Open Access Library Journal
2020, Volume 7, e6159
ISSN Online: 2333-9721
ISSN Print: 2333-9705

An Exact Ranked Linear Assignments Solution to the Symmetric Traveling Salesman Problem

Roy Danchick

Independent Researcher, 2051 S. Bentley Ave., Apt. 301, Los Angeles, CA, USA
Email: dodeee@sbcglobal.net

How to cite this paper: Danchick, R. (2020) An Exact Ranked Linear Assignments Solution to the Symmetric Traveling Salesman Problem. *Open Access Library Journal*, **7**: e6159.
https://doi.org/10.4236/oalib.1106159

Received: February 12, 2020
Accepted: March 8, 2020
Published: March 11, 2020

Abstract

In this paper, we show how Murty's ranked linear assignment algorithm can be applied to exactly solve the symmetric Traveling Salesman Problem (TSP). To increase the Murty algorithm's computational efficiency in solving the TSP, we develop a simple algorithm that determines whether a node that is generated in Murty's sequential node partitioning process contains a sub-cycle of length less than n, where n is the number of cities to be visited. Each such node cannot generate a genuine solution, which must be a full n-cycle, and can thus be eliminated from further partitioning. In exactly, solving the TSP Murty's ranking process continues, discarding all such nodes, terminating in a finite number of rankings when the first such ranked solution is encountered that is a full n-cycle. This first ranked n-cycle is the exact solution to the given TSP problem.

Subject Areas

Applied Statistical Mathematics, Mathematical Analysis, Numerical Mathematics

Keywords

Traveling Salesman, Applied Probability, Assignment

1. Introduction

The NP-hard Traveling Salesman Problem (TSP) is a paradigm of computational complexity. It continues to be one of the most intensively investigated problems in Operations Research. The famed Irish mathematician William Rowan Hamilton first formulated the TSP problem in the 1800's. Merrill Flood, the American mathematician, popularized the TSP as a worthy object for investigation in the 1940's [1]. The Rand Corporation, in the 1950's was a focal point for research on the TSP. At Rand Richard Bellman, George Dantzig, Ray Fulkerson and Selmer Johnson made fundamental contributions to the solution of the TSP. Bellman in [2] proposed a dynamic programming approach to solving the TSP that was limited to a small number of cities. Bellmans's algorithm was rediscovered and improved by Held and Karp in [3]. Their approach provides both sharp bounds for the solution as well as the computational complexity for attaining an approximate solution and was able to find an exact solution for the TSP problem of 64 cities.

Bellman's and the Held-Karp's algorithms are based on a branch and bound approach. The branch and bound approach is based on the principle that the total set of feasible solutions can be partitioned into smaller subsets of solutions. Branch and bound-based TSP algorithms are not exact. Mathematicians and Computer Scientists have since worked to improve their computational efficiency, even at the margins, of the best available of these algorithms. Gutin and Punnin summarized the state of the art of TSP algorithms as of 20,002 in [4]. The record for the solved-for number of cities is 85,900 at Bell laboratories in 2006.

Branch-and-cut is a second general technique to solve the symmetric TSP problem. It requires an exponential number of cut-set elimination constraints. David P. Williams has compiled a very accessible overview of progress on the TSP in [5] which includes a section on branch and cut methods in conjunction with linear programming algorithms. To complete the picture, there is an extensive literature on heuristic TSP solvers [6].

In this paper, we adapt Murty's ranked linear assignment algorithm [7] to the symmetric TSP application. The Murty algorithm solves the ranked linear assignment problem in a sequence of stages. In the first stage, it uses, say, the Jonker-Volgenant (JV) [8] fast 0 - 1 integer linear assignment algorithm to find the exact minimizing solution on the original assignment problem matrix. The first stage solution then generates a set of n nodes from the initial solution by partitioning this first stage solution. Each such node specifies which matrix elements in the first stage solution are to be included in the next solution and which are to be excluded. For each node, the included elements specify which rows and columns are to be struck out of the original first stage matrix to provide the assignment matrices in the next stage while the excluded elements specify which elements in the original matrix are to be replaced by machine infinity. This produces a set of node-correspondent square matrices of decreasing order. Each such node-specific matrix is subjected to the JV algorithm thus producing an assignment value; the smallest of these values is then the second ranked value. The method proceeds recursively from one stage to the next to generate the third ranked solution, the fourth, etc. Our TSP-specific implementation continues the ranking until the first full n-cycle is found, thus making our proposed algorithm exact.

