IJIO
(*International Journal of Industrial Optimization*)
ISSN 2723-3022 (online) | 2714-6006 (printed)
VOL 4, No. 2, 131-144
https://doi.org/10.12928/ijio.v4i2.7875

Augmented tour construction heuristics for the travelling salesman problem

Ziauddin Ursani [1], Ahsan Ahmad Ursani [2,*]
[1] Swansea University, United Kingdom
[2] Mehran University of Engineering and Technology, Pakistan
*Corresponding Author: ahsan.ursani@faculty.muet.edu.pk

ARTICLE INFO

Article history
Received: March 22, 2023
Revised: April 24, 2023
Accepted: June 29, 2023

Keywords
Travelling Salesman Problem;
Tour Construction Heuristics;
Complexity Analysis

ABSTRACT

Tour construction heuristics serve as fundamental techniques in optimizing the routes of a traveling salesman. These heuristics remain significant as foundational methods for generating initial solutions to the Traveling Salesman Problem (TSP), facilitating subsequent applications of tour improvement heuristics. These heuristics effectively comprise the iterative application of city node selection and insertion. However, thus far, no attempts have been made to enhance the basic structure of tour construction heuristics to bring a better initial solution for the advanced heuristics. This study aims to enhance tour construction heuristics without compromising their theoretical complexity. Specifically, an iterative step of partial tour deconstruction has been introduced to the existing heuristics. This additional step has been implemented and evaluated with three highly performing tour construction heuristics: the farthest insertion heuristic, the max difference insertion heuristic, and the fast max difference insertion heuristic. The results demonstrate that augmenting these heuristics with the partial tour deconstruction step improves the best, worst, and average solutions while preserving their theoretical complexity.

INTRODUCTION

The travelling salesman problem (TSP) consists of finding the shortest possible tour of n city-nodes. The exponential time algorithms for the solution of this problem have been proposed [1-3]. However, polynomial time algorithm for this problem is yet to be devised and caries a prize money of one million dollars from Clay Mathematical Institute being one of the seven millennium prize problems [4]. The earliest reference to this problem can be found in the German handbook of travelling salesman [5]. Its first formal definition came from Karl Menger in 1932 [6], where he also observed the computational difficulty of the problem. The problem has remained a subject of curiosity not only inside but also outside of scientific realm. The problem has attracted so much attention that a feature film titled “Travelling Salesman” was also released in 2012. The story of the film revolved around four mathematicians who solved this problem but were weary of its disclosure due to their concerns about its ethical consequences. The TSP serves as a default benchmark testing ground for any combinatorial optimisation method or even now compilers [7]. It is sometimes referred to as mother of all combinatorial optimisation problems. This is because many complex combinatorial problems have shown to be reduced to TSP. Therefore, it is understood that if TSP is solved all the

combinatorial optimisation problems will be solved. TSP finds applications in wide variety of practical areas such as printed circuit boards [8], gas turbine engines [9], X-Ray crystallography [10], Computer wiring [11], warehouses [12], healthcare services [13], micro-aggregation [14], Trajectory Planning [15], helicopter patrolling [16], solar panels diagnostic reconnaissance [17], cutting path optimization [18], storage/retrieval system [19], logistics [20], autonomous underwater vehicles [21], air logistics [22], tourist route paths [23], chemical shipping [24], and transport route optimisation [25], etc. The slight variants of TSP have further applications such as police patrolling [26].

Owing to the difficulty in finding the optimal solution to the problem, researchers have resorted to heuristics with downgraded objective of finding a good solution rather than the best solution. The general structure of these heuristics consists of two parts i.e., tour construction heuristic see e.g. [27] and tour improvement heuristic see e.g. [28]. The tour construction heuristic starts the tour from scratch, i.e., it establishes a subtour of one city-node and then expands the subtour by adding another city-node iteratively, until the accomplishment of the complete tour. The tour improvement heuristic improves the tour obtained from tour construction heuristic by iteratively applying one operation at a time. These operations sometimes become so complex that they involve partial tour deconstruction and then its construction again. The most popular work in this direction is the Lin-Kernighan heuristic [29].

The research objective is to enhance tour construction heuristics without compromising their theoretical complexity. Specifically, an iterative step of partial tour deconstruction has been introduced to the existing heuristics. This paper explores the possibility of carrying out the process of partial tour deconstruction during the construction phase itself. This approach is hereby referred to as the augmented tour construction heuristic. This paper also contributes to the complexity analysis of the proposed approach.

METHOD

1. Revisiting Tour Construction Heuristics

The tour construction heuristics have proved helpful in three different ways. First, they provide initial solutions for the tour improvement heuristics. Second, they are the source of study of upper bounds on the optimal solutions [30], and lastly, they have also provoked exhaustive empirical studies to understand how the structure of optimal solutions differs from the solutions arising from the tour construction heuristics [31-32]. Any tour construction heuristic can be described in four elementary steps as follows. Then, it can be transformed into flowchart as shown in Figure 1.

a. Subtour establishment rule to obtain a small subtour.
b. City selection rule to choose new city for its addition in the subtour.
c. Subtour expansion rule to include selected city in the subtour.
d. Iterative application of rule b and rule c until complete tour is obtained.

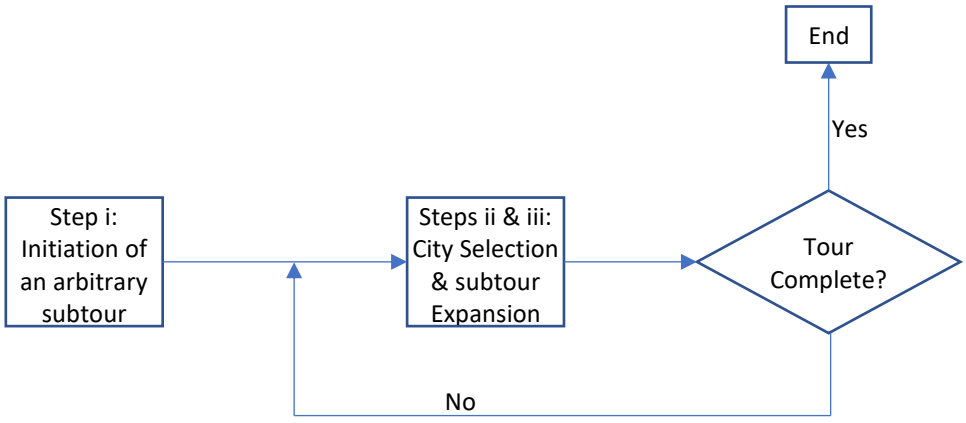

Figure 1. Flowchart of tour construction heuristic

0.12928/ijio.v4i2.7875

Since these heuristics develop tours by inserting each city-node iteratively, they are sometimes referred to as successive augmentation heuristics, e.g., Johnson and McGeoch [33]. Based on subtour expansion rule, the heuristics can be divided into two categories i.e., Goutham [34] and addition heuristics. If the subtour expansion rule is based on the cost of the subtour, then the heuristic is called the insertion heuristic, but if it is based on the next hop distance [35], it is called the addition heuristic. Experimentally performance of insertion heuristics has proved better than addition heuristics [36]. Therefore, the work presented herein considers the insertion heuristics only. In our earlier work [37], we developed complexity curtailing techniques for the tour construction heuristics, which included cheapest see e.g. Hassin et al. [38], largest, and max-difference insertion heuristics, effectively curtailing their complexity from $\mathcal{O}(n^3)$ *to* $\mathcal{O}(n^2)$, with statistically no deterioration in the quality of a solution. In doing so, we compared their performance with the farthest insertion heuristic, e.g., Renaud et al. [39]. Max Difference Insertion Heuristic and its faster version proved better than Farthest Insertion Heuristic, which was considered best then. In this paper, these three insertion heuristics are considered for further development. Therefore, these heuristics must be described in their current form first.

