Domino algorithm: a novel constructive heuristics for traveling salesman problem

To cite this article: Asrul Harun Ismail 2019 IOP Conf. Ser.: Mater. Sci. Eng. 528 012043

View the article online for updates and enhancements.
Domino algorithm: a novel constructive heuristics for traveling salesman problem

Asrul Harun Ismail

1The Department of Mechanical Engineering, University of Birmingham, Edgbaston, Birmingham B15 2TT, United Kingdom
AHI729@student.bham.ac.uk

Abstract. Developing algorithms for solving complex optimisation problems has become a challenging topic recently. This study has applied a novel constructive heuristics algorithm named Domino Algorithm for the Traveling Salesman Problem (TSP) case which is aimed to efficiently reduce the calculation complexity and to find the optimal results of TSP best solution of tour lengths. As the study is a basic version of Domino Algorithm, it is decided to use the small TSP data sets consisting of 100 cities or less, such as Eil51, Berlin52, St70, Eil76, Pr76, and Rat99. This study also applied Nearest Neighbor approach to compare its result with that of Domino Algorithm. The results have shown that better tour lengths were achieved by Domino Algorithm (using 1 or 2 player(s)) for Eil51, Eil76, St70, Rat 99; and were resulted by Nearest Neighbor for Berlin52, and Pr76. For further research, the algorithm should be intensively combined with applying the improvement heuristic and should also be hybridized with meta-heuristics algorithm focusing on finding the optimal results by which the application will be moreover quite simple and easy.

Keywords: Constructive heuristic algorithm, domino algorithm, nearest neighbor algorithm, traveling salesman problem.

1. Introduction

Undoubtedly, the most prominent member of the rich set of combinatorial optimization problems is the traveling salesman problem (TSP). TSP is comprised of finding the shortest closed tour, Hamilton cycle, or visiting N cities. It is an NP-hard problem in combinatorial optimization whose computation will naturally rise exponentially by expanding the quantity of cities [1], that is important in operations research and theoretical computer science.

TSP is a complicated or a NP-complete problem that cannot be solved by using traditional algorithms. This problem consists of N cities and between every two cities there can exist a connections (paths) which every path has specified distance or cost. The salesman wants to start from one city and then proceed with other cities in a way that he will pass through each city for only once and finally, he will return to the origin point. The aim of TSP is finding the sequence of the cities to be traveled through by the traveling salesman that will result in a reduced total distance or cost.

TSP can be solved either with exact or heuristic algorithm. Although it is easy to understand traveling salesman problem, but if the size is getting larger, its solution using exact algorithm will become very difficult. In term of quality and speed, solving the problem using heuristic or meta-heuristic algorithms
is more feasible. Discovering a suboptimal solution with a sensible time or cost might be more beneficial rather than finding an optimal solution with time wasted. Various heuristic and meta-heuristic algorithms resulting in quick and good solutions have been devised. Modern heuristic or meta-heuristic methods can find solutions for extremely large problems (millions of cities) in a reasonable time with only a gap of 2–3% from the optimal solution.[2]

Some of the meta-heuristic algorithms which are used to solve such problems are genetic algorithm, simulated annealing algorithm, bee algorithm, etc. Furthermore, most of the searching step, global and local search of the meta-heuristic algorithms are using heuristic algorithms either the constructive heuristics or improvement ones. There are several popular constructive heuristic algorithms in combinatorial domain such as nearest neighbor, nearest arbitrary city, nearest insertion, farthest insertion, and cheapest insertion.

These algorithms are also already used widely and some of its application are like Machine scheduling problem, Cellular manufacturing, Arc routing problem, Frequency assignment problem, Structuring of matrices, Printed-circuit-boards manufacturing, data transmission in computer networks, power-distribution networks, image processing and pattern recognition, robot navigation, and data partitioning[3],[4].

To date, a few approaches based on deterministic or probabilistic heuristics have been proposed for optimizing TSP and all of these algorithms have to use constructive heuristics algorithms to generate prospective solution(s) as the initial good solution(s). Later, the improvement heuristics will enhance those solutions to get closer to the optimal solution. For a large-size TSP case, it is almost impossible to find an optimal or near-optimal solution using constructive heuristic alone. However, for several small-size TSP cases (less than 50 cities), the constructive heuristics mentioned-above might provide an optimal solution.

In constructive heuristics research, reaching an optimal solution becomes less important compared to resulting a great seed leading to near-optimal solution by providing the randomness of prospective solutions. In term of generating initial solutions, the constructive heuristics also give a faster calculation rather than using random permutation. Population-based heuristics such as ant colony optimization (ACO) algorithms [5],[6],[7],[8], neural networks [3],[9], evolutionary algorithms [10],[11], simulated annealing [12],[13],[14],[15], tabu search [16],[15], artificial bee colony algorithm [17],[18], particle swarm optimization [19],[18], and Bee Algorithm [20],[21],[22],[23] are also using constructive heuristics to solve combinatorial problems by employing tour construction. Hence, building up a constructive heuristics algorithm that generates good random solutions to become as good seed and efficient complex computation remains a challenging topic, especially when it can be used for developing a meta-heuristic algorithm for generating sub-optimum or optimum solution.

In this paper, a novel constructive heuristic “Domino Algorithm” is proposed to provide better seed to be improved. Most of current constructive heuristics are using a “greed” strategy. It makes them provide limited number of prospective solutions and hard to reach optimum solution by improving heuristics on larger-size TSP [24]. The main idea of proposed approach is derived from the domino game. The traditional Sino-European domino game is featuring all combinations of spot counts between zero and six (see figure 1), which generates a sequence of the tiles. Each tile represents a path consisting 2 points as original point and destination one. The game will be over when the sequence gets formed or the player turns down all the tiles. This basic version of proposed algorithm will order each player to make turn and each tile selected from a player’s hand should become the minimum distance and be inserted in the matching edge of the tiles sequence. Although the selection and insertion strategy could be exchanged to the others, but this paper will focus on the basic version strategy.
The main part of this paper will consist of the literature review and the result of experiment that will compare the proposed method with the nearest neighbor since it is very popular and widely used as constructive heuristics. The TSPLIB’s data sets are used as problem sets to compare their performances. The comparison between proposed approach and nearest neighbor seems fair enough to provide comprehensive experiment result – compare to other constructive methods – since both of them use the same selection and insertion strategy. As it is a basic version of Domino Algorithm so we choose to use the small TSP data set, 100 cities or less, such as Eil51, Berlin52, St70, Eil76, Pr76, and Rat99 [25],[26]. This paper will be organized as follows. In next section, we will present the literature review. Proposed Domino Algorithm as a basic version will be presented in section 3. Section 4 will focus on the evaluation of the performance of Domino Algorithm compare to its counterparts using six sets of TSP problem (TSPLIB) as mentioned above [25]. The 2-opt, the improvement heuristics, is used in this experiment to prove that the proposed algorithm has generated a better initial solution as a seed. Finally, conclusions will be presented in section 5.

2. Literature review

2.1. Traveling salesman problem

Combinatorial problems has arisen in many areas of computer science and other disciplines in which computational methods are applied, such as artificial intelligence, operations research, bioinformatics and electronic commerce. Some popular problems are like finding shortest tour also known as the travelling salesman problem, finding models of propositional formulae and determining the 3D-structure of proteins. Other well-known combinatorial problems are found in planning, scheduling, time-tabling, resource allocation, code design, hardware design and genome sequencing [27]. Some of combinatorial problems are N-P Hard type. It is generally believed that N-P hard problems cannot be solved to optimality within polynomial bounded computation times. Consequently, it has increased an interest in approximation algorithm that can find near-optimal solution with reasonable running times.
TSP can be symmetric or asymmetric. In the symmetric one, the distance between the two cities does not depend on the distance of travel. For example, if we show the distance between two nodes called i and j with \( d_{ij} \) and if we have \( d_{ij}=d_{ji} \), then TSP is symmetric or otherwise it is asymmetric.

2.2. Nearest neighbor

There are three types of heuristics method i.e. constructive, improvement, and compound. The constructive starts from building a solution with step by step, based on a set of rules defined beforehand. This approach is also used by meta-heuristics in their global search calculation. The improvement heuristics starts from a feasible solution and improve it by applying successive small changes. There are several popular algorithms such as 2-opt, 3-opt, k-opt, etc. Meanwhile, the compound heuristics has both constructive and improvement phases. Normally, the improvement phase will be conducted following the constructive phase. Hence, the compound is usually built from a combination of constructive and improvement, such as nearest neighbor-2Optimal, nearest neighbor-3Optimal, etc.

Nearest neighbor belongs to constructive heuristics which starts from nothing to then build a prospective solutions. Today, there are several popular constructive heuristic algorithm in combinatorial domain such as nearest neighbor, nearest arbitrary city, nearest insertion, farthest insertion, and cheapest insertion. The nearest neighbor algorithm is one of the first algorithms used to determine a solution to the travelling salesman problem. In such the algorithm, the salesman starts at a random city and proceeds with visiting the nearest city until all cities are finished to visit. It quickly yields a short tour, but usually not the optimal one. The nearest neighbor algorithm is easy to implement and execute quickly, but it can sometimes miss shorter routes which are easily noticed with human eyesight, due to its "greed" nature [24]. The procedure follows is the nearest neighbor algorithm [1] where \( V = \{1, 2, \ldots, n\} \) is a set of nodes representing the cities and \( E = \{(i, j) \mid i, j \in V\} \).

1. Select an arbitrary node \( k \).
2. Find city \( k+1 \) that is not yet visited and closer to \( k \) (min \( d_{ij} \)). Let \( T=\{1, 2, \ldots, k\} \) be the current partial tour (\( k<n \)).
3. Insert \( k+1 \) at the end of the partial tour; and \( k = k+1 \).
4. If all nodes or cities are already inserted then STOP, else go back to step (2).

3. Proposed Algorithm

Basically this proposed method is using the idea of domino game. This game is a traditional tile-based strategy game usually played by 2 to 4 players. Each domino tile is a rectangular shape with a line parting its face in half into two equal square-shaped: front half and end one. Each half is marked with either a blank, one spot, or a number of spots. The game is played this way. A first player will select a tile, then the next player will proceed with selecting a matching tile having a similar value in its front or end half with the tile of the first player to then make a sequential set of matching tiles. They will in turn play this way until the end of the game. A domino set is a generic gaming device, similar to playing cards or dice, in which a variety of games can be played with a set. Generally this game has six main steps: Shuffle the dominoes, Draw open hand, Decide the order of play, Lay first domino, Take turn adding domino, and end the round and award points. This rule will be implemented to solve a TSP case.

Domino Algorithm generates a sequence from a set of matching tiles which each tile represents a path consisting 2 points as original and destination cities. The tiles used in the game are the combinations of spot(s) represent a number of cities between 1 to N-cities. Fig 2 shows the combination of tiles (excluding the zero value represented by the blank square) that is used in 6 cities’ TSP case. The algorithm consists of 4 common steps of constructive heuristics: initialization, selection, insertion, and termination. The first step is distributing the tiles equally to each player in a random. Players make turns to add tiles into the front half or into the end half of the initial tile. The game will be over after a sequence gets formed or the players turns down all the tiles. This algorithm will order each player to make turn and each tile selected from player’s hand should become the minimum distance. The Fig 3 below is showing the flow chart of Domino algorithm procedure.
4. Methodology

4.1. Problem Representation

TSP can be represented as a graph of \( G = (V, E) \), where \( V = \{1, 2, \ldots, n\} \) is a set of nodes representing the cities, and \( E = \{ (i, j) | i, j \in V \} \) is the set of all connecting edges between them. Each edge means the possible path between two connected cities. The distance \( d_{ij} \) or \( d(i,j) \) is associated with edge \((i, j)\) and represents the Euclidean distance from city \( i \) to city \( j \) as equation (1). The information is calculated before performing the main algorithm. The distances of all edges were calculated and saved as a distance matrix.

\[
d_{ij} = \sqrt{(x_i - x_j)^2 - (y_i - y_j)^2}
\]  

The TSP is the problem of finding a minimum length of Hamiltonian tour on the undirected graph \( G = (V, E) \), where an Hamiltonian tour of graph \( G \) is a close-distance tour that needs visiting all the nodes of \( G \) one time only, and its length is given by the sum of the lengths of all the edges by which it is composed. The aim of solving TSP is to minimize the total close tour length \( L \) which is defined as equation (2).

\[
L = \sum_{i=1}^{n} d(i, i+1) + d(n, 1)
\]

4.2. Initialization

To solve TSP using Domino algorithm we should conduct some following steps. First, we generate some distributed initial random tiles by firstly setting up the number of players \( p \), followed by shuffling the tiles up, then distributing the initial set of tiles to each player in equal number or \( n/p \) or it will lead to a little adjustment when the number of nodes is odd number.

4.3. Selection and Insertion

In this manner a random city is selected as initial of the tour (e.g. city \( i \)). Then the next cities \( k \) consecutively are added to the tour which corresponds to their distance from city \( i \). Let \( (1, \ldots, k) \) be the current partial tour \( (k < n) \). The new city \( k \) will be added in the end of the tour if \( d(k,k+1) < d(k+1,k) \), otherwise it will be added or inserted at the beginning of partial tour. The players have to extend the partial tour with their remaining tiles (nodes) in turn.

4.4. Termination

When the next city \( k \) is equal to number of nodes (cities) then it will be the last insertion on the algorithm. The algorithm will calculate the final close tour or Hamilton cycle using Eq. (2). The figure 2 below depicts the flowchart of Domino algorithm where \( V = \{1, 2, \ldots, n\} \) is a set of nodes representing the cities, \( P = \{1, \ldots, p\} \) is a number of players, \( V_p = \{1, \ldots, n/p\} \) and \( E = \{ (i, j) | i, j \in V \} \).

5. Experimental result

In order to evaluate the performance of Domino algorithm, we tested it in the benchmark problems of TSPLIB. The experimental results are summarized in table 1. The first and 2nd column is the problem names and optimum tour lengths (best known solution), 3rd column is the values of the compared algorithms, and 4th and 5th columns are the achieved values by proposed method. Each 3rd, 4th, 5th columns have sub-columns which respectively are the best tour length generated by those methods, the average value of tours length possibly generated, and the number of possibly generated tours. The possible generated tour achieved by Nearest Neighbor is always as much as the-N (number of nodes) since it uses the minimum distance. The Domino Algorithm with 1 player always produces possible tours as much as those of Nearest Neighbor since there is no other players in turn generating sequence. We could also say that every player’s combination of nodes generates \( n \) possible constructed tours and in this case, a player only has \( C(n, n) = 1 \). However Domino Algorithm with 2 players generates more tours since each of them has the same number of nodes (n/2) or \( V_p = \{1, \ldots, n/2\} \) and having their own turn to build the sequence. The number of possible constructed tours of proposed algorithm with two
players is $\frac{n}{2} \cdot C(n, \frac{n}{2})$ where $C(n, \frac{n}{2})$ is a number of nodes combination of 2 players. We assume that the larger number of constructed tours, the better solution will be produced when it is enhanced with the improvement heuristic methods. The experiment uses 1000 iterations, which neither efficient nor effective for Nearest Neighbor and Domino with 1 player, and random number as a seed (initial city) for each method.

**Initialization**
- Set the problem (n of cities and distance matrix).
- Set the p (number of players).
- Distribute n/p tiles each player randomly.
- select initial one city k from randomly chosen.

**Selection**
- Find city $k + 1$ every player’s turn that is not yet in the tour and that is nearest from $k$ or to $k$.
Let $T = (1, ..., k)$ be the current partial tour ($k < n$).

**Insertion**
- Insert $k + 1$ at the beginning of the partial tour.
- Insert $k + 1$ at the end of the partial tour.

no

$dk,k+1 < dk+1,k$ ?

yes

Figure 2. The flowchart of Domino algorithm.

As comparison to the performance shown with a graph, the best solution every approach has made is summarized in table 2. All the best tours still have crossing lines or unsmooth shapes because of the prior research and the exclusion of the improvement heuristics. We believed in further research the problems will be eliminated by using improvement heuristic methods. A better constructed tour length has been achieved by Domino Algorithm (using 1 or 2 player(s)) for Eil51, Eil76, St70, Rat 99; and Nearest Neighbor for Berlin52, and Pr76 results can be seen respectively in table 1.