To further adapt the Murty ranked linear assignment algorithm to the TSP for computational efficiency, we construct a simple algorithm that determines whether a node contains a sub-cycle of length < n. Such nodes cannot generate a full cycle. We will show that most nodes contain sub-cycles of length $l, 2 \le l < n$. We can thus expect that the Murty algorithm, equipped with this node sub-cycle identification and elimination algorithm along with a core fast JV 0 - 1 integer linear assignment problem solution algorithm, which is $O(r^3)$ in complexity, $2 \le r \le n$, will compare well in speed with the currently best available symmetric TSP algorithms but in contrast with them always provide exact answers.

We assume the assignment matrix of distances between the cities to be visited is symmetric and positive except for zeros along the main diagonal. We use the device of replacement of each diagonal zero entry with a positive number that is greater than any other matrix entry. This trick guarantees that no ranked solution contains a 1-cycle. Importantly we show that eliminating all such permutations containing a 1-cycle asymptotically shrinks the number of possible permutations that must be processed by Murty's ranked linear assignment algorithm by a factor of ≈0.6321. The sub-cycle node elimination algorithm removes the remaining $0.3679(1-1/n)n!$ possible permutations as $n \to \infty$. The space of solutions is thus reduced to the set of $(n-1)!$ full n-cycles.

Section 2 Contains our definitions and notation.

Section 3. Provides our key proofs and a computational example

Section 4. Supplies our conclusions

Section 5. Sketches a map for future research.

Section 6. Lists references.

2. Definitions and Notation

We closely follow the definitions and notation found in [7] in which:

n = the number of cities along the traveling salesman's route.

$C = (c_{ij})_{i,j=1,2,\cdots,n}$ is the assignment matrix of inter-city distances such that $c_{ij} = c_{ji} > 0, i \ne j, c_{ii} = d > c_{ij}, i, j \le 1,2,\cdots,n$.

Paraphrasing Murty in [7] and specializing the general ranked assignment problem down to the TSP we can define it as the 0 - 1 integer linear programming problem with an additional full n-cycle constraint (4):

Minimize $Z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$

Subject to: 1) $\sum_{j=1}^{n} x_{ij} = 1, i = 1,2,\cdots,n$ (1)

2) $\sum_{i=1}^{n} x_{ij} = 1, j = 1,2,\cdots,n$

3) $x_{ij} = 1$ or $x_{ij} = 0,\ i, j = 1,2,\cdots,n$

4) If $(1, j_1),(2, j_2),\cdots,(n, j_n)$ are the indices of the variables x_{ij} taking on the value 1 then the corresponding group element that is contained in the symmetric group of permutations on n letters, S_n, is a full cycle of length n.

A node, M, that is generated on the k^{th} stage of the ranking process is given by:

$$M = \left\{(i_1,j_1),(i_2,j_2),\cdots,(i_r,j_r);\overline{(m_1,p_1)},\overline{(m_2,p_2)},\cdots,\overline{(m_{n-r},p_{n-r})}\right\},$$

$(i_1,j_1),(i_2,j_2),\cdots,(i_r,j_r)$ specify which original matrix entries are to retained in the solution and $\overline{(m_1,p_1)},\overline{(m_2,p_2)},\cdots,\overline{(m_{n-r},p_{n-r})}$ specify which original matrix entries are to be excluded from the solution by inserting machine ∞ into the entries $(m_i,p_i), i = 1,2,\cdots,n-r, r < n$.

