iteration *ei*. *ei* is determined as iteration value of PCO usually it achieve optimum value. To reduce time processing, PC and CN step will be done if a generated random value is in maximum comparison probability *cp* after iteration *ei*. *cp* will be set between 0 and 1. *cp* value will be reduced after *ei* until maximum iteration as depicted in figure 1.

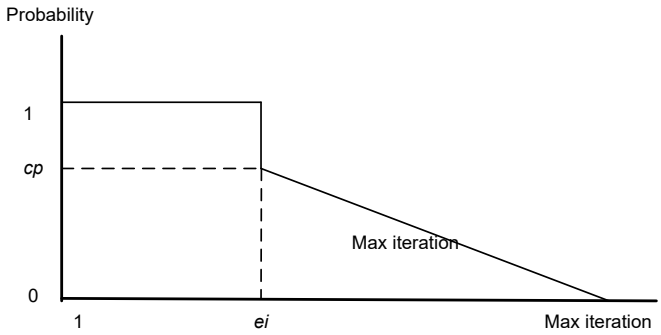

Figure 1. Effective iteration *ei* and comparison probability *cp* used for determining processing of PC and CN step.

Algorithm of PCO is as follow:

Step 1. Determine processed elements J={1,2,3,..., j}, maximum iteration *MaxItr*, effective iteration *ei*, and maximum comparison probability *cp*.

Step 2. Clear K. Get one element from J and put in K

Step 3. Get one element from *J*
Step 4. If iteration > *ei* then *prob* = *ce* – (*ce* × ((iteration – *ei*) / (*MaxItr* – *ei*))) else *prob* = 1
Step 5. Process every position of new element if random number *rand* < *prob*. Put the new element, before, between, and after the elements in *K*. Calculate fitness of new sequence with new element. If the new sequence has better fitness, set the new sequence as best partial sequence for the new element.
Step 6. Process every position of new element on this step if *rand* < *prob*. If the position of new element according to Step 5 is between two elements in *K*, swap the position of two elements and calculate fitness of new sequence with new element. If the new sequence has better fitness, set the new sequence as best partial sequence for the new element.
Step 7. Go to Step 5 until the new element searches all of the position in *K*.
Step 8. The last better partial sequence becomes the best position of new element in *K*.
Step 9. Go to Step 3 until all elements in *J* is put into *K*.
Step 10. *K* is the best local sequence for every iteration.
Step 11. Compare the best local sequence of every iteration. The better local sequence becomes the best global sequence.
Step 12. Add iteration. Go to Step 2 and repeat until *MaxItr*. The best global sequence is the best answer solution of PCO.

5. Experiments

To prove that PCO is a good metaheuristic for solving TSP, a set of benchmarks of symmetric TSP instances selected from the TSPLIB library are used to evaluate its performance against other heuristic algorithms. Four instances with moderate number of cities are generated on the basis of TSPLIB instances (Reinelt 1991) as shown on Table 1.

Table 1. Experiment instances

TSP	Number of Cities	Optimal Solution	Annotation
ulysses22	22	75.64	Odyssey of Ulysses (Groetschel and Padberg)
berlin52	52	7544.32	52 locations in Berlin (Germany) (Groetschel)
st70	70	678.55	70-city problem (Smith/Thompson)
pr76	76	108159.4	76-city problem (Padberg/Rinaldi)

The data from TSPLIB instances only the coordinate of every citiy. They do not show eucledian distance of every city from other cities therefore optimal solution of the TSPLIB instances is recalculated from its optimum traveling path.

6. Results and Discussions

The result of calculation by PCO used to solve the problem of TSP from TSPLIB shown as in Table 2.