**Table 1.** Comparison value between best solution tour length of Nearest Neighbor and Domino Algorithm

| Problem | Optimal Solution | Nearest Neighbor (NN) | Domino-2 players (Dom2pl) | Domino-1 players (Dom1pl) |
|---------|------------------|-----------------------|----------------------------|---------------------------|
|         |                  | Best | Average | Best | Average | Best | Average | Best | Average |
| Eil51   | 426              | 482  | 525.90  | 507  | 622.63  | 469  | 540.34  |
| Berlin52| 7542             | 8181 | 9375.58 | 9343 | 11293.75| 8541 | 9226.44 |
| St70    | 675              | 796  | 841.26  | 854  | 1067.32 | 748  | 822.52  |
| Eil76   | 538              | 608  | 665.75  | 654  | 796.44  | 583  | 654.07  |
| Pr76    | 108159           | 130921| 147178.41 | 143351 | 181450.47 | 137788 | 143343.23 |
| Rat99   | 1211             | 1437 | 1535.83 | 1509 | 1946.40 | 1372 | 1513.58 |
Table 2. Comparison graph between best solution of tour lengths between Nearest Neighbor and Domino Algorithm

| Problem | Nearest Neighbor (NN) | Domino-2 players (Dom2pl) | Domino-1 players (Dom1pl) |
|---------|-----------------------|---------------------------|---------------------------|
| Eil51   | 482                   | 507                       | 469                       |
| Berlin52| 8181                  | 9343                      | 8541                      |
| St70    | 796                   | 854                       | 748                       |
| Eil76   | 608                   | 654                       | 583                       |
| Pr76    | 13021                 | 143315                    | 137788                    |
| Rat99   | 1437                  | 1509                      | 1372                      |

Eil51 (Opt. Solution=426)
Berlin52 (Opt. Solution=7542)
St70 (Opt. Solution=675)
Eil76 (Opt. Solution=538)
Pr76 (Opt. Solution=108159)
Rat99 (Opt. Solution=1211)
6. Conclusion
In this paper, a novel constructive heuristics algorithm named Domino algorithm has been introduced for TSP, which is aimed to efficiently reduce the calculation complexity of meta-heuristics algorithms which can generate \( \frac{n!}{2} \cdot C(n, \frac{n}{2}) \) constructed tours as good prospective solutions. Using random permutation in global search of meta-heuristics will waste the time since it generates \( n! \) of prospective solutions.

For further research, the algorithm should be intensively combined with applying the improvement heuristic and hybridize it with meta-heuristics algorithm focusing on finding the optimal results which will be moreover quite simple and easy to apply.

Acknowledgement
The authors would like to acknowledge the generous support from the Newton Fund - Ristekdikti.

