Performance Evaluation of Convex Hull Node-Based Heuristics for Solving the Travelling Salesman Problem

Emmanuel O. Asani, Aderemi E. Okeyinka, and Ayodele Ariyo Adebiyi

Abstract This experimental study investigated the effect of Convex Hull on Node-based Heuristics. This was motivated by the assertion in the literature that starting some insertion tours with a convex hull theoretically degrades their worst case from twice optimal to thrice optimal. The Node-based techniques considered were Nearest Neighbour Heuristic (NNH) and Nearest Insertion Heuristic (NIH). The derived heuristics with Convex Hull were referred to in this study as Convex Hull Nearest Neighbour (CHNN) and Convex Hull Nearest Insertion (CHNI), respectively. The techniques were experimented on eleven benchmark instances from TSPLIB using Python Programming Language. Experimental results showed that the performances of both the Nearest Neighbour and Nearest Insertion were enhanced in terms of Computational speed and solution quality.

Keywords Nearest Neighbour Heuristic · Nearest Insertion Heuristic · Convex Hull Nearest Neighbour · Convex Hull Nearest Insertion (CHNI) · Node-based heuristics

1 Introduction

As is typical with Combinatorial Optimization Problems (COP), the Travelling Salesman Problem (TSP) can either be approached using exact solutions or heuristics. Although exact techniques will always guarantee optimal solutions, they are however unsuitable for NP-hard problems with large solution space [1]. For instance, the solution renowned as the best performing exact technique is based on dynamic programming and has a complexity of $O\left(2^n n^2\right)$, thus making it impracticable to solve TSP as the search space expands [2]. Heuristics on the other hand provides

E. O. Asani (✉) · A. E. Okeyinka · A. A. Adebiyi
Department of Computer Science, Landmark University, Omu-Aran, Nigeria
e-mail: asani.emmanuel@lmu.edu.ng

E. O. Asani
Landmark University SDG 11, Landmark University, Omu-Aran, Nigeria

X.-S. Yang et al. (eds.), *Proceedings of Sixth International Congress on Information and Communication Technology*, Lecture Notes in Networks and Systems 217,
https://doi.org/10.1007/978-981-16-2102-4_60

solutions within polynomial p time. Heuristics provide approximate solutions within the constraint of polynomial time. Heuristic solutions are referred to as approximate because they are less than optimal but good enough within the constraint of time. Good heuristics are often simple to implement and flexible to accommodate complex constraints.

Heuristics have been classified by researchers based on divergent criteria in previous studies. For instance, authors [3–5] classified heuristics into three based on the atomicity of solution procedure namely, Tour Construction, Improvement / Local Search Heuristics, and Compound Heuristics. The Tour Construction heuristics are stand-alone techniques that build solutions, step by step by following a set of predefined procedures. These procedures describe the processes involved in stages of Initialization, Selection, and Insertion. The construction heuristic techniques have been used extensively in solving classic combinatorial optimization problems. Common techniques include the Nearest Neighbour Heuristic (NNH), the Nearest Insertion (NIH), Cheapest Insertion, Random Insertion, Addition heuristics, Savings Heuristics, and so on. The Improvement/ Local Search Methods combine two or more techniques and iteratively improve solutions until a better solution is not feasible. Compound Heuristics are a hybrid of both the tour construction and local search methods. In this approach, two or more tour heuristics are applied independently, while a selection routine is deployed to determine the best of the solution. Additionally, authors [6, 7] classified heuristics based on their solution paradigm into space-partitioning-based heuristics, edge-based heuristics, and node-based heuristics. The space-partitioning-based heuristics build solutions by first splitting the nodes into subsets $(S_1, S_2, \ldots S_n)$ based on their paired distances, the nodes within the same subset are then connected into a Hamilton path, after which the Hamilton circuit for S is obtained by coupling the Hamiltonian paths of subsets $(S_1, S_2, \ldots S_n)$. Examples of heuristics under this category are Strip and Hilbert. Edge-based heuristics build solutions by first determining the edge with the smallest distance and then placing it into the circuit. Most heuristics under the edge-based category are all built on the Minimum Spanning Tree (MST); they include multiple fragment heuristic, double-MST (DMST), and the Christofides algorithm (Chris). In the third category, the node-based heuristics build the tour by expanding the nodes one at a time till all the nodes have been inserted. Node-based heuristics must first decide which node to be used as the initial node, then determine the succeeding node to explore in each iteration, and where it will be inserted. Examples of node-based heuristics include the nearest-neighbour heuristics, the insertion heuristics, the convex hull-based insertion heuristics, the addition heuristics, and the augmented addition heuristics. Apparently, node-based heuristics are chiefly Tour Construction techniques as well.