Table 2. Result of calculation

TSP	Optimal Solution	PCO Optimal Solution	PCO Optimum Tour
ulysses22	75.64	75.29	1 14 13 12 7 6 15 5 11 9 10 19 20 21 16 3 2 17 4 18 22 8 1
berlin52	7544.32	7544.63	1 22 31 18 3 17 21 42 7 2 30 23 20 50 29 16 46 44 34 35 36 39 40 37 38 48 24 5 15 6 4 25 12 28 27 26 47 14 13 52 11 51 33 43 10 9 8 41 19 45 32 49 1
st70	678.55	682.36	1 16 47 37 58 50 51 65 64 11 56 67 48 54 62 33 34 21 12 60 52 10 5 53 6 41 43 17 9 40 61 39 45 25 46 27 68 44 30 20 14 28 49 55 26 8 3 32 42 18 4 2 7 19 24 15 57 63 66 22 38 59 35 69 31 70 13 29 23 36 1
pr76	108159.4	109101.7	1 76 75 2 4 3 7 8 6 5 10 9 12 13 14 74 15 16 11 17 18 37 36 38 39 40 34 35 33 32 19 20 31 30 29 26 27 28 43 42 54 53 52 55 56 57 58 59 41 60 61 62 63 64 73 72 71 65 66 51 49 50 67 70 68 69 47 48 44 45 46 24 25 21 22 23 1

Table 2 shows that PCO has good solution to solve TSP. The result almost close to the optimal solution of every TSPLIB instance. Even one of the solution of PCO has better solution than optimum tour declared by TSPLIB instance. PCO optimum tour (1 14 13 12 7 6 15 5 11 9 10 19 20 21 16 3 2 17 4 18 22 8 1) give 75.25 compared with TSPLIB instance tour (1 14 13 12 7 6 15 5 11 9 10 19 20 21 16 3 2 17 22 4 18 8 1) that give result 75.64 in ulysses22 data.

7. Conclusion

PCO is new metaheuristic proposed to solve combinatorial problem such as TSP. PCO has high capability to give optimum solution in order to search shortest distance of TSP. In future research, PCO can be improved in the time of calculation.

References

Ismail, M.A., Mirza, S.H., Altaf, T., A parallel and concurrent implementation of Lin–Kernighan heuristic (LKH-2) for solving traveling salesman problem for multi-core processors using SPC^3 programming model, *Int. J. Adv. Comput. Sci. Appl*., vol. 2, pp. 34–43, 2011.

Nawaz, Muhammad, Enscore Jr., E. Emory, Ham, Inyong, Heuristic algorithm for the m-machine, n-job flow shop sequencing problem, *Omega*, vol. 11, pp. 91-95, 1983.

Ozden, S.G., Smith, A.E., Gue, K.R., Solving large batches of traveling salesman problems with parallel and distributed computing, *Computers and Operations Research*, vol. 85, pp. 87–96, 2017.

Reinelt, G., A Traveling Salesman Poblem Library, *ORSA Journal on Computing,* vol. 3, pp. 376–384, 1991.

Wang, Hongjian, Zhang, Naiyu, Créput, Jean-Charles, A massively parallel neural network approach to large-scale Euclidean traveling salesman problems, *Neurocomputing*, vol. 240, pp. 137–151, 2017.

Wang, Yongzhen, Chen, Yan, Lin, Yan, Memetic algorithm based on sequential variable neighborhood descent for the minmax multiple traveling salesman problem*, Computers & Industrial Engineering*, vol. 106, pp. 105–122, 2017.

Yan, Yuzhe, Sohn, Han-suk, Reyes, German, A modified ant system to achieve better balance between intensification and diversification for the traveling salesman problem, *Applied Soft Computing*, vol. 60, pp. 256–267, 2017.

Biography / Biographies

Antono Adhi is a student of Doctoral Program in Industrial Engineering, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. He works as a lecturer in Universitas Stikubank (UNISBANK), Semarang, Indonesia. He got his Diploma from Informatic Department Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. He got master degree from Industrial Engineering, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia and Magister Manajemen from Universitas Stikubank (UNISBANK), Semarang, Indonesia.

Budi Santosa is a professour of Industrial Engineering Department. He is a lecture of Doctoral Program in Industrial Engineering Department, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. He earned Bachelor degree from Industrial Engineering Institut Teknologi Bandung, Indonesia, Masters and Doctoral degree from University of Oklahoma, USA. His concentrations are in optimization and data mining.

Nurhadi Siswanto is a lecture of Doctoral Program in Industrial Engineering Department, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. He is the head of Department of Industrial Engineering, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia.