University of Ibadan
Journal of Science and Logics in ICT Research

Approximation Algorithm for Travelling Salesman Problem

[1]✉ Oladele R. O. and. [2]Mobolaji D. M.

Department of Computer Science University of Ilorin, Ilorin, Nigeria
roladele@unilorin.edu.ng[1], daremobolaji@gmail.com[2]

Abstract
Designing approximation algorithms often involves, among other things, relaxing the integrality constraint and obtaining a convex relaxation of the problem. The most well-known relaxation schemes are the Linear Programming (LP) relaxation and Semi-Definite Programing (SDP) relaxation. While LP relaxation has been widely used for solving TSP, SDP has been rarely employed. The primary goal of this paper therefore is to employ SDP and develop approximation algorithm for metric TSP. The SDP relaxation of the TSP is first obtained, and the approximation algorithm is thereafter developed. When compared to optimal results of some standard TSP instances, implementation result showed a relatively fair performance.

Keywords: Hamiltonian ath, Linear Programming, Relaxation, Semi-Definite Programming

1. Introduction

The traveling salesman problem (TSP) is a combinatorial optimization problem with the task to find the shortest route visiting all cities and returning to the starting point given a set of cities along with the cost of travel between them. Formally, TSP can be described by means of graph theory, as an undirected weighted network, whose nodes represent cities and whose links have certain weights indicating distances between those cities. In this context, finding the optimal solution of TSP means finding the shortest path that connects all nodes within the network. Given a complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$ .Find a Hamiltonian cycle of G with minimum cost (Traub, [9]).

Numerous concepts in combinatorial optimization, including T-Joins, Matroids, and Network Flows, are related to TSP, it also has various practical applications in vehicle routing problems, logistics, and scheduling. Over the years, TSP has been tackled by three classes of methods. One of these classes is heuristics which involves developing intuitive algorithms that run in polynomial time. The drawback of this class is that there is no guarantee on the quality of solution. Another class of methods employed to solve TSP is brute-force, this involves designing a clever enumeration strategy. This approach guarantees the optimality of the solution but does not return solution in polynomial time. Approximation algorithms are the third class of methods. They run in polynomial time and are guaranteed to produce high-quality solutions.

Relaxing the integrality constraint and getting a convex relaxation of a problem are two common steps in the design of approximation algorithms. The Linear Programming (LP) relaxation and Semi-Definite Programming (SDP) relaxation are the most well-known relaxation techniques. SDP has only seldom been used to solve TSP, on the other hand LP relaxation has been utilized often. This paper's main objective is to use SDP and create an approximation algorithm for metric TSP.

2. Related Works

A new, strongly polynomial-time algorithm and improved analysis for the metric s-t path TSP was presented in Sebő & van Zuylen, (2016).The algorithm can be viewed as an improvement on christofides algorithm. Removing some edges from christofides algorithm is a new idea introduce in the algorithm which is accompanied with parity correction of disconnected edges and then reconnecting the disconnected tree edges. Other concept that was introduced includes flow theory for assessing the reconnection cost and the creation of a collection of increasingly

Oladele R. O. and. Mobolaji D. M. (2025). Approximation Algorithm for Travelling Salesman Problem. *University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR)*, Vol. 14 No. 1, pp. 38 - 43

constrained spanning trees that can all be found using the greedy method. The approximation factor and integrality ratio were both improved below 1.53 by the introduction of this new methodology.

Major progress towards a constant-factor approximation algorithm was made by Svensson *et al.*, [7]. The author devised an algorithm for the special case for node-weighted instances. In a node-weighted instance, there is a nonnegative weight cv for every vertex v and the cost of an edge (v, w) ∈ E is given as cv + cw. Such an instance is not necessarily symmetric since G does not need to be a complete graph. Node- weighted instances are essentially equivalent to unit-weight instances.

In a recent breakthrough, (Svensson *et al.*, [8]) devised the first constant-factor approximation algorithm for the general ATSP and they also proved that its standard LP relaxation has constant integrality ratio. The approach is based on reduction to subtour-partition Cover an easier problem obtained by significantly relaxing the general connectivity requirements into local connectivity conditions. The author proved that any algorithm for subtour partition cover can be turned into an algorithm for ATSP while only losing a small constant factor in the approximation ratio. The algorithm gave a 506-approximation ratio for ATSP and an upper bound of 319 on the integrality ratio. Traub, [10] presented an improved and slightly simplified version of their algorithm. The operation of the algorithm is based on five tests. An approximation ratio of (22 + ε) (for any fixed ε > 0) was achieved. Furthermore, the author showed that the integrality ratio of the standard LP relaxation is at most.