In this study, two node-based heuristics namely Nearest Neighbour and Nearest Insertion are studied in relation to Convex Hull. The Convex Hull P of a set of points (S) is the Euclidean plane which is the minimum convex polygon that encompasses all the points in S. The Convex Hull is generally used to build initial points of an approximation technique. Thus, against the findings by [8] that the worst-case performances of some insertion heuristics are theoretically degraded from twice optimal to

thrice optimal when started with a convex hull, there is a need to evaluate the experimental performance of Convex Hull Nearest Neighbour and Convex Hull Nearest Insertion vis-à-vis the classic Nearest Neighbour and Nearest Insertion Heuristics in order to determine the experimental effect of Convex Hull on node-based heuristics. These techniques are experimented on some benchmark instances from TSPLIB using the Python Programming Language as the implementation tool.

2 The Problem Formulation

First formulated in the nineteenth century and enhanced in the 1930s by M. M. Flood, the Travelling Salesman Problem has become the benchmark for several other techniques of optimization [9]. The TSP is the shortest tour (or path) problem to find the optimal route while visiting a set of cities (or nodes), ensuring each city (or node) is visited exactly once and regarding the Hamiltonian circuit, return to the start node or city [10]. The Travelling Salesman must traverse cities 1 *to n* in a Hamiltonian cycle that is, Start from city 1 and traverse the remaining n−1 cities in arbitrary order, and return to the starting point with the objective of touching the cities once at a minimal cost. The distance *d(i, j)* depicts the distance from city *i to j* Thus, TSP is formally defined as follows:

$$F = \min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}$$

$$\sum_{j=1}^{n} x_{ij} = 1;\ j = 1, \ldots, n$$

$$\sum_{i=1}^{n} x_{ij} = 1;\ j = 1, \ldots, n$$

The objective function is marked with F. With a limitation,
$x_{i_1 i_2} + x_{i_2 i_3} + \cdots + x_{i_r i_1} \leq r - 1$.
x_{ij} x_{ij} are the binary variables

$$x_{ij} = \begin{cases} 1 \text{ if the salesman travels from city } i \text{ to city} \\ 0 \text{ if the salesman is not travelling from city } i \text{ to city } j \end{cases}$$

$d_{ij} d_{ij}$ is the distance from city i to city j.

The TSP has applications in several areas, most especially in varying areas of transportation. Despite being an NP-hard problem, which is easily understood but computationally difficult to solve, TSP has several solution algorithms broadly categorized into Exact Algorithms and Approximate Algorithms (heuristics).

3 The Proposed Technique

This study experiments the performance of two node-based heuristics in relation to their convex hull counterpart. Node-based heuristics build circuits on a node by node basis as follows [7]:

Input: Q: a TSP query of a set of points
Output: T: the TSP for Q
1. begin
2. T ← **init(Q);**
3. while T does not contain all nodes in Q do
4. v ← select(Q, T);
5. insert(v, T);
6. return T;

The Nearest Neighbour heuristic is a classic node-based tour construction heuristic for solving the Travelling Salesman Problem. The Nearest Neighbour Heuristic tries to solve the Travelling Salesman Problem using a greedy approach [10]. The Nearest Neighbour starts with a city/node and builds the remaining tour by joining the node closest to the starting node to the tour. This process is iterated for all the nodes that are not yet part of the tour until the tour is fully build and a Hamiltonian circuit is formed. This process is greedy in nature, thus the performance is relatively low. The solution quality of the Nearest Neighbour Heuristic is evaluated as follows:

$$f_s / f_{OPT} = \frac{1}{2}\left[\log(n)\right] + \frac{1}{2}$$

while the worst-case complexity is $T(n) = O\left(n^2\right)$.