1.1 Farthest Insertion Heuristic (FIH)

In this heuristic, the first rule of subtour establishment consists of the random selection of a city-node. The second rule of city-node insertion can be described as follows. Let τ be the set of city-nodes in the subtour and let τ' be the set of city-nodes not in the subtour. Let city-node $k \in \tau'$ and city-node $h \in \tau$. Let $d(h,k)$ be the distance between city-nodes k and h. The city-node k is selected for insertion (subtour expansion) as presented in Equation (1).

$$d(h,k) = max\forall_{j\in\tau'}\{min\forall_{i\,\in\,\tau}(d\,(i,j))\} \tag{1}$$

The third rule of the city-node insertion is defined as follows. Let $k \in \tau'$ and $(i,i') \in \tau$, such that i and i' are a pair of consecutive city-nodes in τ. Let $d(i,k)$ be the distance between the city-nodes i and k. The subtour is expanded by inserting the selected city-node k in the subtour τ as shown in Equation (2).

$$cost(\tau,k) = \min[\forall(i,i') \in \tau\{d(i,k) + d(i',k) - d(i,i')\}] \tag{2}$$

where, $cost(\tau,k)$ represents the cost of insertion of city-node k in tour .
The FIH is characterized as distance-based heuristic because its second step of city-node insertion is based on distance metric rather than the cost metric.

1.2 Max Difference Insertion Heuristic (MDIH)

The first rule of subtour establishment in this heuristic consists of constructing the subtour of three city-nodes. The construction of this subtour is itself a tour construction heuristic which is commonly known as the largest insertion heuristic. In the largest insertion heuristic, the first rule of subtour establishment is a randomly chosen city-node. The second rule of a city-node insertion consists of selecting a city-node k which makes the largest insertion in the current subtour. This can be represented in Equation (3).

$$cost(\tau,k) = max\forall_{j\in\tau'}\{min\forall_{(i\in\tau)}(cost(i,j))\} \tag{3}$$

The description of symbols remains the same as described earlier for Equations (1) – Equation (2). The third rule of city-node insertion remains the same as that of Equation (2). This largest insertion heuristic is used only up to subtour of three city-nodes, which constitutes first rule of subtour establishment for the max-difference insertion heuristic. The initial subtour,

in this case, comprises at least three city-nodes because the second rule of city-node selection requires minimum of three edges of the subtour for its meaningful implementation. The second rule of a city-node insertion is a bit complicated. To understand this, we need to define the function n^{th} minimum, denoted as $min_n\{\forall \mathrm{V} \in \mathrm{U}(V)\}$. The meaning of "$n^{\text{th}}$ minimum" is that if all the values V in set U are sorted from minimum to maximum, then this function represents the n^{th} value in that list. The "n^{th} expansion cost" of a subtour by the insertion of a new city-node is equal to the function $min_n\{\forall \mathrm{V} \in \mathrm{U}(V)\}$, where the set U is the set of all possible expansion costs of the subtour that could be caused by the insertion of the city-node on each of its edges. Let $k \in \tau'$ and $(i, i') \in \tau$, such that i and i' are a pair of consecutive city-nodes in τ. Let $d(i,k)$ be the distance between the city-nodes i and k. Then the n^{th} expansion cost of the tour τ by the insertion of city-node k is given by Equation (4).

$$cost_n(\tau, k) = min_n[\forall (i, i') \in \tau\{d(i,k) + d(k, i') - d(i, i')\}] \quad (4)$$

Where, the edge (i, i') is called n^{th} expansion edge. It should be noted that equation 2 is the representative equation for first expansion cost i.e., $n = 1$. The Equation (4) represents city-node insertion rule of MDIH, where the city-node k is selected such that the cost difference $D(k)$ between its expansion cost and 2nd expansion cost is maximum among all the city-nodes not in the tour as presented in Equation (5).

$$D(k) = \max[\forall \mathrm{k} \in \tau'\{cost_2(\tau, k) - cost(\tau, k)\}] \quad (5)$$

The third rule regarding insertion of a city-node remains the same as that of equation 2. The MDIH is characterized as cost-based insertion heuristic as its city-node selection rule depends on the insertion cost of the city-node.

1.3 Fast Max Difference Insertion Heuristic (FMDIH)

This is a faster version of MDIH. All three rules, i.e., subtour establishment, city-node selection and city-node insertion in this heuristic, remain the same as that of the Max-Difference Insertion Heuristic. The only difference is that this heuristic performs some shortcuts in the second rule at the cost of losing some exactness. It can be seen from equation 5 that the second rule depends on expansion and the second expansion cost for the insertion of the new city-node in the subtour. These costs can be computed efficiently by comparing them from the previous iteration with the expansion cost on two newly formed edges due to the last inserted city-node. However, if the last inserted city-node broke one of the expansions and second expansion edges for the city-node under consideration, then this requires computing the expansion cost of the city on each edge of the subtour again. It is assumed that this makes theoretical complexity of MDIH of the order n^3.

However, in this paper we consider the number of times re-computation of expansion cost of the city-node is needed to articulate precise complexity of heuristic. The fast MDIH proposes that for each city-node record of first, second and third expansion cost should be kept. This record should be updated in each iteration by comparing them with the expansion costs on two newly formed edges for each city-node. If any of the three expansion edges is broken due to last inserted city-node in the subtour, we just compare rest of the two expansion costs with expansion cost on two newly formed edges and take best three out of four as the first second and third expansion cost. This ensures skipping the computation of expansion cost on each edge of the subtour reducing the complexity of the algorithm to $\mathcal{O}(n^2)$. The results show that the quality of solutions remains the same as that of MDIH, even though the exactness of the procedure is sacrificed. For further details on MDIH and FMDIH, readers are referred to Ursani et al. [37]. Like MDIH, FMDIH is also a cost-based insertion heuristic.

0.12928/ijio.v4i2.7875

2. Augmented Tour Construction Heuristic (ATCH)

ATCH is a proposed tour construction heuristic with an additional step of partial tour deconstruction. The elementary stages of the augmented tour construction heuristic are as follows. Then, it can be transformed into a flowchart, as shown in Figure 2.

a. Subtour establishment rule: to obtain a small subtour.
b. City selection rule: to choose new city for its addition in the subtour.
c. Subtour expansion rule: to include selected city in the subtour.
d. City ejection rule: to eject some cities from the subtour (Partial Tour Deconstruction).
e. Iterative application of rules b-d until complete tour is obtained.

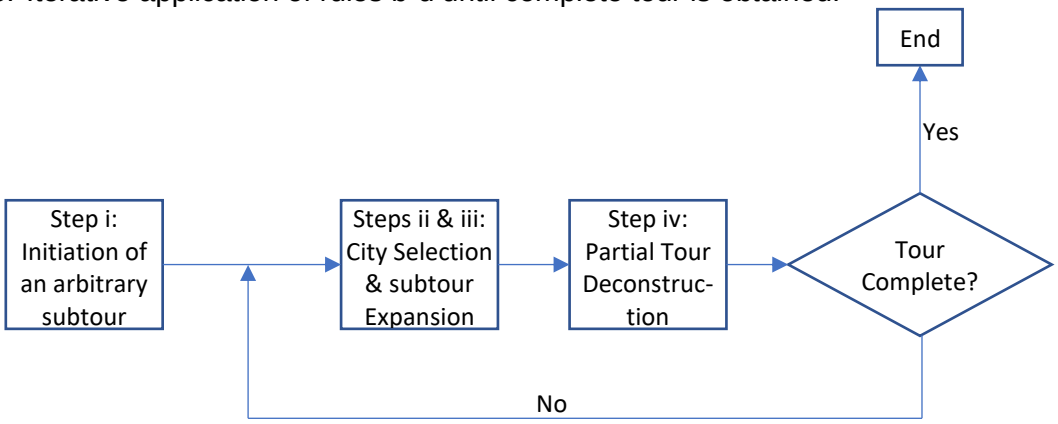

Figure 2. Flowchart of ATCH algorithm

Equation (6) and Equation (7) expose the city-node omission or ejection criteria in the partial deconstruction step. Any city-node is ejected for which at least one of the two expressions turn out to be true.

$$\forall_{i\in\tau, i\neq j}\, d(i-1,i)+d(i,i+1)+d(j-1,j) > d(i-1,i+1)+d(j-1,i)+d(i,j) \quad (6)$$

$$\forall_{i\in\tau, i\neq j}\, d(i-1,i)+d(i,i+1)+d(j,j+1) > d(i-1,i+1)\ +d(j,i)+d(i,j+1) \quad (7)$$

where,
j = ID of last city-node inserted in the subtour.
$j-1$ = ID of city-node preceding to city-node j in the subtour.
$j+1$ = ID of city-node succeeding to city-node j in the subtour.
i = Any city-node in the subtour other than city-node j.
The Equation (6) and Equation (7) compare the benefit of omitting a city-node with the cost of reinserting it into the one of the newly formed edges. The city-node is omitted if benefit overweighs cost.

3. Complexity Analysis of Tour Construction Heuristics

The complexity of distance-based tour construction heuristics such as FIH involves two components i.e., city-node selection and city-node insertion. The former is trivial as it is just a random selection of a city-node, which does not add anything to complexity. The latter involves $\frac{n(n-1)}{2}$ computations. Therefore, the complexity of FIH is of the order n^2. In FMDIH, cost of every city-node not in the subtour is computed on the newly formed two edges. This amounts to $n(n-2)$ cost computations. Therefore, complexity of FMDIH is $\mathcal{O}(n^2)$. The complexity of MDIH is $\mathcal{O}(n^3)$, if cheapest and second cheapest expansion cost of each city-node is computed on each edge of subtour in each iteration. However, its smart implementation can curtail its complexity. If cheapest and second cheapest expansion cost of each city-node is stored, then their expansion cost will only be computed on two newly formed edges of the subtour in each iteration. However, sometimes computation of expansion cost on all edges of the subtour will still be required in case where new edges of the subtour are formed by

breaking the first or second expansion edge of the city-node. Therefore, it computes n^{γ} insertion costs, where $2 < \gamma < 3$. This curtails the complexity of MDIH to $\mathcal{O}(n^2 \log n)$. Since, augmented tour construction heuristics have one additional step of partial deconstruction, which involves omission of a city-node and requires computation of modification cost of the subtour, the complexity of Augmented Farthest Insertion Heuristic involves cost of the omission of a city-node, i.e. $\rho n(n-4)$ in addition to that of insertion of a city-node, i.e. $\frac{\rho n(n-1)}{2}$, whereas ρ is a parameter whose value depends on the number of omitted city-nodes. Therefore, complexity of augmented farthest insertion heuristic is $\mathcal{O}(\rho n^2)$. Similarly, the complexity of augmented fast max-difference heuristic is $\mathcal{O}(\rho n^2)$. We will later get rid of ρ in our analysis. The complexity of augmented max difference insertion heuristic depends upon n^{γ} insertion cost, where $2 < \gamma < 3$, and $\rho n(n-4)$ omission costs. Therefore, the complexity of augmented max difference insertion heuristic is $\mathcal{O}(\rho n^2 \log n)$. The experimental results presented later prove that the parameter ρ remains unaffected by the size of the problem.

RESULTS AND DISCUSSION

The experiments were conducted on six heuristics, namely farthest insertion heuristic (FIH), max-difference insertion heuristic (MDIH) [37], fast max-difference insertion heuristic (FMDIH) [37], augmented farthest insertion heuristic (AFIH), augmented max-difference insertion heuristic (AMDIH) and augmented fast max-difference insertion heuristic (AFMDIH). All the heuristics were coded in C/C++, in Microsoft Visual Studio compiler. The experiments were run on Intel core I5, HP laptop with 12 GB memory and windows 10 (64-bit) operating system. The heuristics were tested on 109 datasets of sizes in the range of 14-15112 cities. The test datasets were taken from most popular test bed TSPlib [40]. The program was run for 30 independent simulations on random seeds on each dataset. The results were recorded for best, worst, and average solution of 30 run on each dataset along with standard deviation and mean execution time. For augmented heuristics, the mean number of omitted city-nodes per run due to city-node omission rule were also recorded to estimate their computational complexity. Summary of results are presented for all the above mentioned six heuristics. However, the detailed results on each dataset are provided for two heuristics FMDIH and AFMDIH. Table 1 contains summary of results on all the six heuristics. Table 1 lists the six heuristics and statistics on their performance over 109 datasets. The first column from the left shows the mean of the best solutions out of 30 simulations. This value is provided in terms of %age difference from the optimal solution. Similarly, the following columns show mean of the worst solutions, mean of average solutions, mean of the standard deviation taken as percentage of optimal solution, and mean of execution times.

Table 1. Summary of results for six heuristics

Name	Mean best solution percentage	Mean worst solution percentage	Mean average solution percentage	Mean standard deviation percentage	Mean execution time
FIH	7.34	15.38	10.27	0.92	9.96
MDIH [37]	4.52	8.66	6.36	0.44	6.79
FMDIH [37]	4.69	10.41	6.71	0.69	2.34
AFIH	4.14	8.47	6.04	0.48	24.99
AMDIH	3.58	7.04	5.13	0.37	24.81
AFMDIH	3.65	7.11	5.18	0.37	20.51

Looking at Table 1, one finds that augmented versions of heuristics have produced better solutions in comparison to traditional versions. In case of FIH, average solutions improve by a huge margin from 10.27% to 6.04%. Similarly, in case of MDIH and FMDIH heuristics, the

0.12928/ijio.v4i2.7875

results improve from 6.36% to 5.13% and from 6.71% to 5.18%, respectively. Similar improvements are found in the mean of best and the mean of worst solutions. Up to 3% improvement is achieved in case of the mean of best solution and up to 7% improvement is attained in case of the mean of worst solution. A great improvement of up to 0.4% can also be seen in the column of standard deviation. However, these improvements are achieved at the cost of computational time which is around four times more in case of augmented heuristics. However, computational time does not offer any help in estimating the computational complexity of the augmented heuristics. Instead, one can investigate the effects of the problem size on parameter ρ, discussed in the section 4.

Figure 3 shows parameter ρ as it varies with the problem size in terms of number of city-nodes. The graphs of all the augmented heuristics i.e., AFIH, AMDIH and AFMDIH despite having initial fluctuations on the smaller datasets they are almost parallel to x-axis. This shows that parameter ρ, which is an indicator of the algorithmic complexity, remains constant as the problem size grows. Therefore, complexities of AFIH, AFMDIH and AMDIH can be rounded to $\mathcal{O}(n^2)$, $\mathcal{O}(n^2)$ and $\mathcal{O}(n^2 \log n)$ respectively, as discussed in section 4. Table 2 presents detailed comparative results of FMDIH and AFMDIH on each dataset.

The results in Table 2 mention minimum, maximum, average, and standard deviation of 30 simulations on 109 datasets. The results also show their percentage differences from the optimal solutions. It also shows computational time. The last row shows the mean of results on 109 datasets. From the last row it can be seen from the table that augmented heuristic has beaten minimum, maximum, and average cost solutions by approximately 1%, 3.3% and 1.5% respectively. It has also reduced standard deviation by 0.3%. However, it has increased computational time eight times. Though theoretical complexity of both schemes remains the same, as concluded in discussion above by examining the graph in Figure 1.

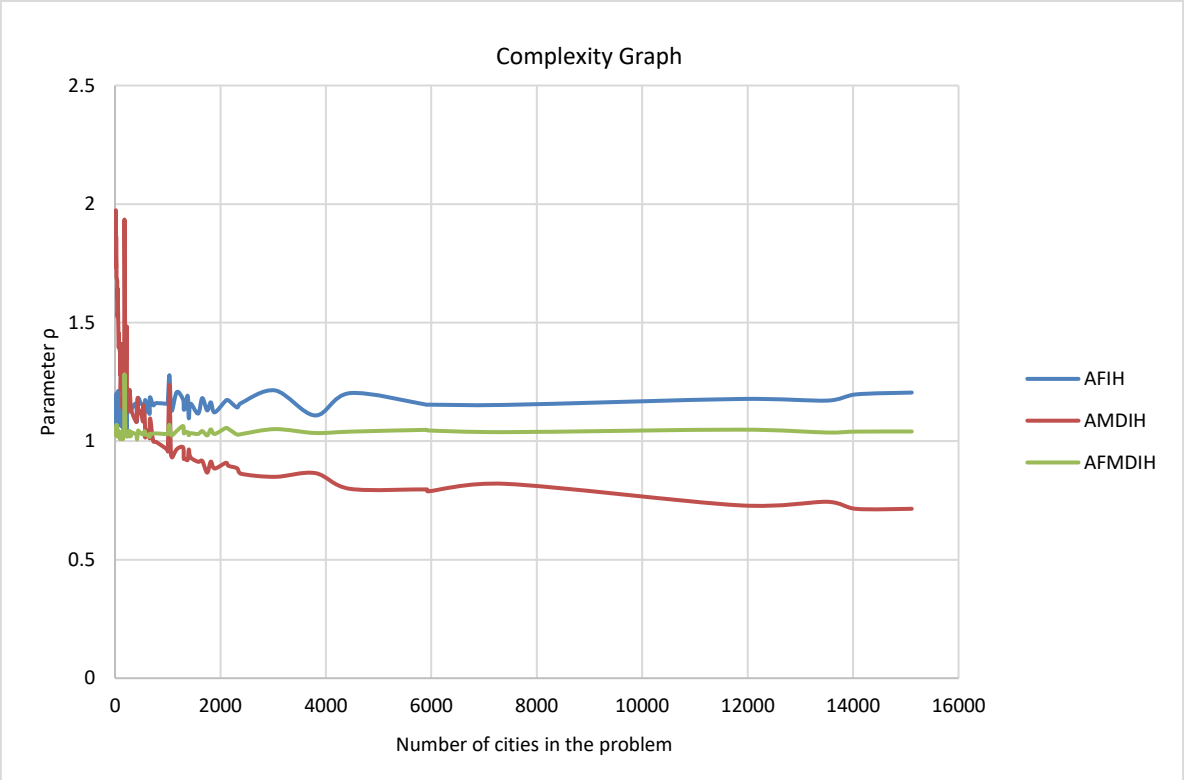

Figure 3. Graph of percentage of average number of ejected cities per simulation against problem size

Table 2. Details of Results on 109 Datasets for FMDIH and AFMDIH

Dataset	Optimal	Fast Max Difference Insertion Heuristic [37]									Augmented Fast Max Difference Insertion Heuristic								
		Min	%Diff	Max	%Diff	Avg.	%Diff	St. Dev	%age	Time	Min	%Diff	Max	%Diff	Avg.	%Diff	St.Dev	%Diff	Time
Burma14	3323	3323	0.00	3442	3.58	3330.93	0.24	29.68	0.89	0.00	3323	0.00	3323	0.00	3323	0.00	0.00	0.00	0.00
Ulysses16	6859	6859	0.00	7082	3.25	6902.93	0.64	51.17	0.75	0.00	6859	0.00	6903	0.64	6889.8	0.45	27.61	0.40	0.00
Gr17	2085	2085	0.00	2095	0.48	2090	0.24	5.00	0.24	0.00	2085	0.00	2095	0.48	2088.67	0.18	6.40	0.31	0.00
GR21	2707	2707	0.00	2707	0.00	2707	0.00	0.00	0.00	0.00	2707	0.00	2707	0.00	2707	0.00	0.00	0.00	0.00
Ulysses22	7013	7013	0.00	7262	3.55	7074.13	0.87	63.86	0.91	0.00	7013	0.00	7262	3.55	7090.3	1.10	74.10	1.06	0.00
GR24	1272	1272	0.00	1341	5.42	1310.8	3.05	19.75	1.55	0.00	1272	0.00	1323	4.01	1298	2.04	20.63	1.62	0.00
Fri26	937	937	0.00	965	2.99	944.8	0.83	9.70	1.04	0.00	937	0.00	955	1.92	943.267	0.67	8.87	0.95	0.00
Bayg29	1610	1624	0.87	1668	3.60	1633.7	1.47	9.45	0.59	0.00	1610	0.00	1624	0.87	1621	0.68	5.70	0.35	0.00
Bays29	2020	2026	0.30	2094	3.66	2040.53	1.02	13.08	0.65	0.00	2026	0.30	2068	2.38	2033.63	0.67	10.08	0.50	0.00
Dantzig42	699	705	0.86	744	6.44	715.1	2.30	12.80	1.83	0.00	699	0.00	738	5.58	707.5	1.22	7.40	1.06	0.00
Swiss42	1273	1273	0.00	1346	5.73	1292.8	1.56	20.27	1.59	0.00	1273	0.00	1334	4.79	1287.13	1.11	16.58	1.30	0.00
ATT48	10628	10653	0.24	11073	4.19	10935.1	2.89	88.58	0.83	0.00	10653	0.24	10959	3.11	10902.3	2.58	60.11	0.57	0.00
GR48	5046	5103	1.13	5295	4.93	5191.07	2.87	66.63	1.32	0.00	5103	1.13	5295	4.93	5174.03	2.54	34.96	0.69	0.00
HK48	11461	11496	0.31	11821	3.14	11590.8	1.13	69.03	0.60	0.00	11461	0.00	11721	2.27	11567.2	0.93	32.76	0.29	0.00
Eil51	426	435	2.11	443	3.99	439.633	3.20	2.44	0.57	0.00	433	1.64	443	3.99	438.4	2.91	2.27	0.53	0.00
Berlin52	7542	7604	0.82	8391	11.26	7939.3	5.27	251.51	3.33	0.00	7604	0.82	8256	9.47	7887.63	4.58	183.75	2.44	0.00
Brazil58	25395	25717	1.27	26551	4.55	26292.7	3.53	228.63	0.90	0.00	25655	1.02	26123	2.87	25943.7	2.16	128.45	0.51	0.00
ST70	675	693	2.67	719	6.52	701.467	3.92	7.99	1.18	0.00	684	1.33	712	5.48	695.467	3.03	4.91	0.73	0.00
Eil76	538	556	3.35	577	7.25	567.433	5.47	5.50	1.02	0.00	553	2.79	572	6.32	562.667	4.58	3.61	0.67	0.00
PR76	108159	109146	0.91	115386	6.68	111767	3.34	2029.03	1.88	0.00	109011	0.79	114553	5.91	111257	2.86	1184.75	1.10	0.00
GR96	55209	56101	1.62	58698	6.32	57606.5	4.34	680.90	1.23	0.00	55847	1.16	58641	6.22	56979.8	3.21	461.06	0.84	0.00
Rat99	1211	1253	3.47	1303	7.60	1272.4	5.07	12.85	1.06	0.00	1250	3.22	1287	6.28	1266.87	4.61	5.46	0.45	0.00
KroA100	21282	21478	0.92	22944	7.81	21694.1	1.94	370.40	1.74	0.00	21417	0.63	22944	7.81	21665.9	1.80	205.38	0.97	0.00
KroB100	22141	22502	1.63	23390	5.64	22920.2	3.52	206.79	0.93	0.00	22364	1.01	23181	4.70	22701.9	2.53	130.69	0.59	0.00
KroC100	20749	20819	0.34	21821	5.17	21017.6	1.29	179.30	0.86	0.00	20819	0.34	21471	3.48	20993.1	1.18	80.72	0.39	0.00
KroD100	21294	21403	0.51	22965	7.85	21955.1	3.10	349.65	1.64	0.00	21403	0.51	22439	5.38	21860.5	2.66	170.10	0.80	0.00
KroE100	22068	22179	0.50	23236	5.29	22775.9	3.21	308.37	1.40	0.00	22179	0.50	23043	4.42	22631	2.55	137.13	0.62	0.00
RD100	7910	7993	1.05	8380	5.94	8176.97	3.38	115.58	1.46	0.00	7958	0.61	8331	5.32	8136.17	2.86	59.99	0.76	0.00
Eil101	629	657	4.45	683	8.59	665.733	5.84	7.03	1.12	0.00	653	3.82	671	6.68	658.767	4.73	2.50	0.40	0.00
Lin105	14379	14379	0.00	15179	5.56	14697.6	2.22	158.93	1.11	0.00	14379	0.00	14872	3.43	14572.3	1.34	70.68	0.49	0.00
PR107	44303	44497	0.44	45177	1.97	44906	1.36	177.29	0.40	0.00	44438	0.30	45154	1.92	44842.5	1.22	102.23	0.23	0.00
GR120	6942	7095	2.20	7507	8.14	7342.33	5.77	105.87	1.53	0.00	7021	1.14	7497	7.99	7253.33	4.48	49.33	0.71	0.00
PR124	59030	59767	1.25	62202	5.37	60457.1	2.42	623.37	1.06	0.00	59767	1.25	60916	3.19	60202	1.99	180.32	0.31	0.00
Bier127	118282	121592	2.80	128374	8.53	123776	4.64	1530.06	1.29	0.00	120860	2.18	125732	6.30	122976	3.97	537.75	0.45	0.00
CH130	6110	6397	4.70	6609	8.17	6485.43	6.14	59.30	0.97	0.00	6254	2.36	6572	7.56	6413.07	4.96	33.87	0.55	0.00
PR136	96772	99925	3.26	104611	8.10	102881	6.31	1132.21	1.17	0.00	99637	2.96	102810	6.24	101473	4.86	396.07	0.41	0.00
GR137	69853	71465	2.31	75314	7.82	72654.1	4.01	737.06	1.06	0.00	70725	1.25	74670	6.90	72247.1	3.43	345.58	0.49	0.00
PR144	58537	58972	0.74	64881	10.84	59903.5	2.33	1270.40	2.17	0.00	58972	0.74	61207	4.56	59689	1.97	318.36	0.54	0.00
CH150	6528	6683	2.37	7061	8.16	6907.3	5.81	99.39	1.52	0.00	6636	1.65	7019	7.52	6838.23	4.75	46.27	0.71	0.00

10.12928/ijio.v4i2.7875

Dataset	Optimal	Fast Max Difference Insertion Heuristic [37]									Augmented Fast Max Difference Insertion Heuristic								
		Min	%Diff	Max	%Diff	Avg.	%Diff	St. Dev	%age	Time	Min	%Diff	Max	%Diff	Avg.	%Diff	St.Dev	%Diff	Time
kroA150	26524	26991	1.76	28864	8.82	27708.5	4.47	457.96	1.73	0.00	26953	1.62	28640	7.98	27504.2	3.70	153.05	0.58	0.00
kroB150	26130	26497	1.40	28412	8.73	26988.3	3.28	469.82	1.80	0.00	26224	0.36	27325	4.57	26719.1	2.25	124.68	0.48	0.00
PR152	73682	74029	0.47	75558	2.55	74411.7	0.99	418.83	0.57	0.00	74029	0.47	75558	2.55	74315.2	0.86	200.98	0.27	0.00
U159	42080	43530	3.45	46846	11.33	44430.6	5.59	903.58	2.15	0.00	43271	2.83	46177	9.74	44040.4	4.66	292.69	0.70	0.00
SI175	21407	21565	0.74	21927	2.43	21725.6	1.49	107.24	0.50	0.00	21459	0.24	21786	1.77	21604.3	0.92	32.32	0.15	0.00
BRG180	1950	2710	38.97	6470	231.79	3429	75.85	1330.53	68.23	0.00	2290	17.44	2570	31.79	2413.33	23.76	33.17	1.70	0.00
Rat195	2323	2480	6.76	2598	11.84	2532.37	9.01	30.08	1.30	0.00	2453	5.60	2573	10.76	2501.3	7.68	11.26	0.48	0.00
d198	15780	16033	1.60	16375	3.77	16166.7	2.45	93.99	0.60	0.00	15961	1.15	16227	2.83	16046.8	1.69	28.57	0.18	0.00
KroA200	29368	30191	2.80	31632	7.71	30973.3	5.47	375.85	1.28	0.00	29981	2.09	31253	6.42	30596.7	4.18	119.87	0.41	0.00
KroB200	29437	30450	3.44	31768	7.92	30965.3	5.19	337.61	1.15	0.00	30304	2.95	31218	6.05	30697.8	4.28	120.87	0.41	0.00
GR202	40160	41280	2.79	43782	9.02	42256.8	5.22	650.87	1.62	0.00	41086	2.31	43111	7.35	41895.2	4.32	238.51	0.59	0.00
TS225	126643	135518	7.01	138827	9.62	136685	7.93	1084.78	0.86	0.00	134142	5.92	137551	8.61	135960	7.36	360.91	0.28	0.00
TSP225	3916	4118	5.16	4265	8.91	4194.13	7.10	32.14	0.82	0.00	4076	4.09	4220	7.76	4162.87	6.30	12.89	0.33	0.00
PR226	80369	81426	1.32	83085	3.38	82460.9	2.60	364.08	0.45	0.00	81344	1.21	82360	2.48	81627.8	1.57	75.32	0.09	0.00
GR229	134602	139251	3.45	144877	7.63	141730	5.30	1727.52	1.28	0.00	137507	2.16	142602	5.94	140171	4.14	460.57	0.34	0.00
Gil262	2378	2457	3.32	2556	7.49	2511.67	5.62	27.76	1.17	0.00	2449	2.99	2536	6.64	2490.83	4.74	7.71	0.32	0.00
PR264	49135	53080	8.03	55170	12.28	54204.4	10.32	542.66	1.10	0.00	52799	7.46	54941	11.82	53824.9	9.54	206.53	0.42	0.00
A280	2579	2707	4.96	2846	10.35	2785.63	8.01	36.11	1.40	0.00	2668	3.45	2782	7.87	2737.63	6.15	9.42	0.37	0.00
PR299	48191	48763	1.19	51552	6.97	50260.9	4.30	635.94	1.32	0.00	48572	0.79	51310	6.47	49786.7	3.31	197.94	0.41	0.00
Lin318	42029	43604	3.75	46054	9.58	44655.2	6.25	551.09	1.31	0.00	43326	3.09	45321	7.83	44074.5	4.87	144.53	0.34	0.00
Linhp318	41345	43103	4.25	45550	10.17	44513.6	7.66	506.62	1.23	0.00	42953	3.89	44717	8.16	43937.4	6.27	140.95	0.34	0.00
RD400	15281	15923	4.20	16477	7.83	16185.6	5.92	117.96	0.77	0.00	15825	3.56	16304	6.69	16046.3	5.01	30.36	0.20	0.00
FL417	11861	11970	0.92	12256	3.33	12162.1	2.54	61.78	0.52	0.00	11947	0.73	12232	3.13	12147.4	2.41	16.59	0.14	0.00
GR431	171414	177984	3.83	190656	11.23	181194	5.71	2673.55	1.56	0.00	176498	2.97	186166	8.61	179409	4.66	578.02	0.34	0.00
PR439	107217	110510	3.07	117652	9.73	114438	6.73	1756.56	1.64	0.00	110111	2.70	115646	7.86	112791	5.20	354.96	0.33	0.00
PCB442	50778	54075	6.49	56938	12.13	55266.8	8.84	695.06	1.37	0.00	53287	4.94	55430	9.16	54338.5	7.01	118.32	0.23	0.00
D493	35002	36402	4.00	37550	7.28	37039.6	5.82	291.92	0.83	0.00	36251	3.57	37224	6.35	36703.8	4.86	62.88	0.18	0.00
ATT532	27686	28786	3.97	29594	6.89	29346.8	6.00	194.95	0.70	0.00	28679	3.59	29495	6.53	29082	5.04	46.12	0.17	0.01
Ali535	202339	207794	2.70	224909	11.15	216249	6.87	3526.80	1.74	0.00	206986	2.30	221538	9.49	214072	5.80	712.72	0.35	0.01
SI535	48450	48987	1.11	49304	1.76	49152.8	1.45	84.77	0.17	0.00	48869	0.86	49189	1.53	49025.1	1.19	19.50	0.04	0.01
PA561	2763	3030	9.66	3147	13.90	3099.37	12.17	29.96	1.08	0.00	2984	8.00	3079	11.44	3034.43	9.82	5.62	0.20	0.01
U574	36905	38724	4.93	40101	8.66	39178.6	6.16	283.95	0.77	0.00	38237	3.61	39427	6.83	38800.7	5.14	60.73	0.16	0.01
Rat575	6773	7216	6.54	7356	8.61	7298.13	7.75	40.92	0.60	0.00	7155	5.64	7307	7.88	7223.53	6.65	10.31	0.15	0.01
P654	34643	35379	2.12	36157	4.37	35975.8	3.85	202.18	0.58	0.00	35231	1.70	36282	4.73	35907.5	3.65	44.79	0.13	0.01
D657	48912	51475	5.24	53400	9.18	52272	6.87	409.12	0.84	0.00	51172	4.62	52678	7.70	51827.3	5.96	74.83	0.15	0.01
GR666	294358	311171	5.71	325575	10.61	315951	7.34	3065.80	1.04	0.00	306692	4.19	320896	9.02	312390	6.13	637.50	0.22	0.01
U724	41910	44279	5.65	45417	8.37	44949.8	7.25	262.92	0.63	0.00	43959	4.89	44883	7.09	44477.2	6.13	53.49	0.13	0.01
Rat783	8806	9406	6.81	9642	9.49	9505.9	7.95	57.24	0.65	0.01	9282	5.41	9498	7.86	9383.97	6.56	10.50	0.12	0.02