3. Derivations

3.1. Node Rejection and Solution Identification

The basic idea is to sequentially form the node-correspondent product of 2-cycles, $(i_1,j_1),(i_2,j_2),\cdots,(i_r,j_r) \in S_r$, into a product of disjoint cycles. A necessary condition for the node to generate a genuine solution to the TSP is that this product must not contain a single cycle of length ≤ *r*. We have constructed a simple algorithm to determine whether this necessary condition is met. It requires $O(r^2)$ integer comparisons. The MATLAB code implementation of this algorithm is shown below. The input variables are defined as the integer vectors:

$$I = [i_1,i_2,\cdots,i_r]$$
$$J = [j_1,j_2,\cdots,j_r]$$

```
function test = node_reject(I,J,r)
k = 1;
test = 0;
L_k = 1;
while k <= r & test == 0 & L_k <= r
   i = I(k);
   j = J(k);
   i_next = k;
      l = 1;
      while l <= r & i > 0 & test == 0
         i_next = next_i(i_next,I,J,r);
         if i_next ~= 0
            j_i = J(i_next);
            L_k = L_k + 1;
            if j_i == i & L_k < r
               test = 1;
            end
         end
          l = l + 1;
     end
      k = k + 1;
 end
function i_next = next_i(l,I,J,r)
i_next = 0;
j = J(l);
   k = 1;
   while k <= r & i_next == 0;
         i = I(k);
         if i == j;
            i_next = k;
         end
         k = k + 1;
   end
```

The output variable is: $\text{test} = \begin{cases} 0 \text{ if the node does not contain a sub-cycle} \\ 1 \text{ if the node contains a sub-cycle} \end{cases}.$

An important consequence of sub-cycle identification and rejection is that the node partitioning process for a given ranked assignment solution stops whenever a sub-cycle is encountered. The same algorithm can be exploited to determine whether a node-partitioning solution is a genuine n-cycle and is a solution candidate, depending on whether its Z value is smallest among those from the set of eligible nodes—those whose two-cycle products do not contain sub-cycles in their representation as products of disjoint sub-cycles.

3.2. Diagonal ∞'s and Node Elimination Algorithm Probabilities and Statistics

The efficiency advantage that the Murty ranked assignment approach confers to the TSP problem derives from two essential features: 1) The large fraction of the $n!$ possible permutation solutions that are eliminated by the machine ∞'s main diagonal entries and 2) the elimination of candidate nodes by the node rejection algorithm.

Table 1 below shows the number of permutations that keep at least 1 of the n letters fixed and are thus eliminated by the node identification and rejection algorithm. Table 2 shows the result of a set of Monte Carlo sampling experiments with a sample number of 1000 per each case of the number of cities.

Examination of Table 1 motivates the key *ansatz*:

The number of permutations, R_n, *eliminated because of diagonal ∞'s is given by the recursion*:

Table 1. Diagonal ∞ permutation elimination.

Number of cities	Number of permutations eliminated by diagonal ∞'s	Probability of diagonal entry elimination
2	1	0.5
3	4	0.66667
4	15	0.625
5	76	0.63333
6	455	0.61394

Table 2. Node rejection fractions.

Number of cities	Fraction of nodes eliminated by machine ∞ diagonals	3σ uncertainty in fraction of nodes eliminated by diagonal ∞'s	Fraction of nodes eliminated by node rejection algorithm	3σ uncertainty in fraction of nodes eliminate by rejection algorithm
10	0.6130	0.0462	0.2780	0.0425
20	0.6350	0.0457	0.3170	0.3170
50	0.6470	0.0453	0.3370	0.0448
100	0.6400	0.0455	0.3470	0.0452
200	0.6160	0.0461	0.3750	0.0459
500	0.6170	0.0461	0.3820	0.0461

$$R_n = \begin{cases} nR_{n-1} - 1 \text{ if } n \text{ is even} \\ nR_{n-1} + 1 \text{ if } n \text{ is odd} \end{cases}, \; n = 3,4,5,\cdots \tag{2}$$

If (2) holds for all $n \geq 3$ then the corresponding diagonal ∞'s permutation rejection probability recursion is given by:

$$p_n = \begin{cases} p_{n-1} - 1/n! \text{ if } n \text{ is even} \\ p_{n-1} + 1/n! \text{ if } n \text{ is odd} \end{cases}, \; n = 3,4,5,\cdots \tag{3}$$

If the recursion (2) above holds for all $n \geq 3$ we can expand recursion (3) to obtain

$$p_{n+1} = p_2 + \sum_{k=3}^{n} (-1)^{i-1} / k! \tag{4}$$

From (4) it is easy to see that

$$\lim_{n \to \infty} p_n = 1 - \exp(-1) \approx 0.6321 \tag{5}$$

which is agreement with what **Table 1** and **Table 2** show.