Where f_s is the length of a tour by the solution and f_{OPT} is the optimal tour length of the NNH. Generally, the Nearest Neighbour Heuristic can solve the TSP in good time, with less than optimal solution quality. Thus, recent literature focuses on using the Nearest Neighbour either as part of a hybrid method as in [1, 7, 11, 12] or as a seed technique in a metaheuristic for building initial solutions.

The Nearest Insertion Heuristic, on the other hand, chooses the next node $x^* = \arg\min_{v \notin T_i}\{\forall x_i \in T_i\}$. The following pseudocode depicts the workings of the Nearest Insertion Techniques.

1. Start the tour from an arbitrary node i
2. Add a node $j$nearest to $i$to form a partial circuit $T = i - j - i$
3. Find a node $k$not in the partial circuit for which the distance to any of the nodes in the subtour is shortest, $d(k, T) = \min_{i \notin T} d(i, T)$
4. Find an edge $[i, j]$of the partial circuit to insert k, such that $\Delta f = c_{ik} + c_{kj} - c_{ij}$is minimial and insert k.
5. Iterate step 3 until a Hamiltonian cycle is formed

The solution quality of Nearest Insertion is evaluated as follows:
$f_s / f_{OPT} \leq 2$ while the worst-case complexity is $T(n) = O(n^2)$
Where f_s is the length of a tour by the solution and f_{OPT} is the optimal tour length of the NIH.

The Convex Hull P of a set of points (S) is the Euclidean plane which is the minimum convex polygon that encompasses all the points in S. It is generally used to build initial points of an approximation technique [13].

Given a set of specified vertices, the Convex Hull Nearest Neighbour (CHNN) method begins by constructing the convex hull subtour associated with the nodes, start the tour from a point in the edge, then inserts the nearest neighbour of subsequent nodes not already on the tour. Equally given the Convex Hull subtour, the Convex Hull Nearest Insertion Technique is depicted in the pseudocode below:

1. Begin with the Convex Hull subtour i
2. Add a node $j$nearest to $i$to form a partial circuit $T = i - j - i$
3. Find a node $j$not in the partial circuit for which the distance to any of the nodes in the subtour is shortest, $d(k, T) = \min_{i \notin T} d(i, T)$
4. Find an edge of the partial circuit to insert j, such that $\Delta f$is minimal and insert k($\Delta f$is the increment in the subtour length).
5. Iterate step 3 until a Hamiltonian cycle is formed

4 Computational Results

The NNH, NIH, CHNN, and CHNI were implemented on eleven (11) benchmark instances from TSPLIB made available by Heidelberg University on http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/. Table 1 shows the performances of the methods (NNH, NIH, CHNN, and CHNI) on eleven instances, in terms of solution