Dataset	Optimal	Fast Max Difference Insertion Heuristic [37]									Augmented Fast Max Difference Insertion Heuristic								
		Min	%Diff	Max	%Diff	Avg.	%Diff	St. Dev	%age	Time	Min	%Diff	Max	%Diff	Avg.	%Diff	St.Dev	%Diff	Time
DSJ1000	18659688	19707600	5.62	20167600	8.08	19909600	6.70	109255.0	0.59	0.01	19502600	4.52	19959500	6.97	19714100	5.65	19524.70	0.10	0.04
DSJ1000Ceil	18660188	19748800	5.83	20092700	7.68	19921700	6.76	106247.00	0.57	0.01	19500000	4.50	19950200	6.91	19710800	5.63	18549.00	0.10	0.03
Pr1002	259045	272338	5.13	278387	7.47	276065	6.57	1564.93	0.60	0.01	270099	4.27	275389	6.31	273113	5.43	219.75	0.08	0.03
Si1032	92650	95602	3.19	96325	3.97	96014.7	3.63	164.48	0.18	0.01	94716	2.23	95430	3.00	95121.6	2.67	33.51	0.04	0.03
U1060	224094	235663	5.16	242235	8.10	238125	6.26	1699.78	0.76	0.01	233007	3.98	239467	6.86	235688	5.17	239.05	0.11	0.04
Vm1084	239297	252247	5.41	261433	9.25	256233	7.08	2262.96	0.95	0.01	249526	4.27	257148	7.46	253125	5.78	332.04	0.14	0.05
Pcb1173	56892	61970	8.93	64127	12.72	63089.4	10.89	449.72	0.79	0.01	61368	7.87	62969	10.68	62185.8	9.30	53.60	0.09	0.05
D1291	50801	57900	13.97	60130	18.36	58752.4	15.65	579.59	1.14	0.01	56315	10.85	58393	14.94	57389	12.97	80.77	0.16	0.05
RL1304	252948	278739	10.20	293163	15.90	284520	12.48	3029.34	1.20	0.02	274058	8.35	288834	14.19	279808	10.62	422.27	0.17	0.06
RL1323	270199	296001	9.55	309100	14.40	303208	12.22	2464.82	0.91	0.02	292419	8.22	301960	11.75	296973	9.91	304.65	0.11	0.06
NRW1379	56638	60349	6.55	61646	8.84	60780.1	7.31	308.76	0.55	0.02	59657	5.33	60635	7.06	60110.8	6.13	33.50	0.06	0.06
FL1400	20127	20463	1.67	21209	5.38	20845.5	3.57	236.47	1.17	0.02	20333	1.02	20818	3.43	20550.1	2.10	19.20	0.10	0.06
U1432	152970	165664	8.30	169421	10.75	167179	9.29	888.93	0.58	0.02	163478	6.87	166693	8.97	164834	7.76	121.78	0.08	0.05
FL1577	22249	24417	9.74	25948	16.63	24878.1	11.82	353.17	1.59	0.02	24109	8.36	25317	13.79	24546.3	10.33	39.78	0.18	0.07
D1655	62128	67994	9.44	70461	13.41	69060.4	11.16	571.58	0.92	0.02	67052	7.93	69538	11.93	68000.4	9.45	82.42	0.13	0.08
VM1748	336556	355945	5.76	366383	8.86	360543	7.13	2068.97	0.61	0.03	351881	4.55	362621	7.74	355763	5.71	260.97	0.08	0.10
U1817	57201	64176	12.19	65964	15.32	64870.7	13.41	421.55	0.74	0.03	62686	9.59	64618	12.97	63539.5	11.08	53.22	0.09	0.10
RL1889	316536	348670	10.15	359254	13.50	353293	11.61	2324.25	0.73	0.03	343026	8.37	355356	12.26	348193	10.00	323.76	0.10	0.12
D2103	80450	92946	15.53	95928	19.24	94702.7	17.72	694.52	0.86	0.04	91031	13.15	94100	16.97	92908.3	15.49	94.31	0.12	0.14
U2152	64253	72237	12.43	74684	16.23	73268.2	14.03	533.71	0.83	0.05	70712	10.05	72853	13.38	71766.7	11.69	49.55	0.08	0.14
U2319	234256	245879	4.96	249237	6.40	247806	5.78	698.47	0.30	0.05	242170	3.38	244369	4.32	243402	3.90	59.87	0.03	0.16
Pr2392	378032	406594	7.56	416449	10.16	411182	8.77	2682.23	0.71	0.05	402888	6.58	409357	8.29	406441	7.51	225.30	0.06	0.18
Pcb3038	137694	151286	9.87	153967	11.82	152492	10.75	608.29	0.44	0.08	149006	8.22	150937	9.62	149917	8.88	42.10	0.03	0.32
FL3795	28772	31021	7.82	33579	16.71	32410.6	12.65	706.06	2.45	0.12	30506	6.03	33126	15.13	31842.2	10.67	63.28	0.22	0.56
FNL4461	182566	197003	7.91	198967	8.98	197923	8.41	408.59	0.22	0.16	194648	6.62	196081	7.40	195439	7.05	29.53	0.02	0.82
RL5915	565530	650532	15.03	661159	16.91	655440	15.90	2780.78	0.49	0.28	634848	12.26	646274	14.28	639135	13.02	198.77	0.04	1.64
RL5934	556045	635088	14.22	648038	16.54	640614	15.21	3105.01	0.56	0.28	619736	11.45	631437	13.56	625645	12.52	194.59	0.03	1.66
pla7397	23260728	25193000	8.31	25675000	10.38	25406100	9.22	122079.0	0.52	0.42	24802800	6.63	25242900	8.52	25038700	7.64	6537.00	0.03	2.47
rl11849	923288	1049180	13.64	1060600	14.87	1055600	14.33	3065.17	0.33	1.10	1026850	11.22	1040220	12.66	1033190	11.90	169.50	0.02	10.73
usa13509	19982859	21562300	7.90	21810500	9.15	21680500	8.50	57951.30	0.29	1.44	21325000	6.72	21482300	7.50	21400100	7.09	1837.15	0.01	12.34
brd14051	469385	506648	7.94	509902	8.63	508703	8.38	857.35	0.18	1.53	501011	6.74	503692	7.31	502305	7.01	32.55	0.01	12.96
d15112	1573084	1699920	8.06	1710430	8.73	1704070	8.33	2902.15	0.18	1.82	1678830	6.72	1686260	7.19	1682650	6.97	89.74	0.01	18.86
Average			**4.69**		**10.41**		**6.71**		**0.69**	**0.07**		**3.65**		**7.11**		**5.18**		**0.37**	**0.59**