Evnin [2] described an approximation algorithm for the maximum traveling salesman problem using Held-Karp LP relaxation for TSP, based on two known polynomial time approximation algorithms. The algorithm creates three tours and chooses the one with the highest weight. The approach calculates a maximum Hamiltonian path on the short cycle's vertices for each short cycle. The algorithm removes a minimum-length edge from each lengthy cycle. The resulting Hamiltonian route was combined to produce a tour T1, and the algorithm's approximation factor is 25/33. The computation of the maximum 2-matching and the computation of the maximum Hamiltonian routes on the subgraphs caused by the short cycles take the most time in the procedure. By using the dynamic programming method for the subgraph created by l vertices, the first and second may both be computed in polynomial time.

For the s-t-path TSP, Traub and Vygen (2019) suggested a 3/2 approximation technique. The method predicts lonely cuts and edges using dynamic programming. The instance was then divided into smaller instances in order to reinforce the LP, with busy cuts requiring a value of at least three. A spanning tree (V, S) and an LP solution y such that y is in the T-join polyhedron were computed by setting up a k-stage recursive dynamic program, where T is the collection of vertices whose degree in S has the wrong parity.

Zenklusen [11] developed approximation for Path TSP. The algorithm was analyzed using a high level, analysis similar to Wolsey's classical analysis for TSP. The algorithm starts by constructing a spanning tree T of $G = (V, E)$. Through parity correction of T, adding the shortest QT-join, and shortening the eulerian path discovered, a 1.5-approximation was achieved. By using a seminar result from Karger on the quantity of close-to-minimum cuts, the author demonstrated that a variation of the dynamic programming concept recently introduced by Traub and Vygen is adaptable enough to handle greater dimension cuts. This prevents using dynamic programming in a recursive manner, as Traub and Vygen did, and results in a much simpler approach that doesn't add an extra error term to the approximation ratio.

A polynomial - time approximation algorithm was designed for s-t-path graph TSP. The created approach can be seen as a stronger lower bond version of Sebo and Vygen's 3/2-approximation algorithm [6]. A stronger lower bound, a new sort of ear-decomposition, an improved ear induction that shows a fresh relationship to matroid union, and a reduction of general cases to situations when s and t are close together are all part of the refinement (which works for general metric).

The Travelling Salesman Problem with Neighbourhood (TSPN) is an extension of the

traditional travelling salesman problem (TSP) that has received substantial attention in the fields of combinatorial optimization and computational geometry. In the TSPN framework, the goal is to find a minimum-length closed tour that covers a group of areas or neighborhoods rather than specific places. This option requires the tour to intersect each neighborhood at least once, adding another layer of computing complexity to the operation. To tackle this problem, Ghaseminia, and Salavatipour [12] presented a Polynomial-Time Approximation Scheme. The PTAS algorithm returns a (1 + ε)-factor approximation for an instance of the problem for n segments with lengths in [1, λ] in time $n^{o(\lambda/\varepsilon^3)}$.

Huang *et a*l [13] studied a new generalization issue of the TSP known as the Minimum-Cost Bounded Degree Connected Subgraph (MBDCS). The MBDCS issue aims to find a minimum-cost connected subgraph with n=$|V|$ edges from an input graph G= (V, E) with degree upper bounds for certain vertices. The authors demonstrate that, for certain particular circumstances of MBDCS, the goal is similar to finding a minimum-cost Hamiltonian cycle for the input graph, just like the TSP. To properly solve an integer programming framework was proposed for the problem. Following that, an approach was provided for approximating the best solution using the iterative rounding technique to solve the integer programming relaxation.