Table 1 Performances of NNH, NIH, CHNN, and CHNI on 11 instances

Instances	Optimal tour length	Avg. Tour length				Computational time $T_{Average}(S)$			
		NNH	NIH	CHNN	CHNI	NNH	NIH	CHNN	CHNI
pr76*	108,159	154,936.8	113,958.8	118,774.5	113,676.7	0.0185	0.0178	0.0135	0.0156
ch130*	6116	7537.2	6539.4	6674.2	6138.7	0.0427	0.0401	0.028	0.0335
pr144	58,537	64,926.5	63,039.8	60,273.1	60,184.8	0.0461	0.0433	0.0277	0.0351
ch150	6528	7478.4	7183.0	7144.4	7005.3	0.043	0.0400	0.0230	0.0298
pr299*	48,189	60,369.9	53,691.1	53,969.2	52,918.3	0.1732	0.1613	0.0941	0.1259
pr439*	107,217	134,389.0	121,819.2	131,652.5	118,324.3	0.4173	0.3920	0.2531	0.3218
pcb442	50,778	63,441.9	58,589.3	59,040.3	57,313.9	0.3647	0.3387	0.1983	0.2665
d493*	35,002	43,471.5	38,869.2	36,105.6	36,507.5	0.4688	0.435	0.2508	0.3416
d657*	48,912	63,845.3	55,121.4	54,420.1	53,503.8	0.8425	0.7824	0.4454	0.6109
d1291	50,801	62,121.2	61,647.9	61,538.1	59,582.9	3.4465	3.2159	1.8268	2.5385
pr2392	378,032	475,322.6	436,523.0	482,834.8	422,642.7	11.999	11.167	6.3006	8.7052

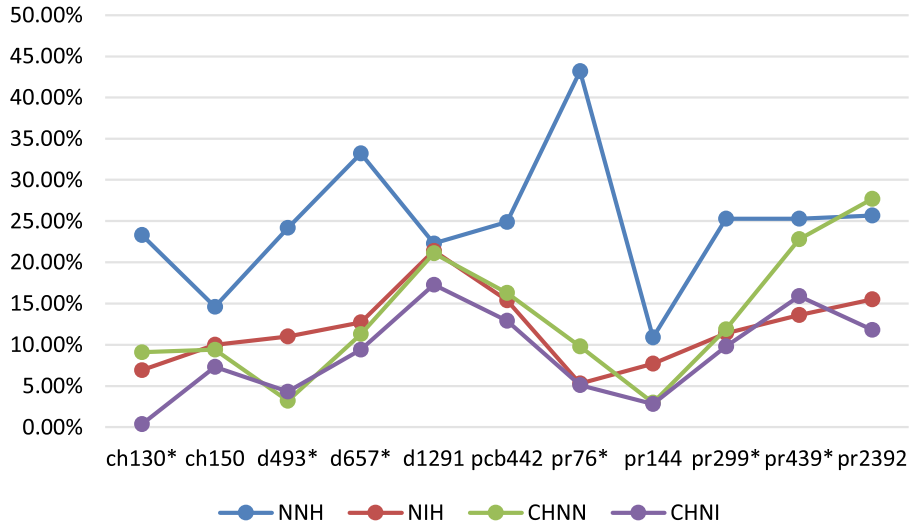

Fig. 1 Percentage deviation of (NN, NI, CHNN, and CHNI) solutions from optimal tour length

quality and computational speed. The percentage deviation of the methods from the optimal solution was also depicted graphically in Fig. 1.

CHNN outperformed NNH in terms of quality in all instances except for *pr2392*. The speed of computation was also reduced for all instances. CHNN did better in terms of speed with relation to NIH also in all instances although NIH performed more competitively with CHNN than NNH in terms of solution quality. The performance of NIH in relation to that of NNH is consistent with findings in literature [7, 11] that the Nearest Insertion Heuristic performs better than the Nearest Neighbour Heuristic, in terms of closeness to optimality. Consequently, CHNI outperformed CHNN, Nearest Neighbour, and Nearest Insertion. This is consistent with findings by [8] that although Convex Hull tends to theoretically degrade the worst case of insertion heuristics, the experimental performance of insertion heuristics is in fact enhanced. Our findings show that Convex Hull tends to relatively improve performances of node-based insertion. Figure 1 shows the percentage deviation of NNH, NIH, CHNN, and CHNI from the optimal solution.

The Nearest Neighbour Heuristic deviated the most. This is perhaps due to its greedy technique which results in *curse of dimensionality*. The chart also shows that the Convex Hull has the potential of enhancing solution quality.