10.12928/ijio.v4i2.7875

Furthermore, we will see that this eight times increase in the execution time is not much as compared to betterment in solution quality. In short benefit overweighs loss. To analyze this, we need to examine results of each dataset instead of looking at overall average results. On 11 of 109 datasets maximum cost solution of AFMDIH is even better than minimum cost solution of FMDIH, which makes approximately 10% of datasets. Interestingly 10 of these 11 datasets comprise more than 2000 city-nodes, since datasets are ordered from smallest to largest. This means that on large datasets it is more likely that each of the solutions from 30 simulations of AFMDIH turns out to be better than all the solutions of 30 simulations of FMDIH. This would mean we need to run AFMDIH only once to get better solution than all the 30 simulations of FMDIH on large datasets. Therefore, we can reduce computational time of AFDIH 30 times and get better solution than FMDIH in large datasets. This 30 times reduction in computational time overweighs the 8 times increase of computational time in large datasets.

Therefore, it can be concluded that AFMDIH not only provides better quality but also saves time if datasets of large size are involved, which is mostly the case in real world scenarios. Looking more deeply into the results reveals that average cost solution of AFMDIH is better than the minimum cost solution of FMDIH in 24 out of 109 datasets, which makes approximately 20% of datasets. Moreover, 16 of these 24 datasets happen to be amongst the largest 17 datasets of size more than 1700 cities. This is another evidence of excel in performance of AFMDIH with the increase in the problem size. Finally, we can also see maximum cost solution of AFMDIH is better than the average cost solution of FMDIH on 32 out of 109 datasets, which makes approximately 30% of datasets and most of these wins are concentrated at larger datasets i.e., 12 out of 14 largest datasets. This does also point towards what have been concluded earlier about performance excellence of AFMDIH on larger datasets.

CONCLUSION

This paper presents the concept of augmented tour construction heuristic, which is an advancement upon a simple tour construction heuristic. The method adds one additional step of partial tour deconstruction, which involves omitting a city node, in the standard set of steps of the tour construction heuristic. This additional step does not increase the theoretical complexity of the scheme; however, it does increase the computational time. The increase in computational time benefits solution quality in all average, best, and worst-case scenarios, along with the standard deviation. This observation is further supported by the results, indicating that the improvement in solution quality becomes more pronounced with larger datasets. In 30 simulations, the maximum cost solution obtained through the augmented heuristic outperformed the minimum cost solution achieved by the standard tour construction heuristic. This benefit outweighs the increase in computational cost, as a single run of the augmented heuristic can generate a better solution than multiple runs of the standard tour construction heuristic for larger datasets. This study represents a significant step toward addressing whether a sequence of city nodes exists that can be incrementally expanded into an optimal tour with the assistance of a tour construction heuristic. Further research is currently being conducted in this direction, exploring additional complementary steps within tour construction heuristics to approach near-optimal solutions. Given the consistently improved results with different heuristics, adding the iterative phase of partial tour deconstruction is the only explanation behind the improved results with the augmented heuristics.

REFERENCES

[1] R. Bellman, "Combinatorial Processes and Dynamic Programming," in *Bellman, R.; Hall, M. Jr. (eds.), Combinatorial Analysis, Proceedings of Symposia in Applied Mathematics*

10, American Mathematical Society, 1958, pp. 217–249. Available online: https://apps.dtic.mil/sti/tr/pdf/AD0606844.pdf

[2] R. Bellman, "Dynamic Programming Treatment of the Travelling Salesman Problem," *J. Assoc. Comput. Mach.*, vol. 9, pp. 61–63, 1962, doi: https://doi.org/10.1145/321105.321111.

[3] M. Held and R. M. Karp, "A Dynamic Programming Approach to Sequencing Problems," *J. Soc. Ind. Appl. Math.*, vol. 10, no. 1, pp. 196–210, 1962, doi: https://doi.org/10.1137/0110015, hdl:10338.dmlcz/103900

[4] P. T. E. Cusack, "The Seven Millennium Problems & AT Math," J. Math. Techniques Comput. Math., vol. 2, no. 2, pp. 89-103, 2023. Available online: https://www.opastpublishers.com/open-access-articles/the-seven-millenium-problems--at-math.pdf

[5] "Der Handlungsreisende – wie er sein soll und was er zu tun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiß zu sein – von einem alten Commis-Voyageur" (The Travelling Salesman — How He Must Be and What He Should Do in Order to Get Commissions and Be Sure of the Happy Success in His Business — by an Old Commis-Voyageur), 1832.

[6] K. Menger, "Das botenproblem," *Ergebnisse eines mathematischen kolloquiums*, vol. 2, no.4, pp 11-12, 1932.

[7] D. Dal and E. Celik, "Investigation of the impact of different versions of GCC on various metaheuristic-based solvers for traveling salesman problem," J. Supercomputing, pp. 1-47, 2023, doi: https://doi.org/10.1007/s11227-023-04109-0.

[8] M. Grötschel, M. Jünger, and G. Reinelt, "Optimal Control of Plotting and Drilling Machines: A Case Study," Math. Methods Oper. Res., vol. 35, no. 1, pp. 61-84, Jan. 1991. doi: https://doi.org/10.1007/BF01453944.

[9] R. D. Plante, T. J. Lowe, and R. Chandrasekaran, "The Product Matrix Traveling Salesman Problem: An Application and Solution Heuristics," Oper. Res., vol. 35, pp. 772-783, 1987. doi: https://doi.org/10.1287/opre.35.5.772.

[10] R. E. Bland and D. E. Shallcross, "Large traveling salesman problem arising from experiments in X-ray crystallography: a preliminary report on computation," Oper. Res. Lett., vol. 8, no. 3, pp. 125-128, 1989. doi: https://doi.org/10.1016/0167-6377(89)90051-3.

[11] J. K. Lenstra and A. H. G. Rinnooy Kan, "Some Simple Applications of the Travelling Salesman Problem," BW 38/74, Stichting Mathematisch Centrum, Amsterdam, 1974. Available online: https://ir.cwi.nl/pub/21760/21760A.pdf

[12] H. D. Ratliff and A. S. Rosenthal, "Order-Picking in a Rectangular Warehouse: A Solvable Case for the Travelling Salesman Problem," Oper. Res., vol. 31, pp. 507-521, 1983. doi: https://doi.org/10.1287/opre.31.3.507.

[13] L. Gouveia, A. Paias, and M. Ponte, "The travelling salesman problem with positional consistency constraints: An Application to healthcare services," Eur. J. Oper. Res., vol. 308, no. 3, pp. 960-989, 2023. doi: https://doi.org/10.1016/j.ejor.2021.06.013.