3.0 Methodology

3.1. TSP Formulation

$$Min \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} * y_{ij} \quad (1)$$

$$\sum_{i=1}^{n} y_{ij} = 1, i = 1, \ldots\ldots n \quad (2)$$

$$\sum_{j=1}^{n} y_{ij} = 1, j = 1, \ldots\ldots n \quad (3)$$

$$y_{ij} \epsilon \{0,1\}, \quad i,j = 1, \ldots\ldots, n \quad (4)$$

$$\sum_{i \epsilon Q j \epsilon Q} y_{ij} \leq |Q| - 1 \text{ for all } Q \subset \{1,2,\ldots\ldots,n\} \text{ and } 2 < |Q| \leq n-1 \quad (5)$$

(From: Bazrafshan *et al* [1])

In this formulation, Equation (1) represent the objective function that minimizes the total distance and C_{ij} is distance or weight of arc (i,j), Equations (2) and (3) are the assignment constraint, which ensures that each node is visited and left exactly once, and Equations (4) and (5) indicate that y_{ij} is a binary variable and equals to 1 if arc (i,j) are in the tours. (5) represents Dantzig - Fulkerson-Johnson (DFT) sub-tour elimination constraints in each subset Q, sub-tours are prevented, ensuring that the number of arcs selected in Q is Smaller than the number of Q nodes. Yij is a binary variable and is equal to 1 when the nodes of i,j are visited [1].

3.2. DE KLERK SDP RELAXATION OF TSP

$$Min \quad \frac{1}{2} trace(DX^{(1)})$$

$$Subject\ to: X^{(k)} \succcurlyeq 0, \qquad k = 1\ldots\ldots, d$$

$$\sum_{k=1}^{d} X^{k} = j - 1,$$

$$I + \sum_{k=1}^{d} cos(\frac{2\pi ik}{n}) X^{(k)} \succcurlyeq 0, \qquad i = 1\ldots\ldots, d$$

$$X^{(k)} \epsilon S^{n} \qquad k = 1\ldots\ldots, d$$

$$\sum_{T \epsilon T_G \{i,j\} \epsilon T} \Pi(X^{(1)}) \geq n$$

Where d=[$\frac{1}{2}n$] is the diameter C_n
(From : de Klerk et al., 2008)

From the above SDP relaxation, a space of k * k symmetric matrices is denoted by S_k and a space of k * k symmetric positve semidefinite matrices by $X \succcurlyeq 0$. $X^{(1)}, \ldots\ldots X^{(d)} \epsilon R^{n*n}$ are matrix variable with the cost of the solution depending on $X^{(1)}$, c represents hamiltonian cycle.For an undirected graph G let A_k(G) be the k-th distance matrix (the matrix i,j-th entry) equal to 1 if and only if the shortest path between vertices i and j in G is of distance K,and equal to 0 otherwise).Let C_n be a cycle of length n (i.e any hamiltonian cycle on [n]).The solution where $A_k(C_n)$ for k=1,…., is a feasible solution for the SDP. $cos(\frac{2\pi ik}{n})$ represents the Eigen value of the matrix A_k while the last constraint of the SDP relaxation represents the subtour elimination constraint.

Algorithm 1: Approximation algorithm for metric TSP

Input: An undirected weighted graph $G\ (V, E, W)$
Output: Hamiltonian path
Obtain solution to the SDP relaxation
Create adjacency matrix A_{ij}
Create degree matrix D_{ij}
$L_{ij} = D_{ij} - A_{ij}$
$n = co - factor\ (L_{ij})$

$x = store(edgeList)$
$T = [\,]$
$While\ i >= N$:
    $T[i] = \sum_{n=1} x[i].W[n]$
    $i = i + 1;$
    $T.append(S[i])$
$M_t = minimum(T)$
$P_c = minimum\_cost\_perfect\_matching(M_t)$
$Multigraph\ (P_c, M_t)$
$Compute\ eulerCycle()$
$Output\ Hamiltonian\ Cycle$

Theorem 1(Gutekunst & Williamson, [3])
Let G be a connected labeled graph and let L be its Laplacian matrix. Then all the cofactors of L are equal and their common value equals the number of spanning trees of G.

Proof: Let $A \epsilon R^{n*n}$ and take E_{ii} to be the matrix with 1 in the $(i,i)^{th}$ the entry and 0s elsewhere. Then

$$det(A + E_{ii}) = det(A) + det(A[i]]) \quad (3.1)$$

This claim is evident if one considers the permutation sum definition of the determinant

$$det(A = (a_{ij})) = \sum_{\pi} sign(\pi) a_{1\pi(1)} a_{2\pi(2)} \quad (3.2)$$

By induction it follows that

$L_{[i]} = [0\ 0\ 0\ 0\ ]$

$$L_{G[i]} = [0] \quad \text{and } (L_{G[i]} = 0) \quad (3.3)$$

$$det(L_G[i]) = det(L_{G-e} + E_{jj}) \quad (3.4)$$

$$det(L_G - e[i]) + det(L_{G-e}[i,j]) \quad (3.5)$$

$$det(L_G - e[i] = det(L_{G-e}[i]) + det(L_{\underline{G}}) = T(G) \quad (3.6)$$

Theorem 2
Given a graph $G\ (V, E)$, a Hamiltonian path is a spanning tree and not every spanning tree in Graph $G(V, E)$ is an Hamiltonian path.