We will use the four California cities: 1) Los Angeles, 2) San Diego, 3) San Jose, and 4) San Francisco, to pose a "toy problem" that nevertheless captures the essential features and potential effectiveness of our approach. The corresponding assignment matrix, with entries rounded to the nearest mile, is:

$$C = \begin{bmatrix} \infty & 120 & 340 & 382 \\ 120 & \infty & 466 & 508 \\ 340 & 466 & \infty & 48 \\ 382 & 508 & 48 & \infty \end{bmatrix}$$

By inspection the optimal stage 1 solution is:

$$(1,2),(2,1),(3,4),(4,3), Z = 336.$$

Note that this stage 1 solution is not a solution of the TSP problem because it contains two sub-cycles of length 2.

The stage 2 nodes that are partitioned by the optimal solution of stage 1 are:

$$M_1 = \overline{(1,2)},$$
$$M_2 = (1,2),\overline{(2,1)},$$
$$M_3 = (1,2),(2,1),\overline{(3,4)},$$
$$M_4 = (1,2),(2,1),(3,4),\overline{(4,3)}.$$

The node rejection test tells us that only M_1 and M_2 need be processed.

The optimal stage 2 solution on M_1 is the 4-cycle:
$(1,4),(4,3),(3,2),(2,1), Z = 966$

The optimal stage 2 solution on M_2 is the 4-cycle
$(1,2),(2,3),(3,4),(4,1), Z = 1010$.

Thus the solution to the given TSP problem is the solution is on M_1.

4. Conclusions

We have proposed an exact algorithm for the solution the symmetric traveling salesman problem that exploits Murty's ranked linear assignment algorithm. We have shown how the Murty algorithm can be equipped with a simple test that eliminates from further processing any node that cannot generate a full n-cycle. We have proved that if the recursion (2) holds that the device of inserting machine ∞'s down the main diagonal asymptotically reduces the number of candidate permutations by a factor of $1-\exp(-1) \approx 0.6321$. The remaining $0.3679(1-1/n)n!$ permutations, each of which corresponds to a sub-cycles of length < n, are eliminated by the node rejection test. Exploiting the fast JV algorithm as the core 0 - 1 integer linear assignment problem solution method gives the proposed algorithm the potential for great efficiency in exactly solving large problems with $n \gg 1$.

5. Directions for Future Research

Our first objective would be to prove the conjecture that the recursion (2) holds for all $n \geq 3$, say, by mathematical induction.

Second would be the derivation of the computational complexity of our proposed algorithm.

Finally, we would either acquire reliable Murty and JV codes or program them from scratch, equip the Murty code with the node rejection algorithm and run the Murty + JV + node rejection combination on sample problems in the literature. We would then compare run times for our proposed algorithm with those of the best current symmetric TSP codes.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Flood, M. (1956) The Traveling-Salesman Problem. *Operations Research*, **4**, 61-75. https://doi.org/10.1287/opre.4.1.61

[2] Bellman, R. (1962) Dynamic Programming Treatment of the Traveling Salesman Problem. *Journal of the ACM*, **9**, 61-63. https://doi.org/10.1145/321105.321111

[3] Held, M. and Karp, R. (1962) A Dynamic Programming Approach to Sequencing. *Journal for the Society for Industrial and Applied Mathematics,* **10**, 1-10. https://doi.org/10.1137/0110015

[4] Gutin, G. and Punnen, E. (2002) The Traveling Salesman Problem and Its Variation. Vol. 12, Springer, New York.

[5] Williamson, D. (2014) The Traveling Salesman Problem: An Overview. Lecture Notes, Cornell University Ebay Research, 21 January.

[6] Rego, C., Gamboa, D., Glover, F. and Osterman, C. (2011) Traveling Salesman Problem Heuristics: Leading Methods, Implementations and Latest Advances. *European Journal of Operational Research*, **211**, 427-441. https://doi.org/10.1016/j.ejor.2010.09.010

[7] Murty, K. (1968) An Algorithm for Ranking All the Assignments in Order of Increasing Cost. *Operations Research*, **16**, 682-687.

https://doi.org/10.1287/opre.16.3.682

[8] Jonker, R. and Volgenant, A. (1987) A Shortest Path Algorithm for Dense and Sparse Linear Assignment Problems. *Computing*, **38**, 325-340. https://doi.org/10.1007/BF02278710