5 Conclusion

In this study, we have investigated the experimental effect of Convex Hull on two node-based heuristics namely, Nearest Neighbour and Nearest Insertion. This was against the background that starting some insertion tours with convex hull theoretically degrades their worst case from twice optimal to thrice optimal. The derived

heuristics Convex Hull Nearest Neighbour (CHNN) and Convex Hull Nearest Insertion (CHNI) as well as Nearest Neighbour and Nearest Insertion were experimented on eleven benchmark instances from TSPLIB using Python Programming Language. Experimental results showed that the performance of both the Nearest Neighbour and Nearest Insertion Heuristics were improved in terms of Computational speed and solution quality. Future work may involve the inclusion of more node-based heuristics such as Addition heuristics to arrive at a broader conclusion.

Acknowledgements We gratefully acknowledge the support of the Landmark University Center for Research Innovation and Development (LUCRID) for access to research repositories, literary materials, useful insights from affiliate researchers, and funding.

References

1. Asani EO, Okeyinka AE, Adebiyi AA (2020) A construction tour technique for solving the travelling salesman problem based on convex hull and nearest neighbour heuristics. Proceeding of IEEE international conference in mathematics, computer engineering and computer science (ICMCECS), (2020), pp 1–4. https://doi.org/10.1109/ICMCECS47690.2020.240847.
2. Deudon M, Cournut P, Lacoste A, Adulyasak Y, Rousseau LM (2018) Learning heuristics for the TSP by policy gradient. In: van Hoeve WJ (eds) Integration of constraint programming, artificial intelligence, and operations research. CPAIOR (2018), Lecture Notes in Computer Science, vol 10848. Springer, Cham
3. Oliveira JF, Carravilla MA (2009) Heuristics and local search, Available online: https://paginas.fe.up.pt/~mac/ensino/docs/OR/CombinatorialOptimizationHeuristicsLocalSearch.pdf. Accessed 13 Feb 2019
4. Marti R, Reinelt G (2011) Heuristic Methods. The linear ordering problem exact and heuristic methods in combinatorial optimisation, Springer, Berlin, Heidelberg. ISBN: 978–3–64–16728–7, pp 17–40. https://doi.org/10.1007/978-3-642-16729-42
5. Kyritsis M, Gulliver SR, Feredoes E, Ud Din S (2018) Human behaviour in the Euclidean travelling salesperson problem: computational modelling of heuristics and figural effects. Cognitive Syst Res 52:387–399
6. Huang W, Yu JX, Shang Z (2016) A sketch-first approach for finding TSP. In: Cheema M, Zhang W, Chang L (eds) Databases theory and applications. ADC 2016. Lecture Notes in Computer Science, vol 9877. Springer, Cham. https://doi.org/10.1007/978-3-319-46922-5_10
7. Huang W, Yu JX (2017) Investigating TSP Heuristics for location-based services. Data Sci Eng 2:71–93. https://doi.org/10.1007/s41019-016-0030-0
8. Warburton AR (1993) Worst-case analysis of some convex hull heuristics for the Euclidean travelling salesman problem. Oper Res Lett 13:37–42
9. Ajaz AK, Himani A (2016) Determining the shortest Path for travelling salesman problem using nearest neighbor algorithm. Int J Scientific Res Dev 3(12):856–859
10. Asani EO, Ayegba PO, Ayoola JA, Okeyinka AE, Adebiyi AA (2019) A preliminary study on the complexity of some heuristics for solving combinatorial optimization problems. Int J Eng Res Technol 12(10):1615–1620
11. Rao W, Jin C (2010) A new hybrid algorithm for solving TSP. Proceedings of ICLEM 2010: logistics for sustained economic development, pp 3327–3334

12. Lity S, Al-Hajjaji M, Thüm T, Schaefer I (2017) Optimizing product orders using graph algorithms for improving incremental product-line analysis. VAMOS '17: Proceedings of the eleventh international workshop on variability modelling of software-intensive systems, pp 60–67. https://doi.org/10.1145/3023956.3023961
13. García NLF, Martínez MD, Poyato AC, Cuevas FJM, Carnicer RM (2020) Unsupervised generation of polygonal approximations based on the convex hull. Pattern Recogn Lett 135:138–145. https://doi.org/10.1016/j.patrec.2020.04.014