3.3. Creating an Adjacency Matrix of a Graph

The adjacency matrix of a graph $G\ (V, E)$ can be used to depict it. A square matrix known as an adjacency matrix is one in which the number of rows or columns matches the number of graph vertices. An adjacency matrix item can either be 1 or 0, with 1 denoting an adjacent pair of vertices and 0 otherwise. The vertices' names can be added to an empty matrix in a column, a row, and at the top to construct an adjacency matrix. An edge between a pair of vertices is represented by 1 while 0 represent absence of an edge.

Table 3.1: Adjacency Matrix

	A	B	C	D	E
A	0	1	0	1	0
B	1	0	1	0	1
C	0	1	0	1	0
D	1	0	1	0	1
E	0	1	0	1	0

3.4. Creating A Degree Matrix Of A Graph

The degree of every vertex in a graph is represented by a degree matrix, D_{ij}. The adjacency matrix can be used to build the degree matrix of a given graph by replacing all other elements with 0 and the degree of each vertex in the diagonal of the adjacency matrix. The matrix that follows is the adjacency matrix's degree matrix.

$[2\ 0\ 0\ 0\ 0\ 0\ 3\ 0\ 0\ 0\ 0\ 0\ 2\ 0\ 0\ 0\ 0\ 0\ 3\ 0\ 0\ 0\ 0\ 0\ 2\ ]$

3.5. Creating the Laplacian Matrix of a Graph

The Laplacian L_{ij} of a graph can be obtained by subtracting the adjacency matrix from the degree matrix.

$$L_{ij} = D_{ij} - A_{ij} \quad (3.7)$$

Where L_{ij} , $D_{ij,}$ A_{ij} represent Laplacian, degree and adjacency matrix of the graph respectively.

3.6. Total Number of Spanning Trees

To compute the total number of spanning trees, there is need to calculate any of the positive cofactor from the Laplacian matrix.

$$C_{ij} = (-1)^{1+j} |M_{ij}| \quad (3.8)$$

where i j is the index of the matrix.

3.7. Storing the edge list of each spanning tree with their weights in E.

It is necessary to add up the edge weights of each spanning tree, denoted by s, and store the result in List T. The smallest entry in the List T is the minimum spanning tree.

3.8. Minimum Cost Perfect Matching

Blossom algorithm was employed to perform minimum cost perfect matching. The algorithm was implemented on the minimum spanning tree as shown in algorithm 2.

Algorithm 2: Blossom Algorithm (Kolmogorov, [4])

Input: M_t , *initial matching* M on M_t
Output: *minimum cost matching* $M*$ on M_t
function minimum_cost_matching (M_t, M): $M*$
    $P \leftarrow$ compute_augmented_path (M_t, M)
        if P is non-empty then
            return find_minimum_cost_
            matching (M_t, augment M along P)
        else
            return M
        end if
end function

4. Results and Discussion

The algorithm was implemented on symmetric instances of TSP from TSPLIB. The results are as shown in Table4.1

According to Table 4.1 there are six instances in which the algorithm outperformed the existing best-known solution which are Ulysses, U59, KroC100, KroD100, ch130 and bier127 with tour length 6859,28598,20502,21289,6530 and 118258 respectively. The approximate tour constructed by the algorithm in ulyssis16 is as illustrated below in Figure 4.1. The starting city which is 16 is indicated by a thick border.

5. Conclusion

In this paper, an approximation algorithm for metric TSP is presented. The algorithm operates by taking a vertex as input and constructing minimum spanning tree of a graph representing a TSP. The algorithm was tested on TSP instances from TSPLIB and compared with previous best solutions. The experimental results showed that the proposed algorithm could always converge to a solution even for difficult and large problems which makes it a promising approach for future development in the combinatorial optimization field. However, the proposed algorithm was only tested on Symmetric instances of metric TSP.