7. References
[1] Reinelt G 1994 Traveling Salesman: Computational solution The TSP applications. (Springer-Verlag, Berlin).
[2] Rego C, Gamboa D, Glover F and Osterman C 2011 Traveling salesman problem heuristics: Leading methods, implementations and latest advances. Eur. J. Oper. Res. vol 211 no 3 pp 427–41
[3] Saadatmand-Tarzjan M, Khademi M, Akbarzadeh-T M R and Abrishami Moghaddam H 2007 A novel constructive-optimizer neural network for the traveling salesman problem. IEEE Trans. Syst. Man, Cybern. Part B Cybern. vol 37 no 4 pp 754–70
[4] Gutin A P, Punnen G 2006 The Traveling Salesman Problem and Its Variations. (Springer).
[5] Dorigo M and Gambardella L M 1997 Ant colonies for the travelling salesman problem. Biosystems. vol 43 no 2 pp 73–81
[6] Dorigo M and Gambardella L M 1997 Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Trans. Evol. Comput vol 1 no 1 pp 1–24
[7] Gambardella L and Dorigo M 1997 Ant-Q: Reinforcement Learning Approach to the Traveling Salesman Problem. IEEE Trans. Evol. Comput. vol 1 no 1 pp 252–60
[8] Liao E and Liu C 2018 A Hierarchical Algorithm Based on Density Peaks Clustering and Ant Colony Optimization for Traveling Salesman Problem. IEEE Access. vol 6 pp 38921–33
[9] Creput J and Koukam A 2009 A memetic neural network for the Euclidean traveling salesman problem. Neurocomputing. vol 72 pp 1250–64
[10] Vaishnav P, Choudhary N and Jain K 2017 Traveling Salesman Problem Using Genetic Algorithm: A Survey. Int. J. of Scientific Research in Computer Science, Engineering and Information Technology. vol 2 no 3 pp 105–8
[11] Hussain A, Muhammad Y S, Sajid M N, Hussain I, Shoukry A M and Gani S 2017 Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator. Hindawi Computational Intelligence and Neuroscience. vol 2017 pp 1–7
[12] Mafarja M M and Mirjalilí S 2017 Hybrid Whale Optimization Algorithm with simulated annealing for feature selection. Neurocomputing vol 260 pp 302–12
[13] Wang C, Lin M and Zhong Y 2016 Swarm simulated annealing algorithm with knowledge-based sampling for travelling salesman problem. Int. J. Intelligent Systems Technologies and Applications. vol 15 no 1 pp 74–94
[14] Helshani L 2016 Solving the Traveling Salesman Problem using Google Services and Simulated Annealing Algorithm. European Academic Research. vol IV no 3 pp 2321–30
[15] Lin Y, Bian Z and Liu X 2016 Developing a dynamic neighborhood structure for an adaptive hybrid simulated annealing – tabu search algorithm to solve the symmetrical travelling salesman problem. Appl. Soft Comput. J. vol 49 pp 937–52
[16] Ali Z A 2016 Concentric Tabu Search Algorithm for Solving Traveling Salesman Problem (Eastern Mediterranean University January-North Cyprus)
[17] Sabet S, Farokhi F and Shokouhifar M 2013 A Hybrid Mutation-based Artificial Bee Colony for Traveling Salesman Problem. *Lect. Notes Inf. Theory*. vol 1 no 3 pp 99–103
[18] Neelima S, Satyanarayana N and Murthy P K 2016 A Comprehensive Survey on Variants in Artificial Bee Colony. *Int. J. of Computer Science and Information Technologies*. vol 7 no 4 pp 1684–89
[19] Shi X H, Liang Y C, Lee H P, Lu C and Wang Q X 2007 Particle swarm optimization-based algorithms for TSP and generalized TSP. *Inf. Process. Lett.* vol 103 no 5 pp 169–76
[20] Pham D T, Otri S and Darwish A 2007 Application of the Bees Algorithm to PCB assembly optimisation. *3rd Virtual Int. Conf. Innov. Prod. Mach. Syst.* pp. 511-16
[21] Ang M C, Pham D T, Soroka A J and Ng K W 2010 PCB assembly optimisation using the bees algorithm enhanced with TRIZ operators. *IECON Proc. Industrial Electron. Conf.* pp 2708–13
[22] Ahmad S A, Pham D T, Ng K W and Ang M C 2012 TRIZ-inspired asymmetrical search neighborhood in the bees algorithm *Proc. - 6th Asia Int. Conf. Math. Model. Comput. Simulation, AMS 2012*. pp 29–33
[23] Otri S 2011 Improving The Bees Algorithm for Complex Optimisation Problems. (Cardiff-United Kingdom)
[24] Gutin G, Yeo A and Zverovich A 2002 Traveling salesman should not be greedy : domination analysis of greedy-type heuristics for the TSP. *Discrete Applied Mathematics*. vol 117 pp 81–6
[25] Reinelt G 1991 TSPLIB-- A traveling salesman problem library. *ORSA Journal on Computing*. vol 3 no 4 pp 376–84
[26] Reinelt G 1995 *Tsplib 95* (Universit¨at Heidelberg- Heidelberg) pp. 1–16
[27] Hoos H and Stutzle T 2005 *Stochastic Local Search: Foundations and Applications*. (Morgan Kaufmann Publishers)