A practical application of the approximation algorithm for TSP described in this study is in transportation and logistics to create efficient routes for delivery, cutting fuel costs, lowering expenses, and improving delivery times. Another application is in circuit board manufacturing, where engineers design effective connections. Approximation algorithms help by shortening wiring paths and avoiding overlaps, leading to reduced costs and increased efficiency in materials and production time.

Table 4.1: Simulation results on various instances compared to the best-known solution

Instances	Cities	Solution	Time (MS)	Best known solution	Time (MS)
Ulysses16	16	6859	17	6859	18
Berlin52	52	7620	41	7542	35
kroB150	150	26300	26111	26130	193757
U59	59	28598	29357	29368	192395
KroA200	200	893425	886330	886321	192395
KroC100	100	20502	655	20749	695
KroD100	100	21289	10601	21294	10603
KroE100	100	22077	86	22068	84
ch130	130	6530	25742	6610	25887
bier 127	127	118258	38029	118282	38035

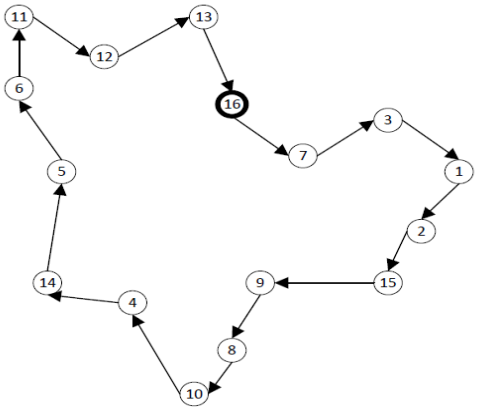

Figure 4.1: Approximate tour of Ulyssis16 city proble

References

[1] Bazrafshan, R., Hashemkhani Zolfani, S., & Mirzapour Al-e-hashem, S. M. J. (2021). Comparison of the sub-tour elimination methods for the asymmetric traveling salesman problem applying the SECA method. *Axioms*, *10*(1), 19. Etienne de Klerk, Dmitrii V Pasechnik, and Renata Sotirov. On semidefinite programming relaxations of the traveling salesman problem. SIAM Journal on Optimization, 19(4):1559–1573, 2008.

[2] Evnin, A. Y., & Yusova, N. I. (2017). An approximation algorithm for the maximum traveling salesman problem. *Journal of Computational and Engineering Mathematics*, *4*(3), 49-54.

[3] Gutekunst, S. C. (2020). *Fantastic Relaxations of the TSP and How to Bound Them: Relaxations of the Traveling Salesman Problem and Their Integrality Gaps*. Cornell University.

[4] Kolmogorov, V. (2009). Blossom V: a new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation*, *1*, 43-67.

[5] Pokharel, M. (2020). Computational Complexity Theory (P, NP, NP-Complete and NP-Hard Problems). *Tribhuvan University, Nepal*.

[6] Sebo, A., & Van Zuylen, A. (2016). The salesman's improved paths: A 3/2+ 1/34 approximation. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)* (pp. 118-127). IEEE.

[7] Svensson, O., Tarnawski, J., & Végh, L. A. (2018). Constant factor approximation for ATSP with two edge weights. *Mathematical Programming*, *172*(1), 371-397.

[8] Svensson, O., Tarnawski, J., & Végh, L. A. (2020). A constant-factor approximation algorithm for the asymmetric traveling salesman problem. *Journal of the ACM (JACM)*, *67*(6), 1-53.

[9] Traub, V. (2020). *Approximation algorithms for traveling salesman problems* (Doctoral dissertation, Universitäts-und Landesbibliothek Bonn).

[10] Traub, V., & Vygen, J. (2020). Beating the integrality ratio for s-t-tours in graphs. *SIAM Journal on Computing*, *52*(6), FOCS18-37.

[11] Zenklusen, R. (2019). A 1.5-approximation for path TSP. In *Proceedings of the thirtieth annual ACM-SIAM symposium on discrete algorithms* (pp. 1539-1549). Society for Industrial and Applied Mathematics.

[12] Ghaseminia, B., & Salavatipour, M. R. (2025). A PTAS for Travelling Salesman Problem with Neighbourhoods Over Parallel Line Segments of Similar Length. arXiv preprint arXiv:2504.02190.

[13] Huang, Z., Liao, X., Naik, P. A., & Lu, X. (2024). On approximating a new generalization of traveling salesman problem. *Heliyon*, *10*(10)