[14] A. Maya-López, A. Martínez-Ballesté, and F. Casino, "A compression strategy for an efficient TSP-based microaggregation," Expert Syst. Appl., vol. 213, 118980, 2023. doi: https://doi.org/10.1016/j.eswa.2020.118980.

[15] C. Cariou, L. Moiroux-Arvis, F. Pinet, and J. P. Chanet, "Evolutionary Algorithm with Geometrical Heuristics for Solving the Close Enough Traveling Salesman Problem: Application to the Trajectory Planning of an Unmanned Aerial Vehicle," Algorithms, vol. 16, no. 1, 44, 2023. doi: https://doi.org/10.3390/a16010044.

[16] Y. Wang and N. Wang, "Moving-target travelling salesman problem for a helicopter patrolling suspicious boats in antipiracy escort operations," Expert Syst. Appl., vol. 213, p. 118986, 2023. doi: https://doi.org/10.1016/j.eswa.2020.118986.

[17] A. Di Placido, C. Archetti, and C. Cerrone, "A genetic algorithm for the close-enough traveling salesman problem with application to solar panels diagnostic reconnaissance,"

10.12928/ijio.v4i2.7875

Comput. Oper. Res., vol. 145, p. 105831, 2022, doi: https://doi.org/10.1016/j.cor.2021.105831.
[18] Z. Zhang and J. Yang, "A discrete cuckoo search algorithm for traveling salesman problem and its application in cutting path optimization," Comput. Ind. Eng., vol. 169, p. 108157, 2022. doi: https://doi.org/10.1016/j.cie.2022.108157.
[19] A. Gharehgozli, C. Xu, and W. Zhang, "High multiplicity asymmetric traveling salesman problem with feedback vertex set and its application to storage/retrieval system," Eur. J. Oper. Res., vol. 289, no. 2, pp. 495-507, 2021. doi: https://doi.org/10.1016/j.ejor.2020.08.041.
[20] P. Baniasadi, M. Foumani, K. Smith-Miles, and V. Ejov, "A transformation technique for the clustered generalized traveling salesman problem with applications to logistics," Eur. J. Oper. Res., vol. 285, no. 2, pp. 444-457, 2020. doi: https://doi.org/10.1016/j.ejor.2020.02.024.
[21] S. Mirjalili, J. Song Dong, and A. Lewis, "Ant colony optimizer: theory, literature review, and application in AUV path planning," In: Mirjalili, S., Song Dong, J., Lewis, A. (eds) *Nature-Inspired Optimizers. Studies in Computational Intelligence*, vol 811. Springer, Cham, pp. 7-21, 2020, doi: https://doi.org/10.1007/978-3-030-12127-3_2
[22] J. Wu, L. Zhou, Z. Du, and Y. Lv, "Mixed steepest descent algorithm for the traveling salesman problem and application in air logistics," Transp. Res. Part E: Logist. Transp. Rev., vol. 126, pp. 87-102, 2019. doi: https://doi.org/10.1016/j.tre.2019.03.003.
[23] S. Baressi Šegota, I. Lorencin, K. Ohkura, and Z. Car, "On the traveling salesman problem in nautical environments: an evolutionary computing approach to optimization of tourist route paths in Medulin, Croatia," Pomorski zbornik, vol. 57, no. 1, pp. 71-87, 2019. doi: https://doi.org/10.18048/2019.57.05.
[24] A. S. Elgesem, E. S. Skogen, X. Wang, and K. Fagerholt, "A traveling salesman problem with pickups and deliveries and stochastic travel times: An application from chemical shipping," Eur. J. Oper. Res., vol. 269, no. 3, pp. 844-859, 2018. doi: https://doi.org/10.1016/j.ejor.2018.02.041.
[25] U. Hacizade and I. Kaya, "GA based traveling salesman problem solution and its application to transport routes optimization," IFAC-Pap., vol. 51, no. 30, pp. 620-625, 2018. doi: https://doi.org/10.1016/j.ifacol.2018.11.106.
[26] Y. Luo, B. Golden, S. Poikonen, and R. Zhang, "A fresh look at the Traveling Salesman Problem with a Center," Comput. Oper. Res., vol. 143, p. 105748, 2022. doi: https://doi.org/10.1016/j.cor.2022.105748.
[27] B. Manthey and J. van Rhijn, "Approximation Ineffectiveness of a Tour-Untangling Heuristic," arXiv preprint arXiv:2302.11264, 2023, doi: https://doi.org/10.48550/arXiv.2302.11264
[28] J. Perttunen, "On the significance of the initial solution in travelling salesman heuristics," J. Oper. Res. Soc., vol. 45, no. 10, pp. 1131-1140, 1994. doi: https://doi.org/10.1057/jors.1994.150.
[29] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," Oper. Res., vol. 21, no. 2, pp. 498-516, 1973. doi: https://doi.org/10.1287/opre.21.2.498.
[30] D. J. Rosenkrantz, R. E. Stearns, and I. Lewis, "An analysis of several heuristics for the traveling salesman problem," SIAM J. Comput., vol. 6, no. 3, pp. 563-581, 1977. doi: https://doi.org/10.1137/0206041.
[31] D. S. Johnson and L. A. McGeoch, "Experimental analysis of heuristics for the STSP," in *The Traveling Salesman Problem and Its Variations*, G. Gutin and A. P. Punnen, Eds., vol. 12 of Combinatorial Optimization, chapter 9, pp. 369-443, Kluwer Academic Publishers, 2002, doi: https://doi.org/10.1007/0-306-48213-4_9
[32] J. J. Bentley, "Fast algorithms for the geometric travelling salesman problem," ORSA J. Comput., vol. 4, no. 4, pp. 387-411, 1992. doi: https://doi.org/10.1287/ijoc.4.4.387.
[33] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: a case study in local optimization," in *Local Search in Combinatorial Optimization*, vol. 1, pp. 215-310,

John Wiley & Sons, 1997, Available online: http://techfak.uni-bielefeld.de/ags/wbski/lehre/digiSA/WS0304/IntAlg/TSPchapter.pdf.
[34] M. Goutham, M. Menon, S. Garrow, and S. Stockar, "A Convex Hull Cheapest Insertion Heuristic for the Non-Euclidean and Precedence Constrained TSPs," arXiv preprint arXiv:2302.06582, 2023, doi: https://doi.org/10.48550/arXiv.2302.06582
[35] K. N. Qureshi, F. Bashir, and A. H. Abdullah, "Distance and signal quality aware next hop selection routing protocol for vehicular ad hoc networks," Neural Comput. Appl., vol. 32, pp. 2351-2364, 2020. doi: https://doi.org/10.1007/s00521-018-3520-2.
[36] J. J. Bentley, "Fast algorithms for the geometric travelling salesman problem," ORSA J. Comput., vol. 4, no. 4, pp. 387-411, 1992. doi: https://doi.org/10.1287/ijoc.4.4.387.
[37] Z. Ursani and D. W. Corne, "Introducing Complexity Curtailing Techniques for the Tour Construction Heuristics for the Travelling Salesperson Problem," J. Optim., vol. 2016, Article ID 4786268, 15 pages, 2016. doi: https://doi.org/10.1155/2016/4786268.
[38] R. Hassin and A. Keinan, "Greedy heuristics with regret, with application to the cheapest insertion algorithm for the TSP," Oper. Res. Lett., vol. 36, no. 2, pp. 243-246, 2008. doi: https://doi.org/10.1016/j.orl.2007.09.002.
[39] J. Renaud, F. F. Boctor, and J. Ouenniche, "A heuristic for the pickup and delivery traveling salesman problem," Comput. Oper. Res., vol. 27, no. 9, pp. 905-916, 2000. doi: https://doi.org/10.1016/S0305-0548(99)00066-3.
[40] MP-TESTDATA - The TSPLIB Symmetric Traveling Salesman Problem Instances http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html (Accessed 20/03/2020).

10.12928/ijio.v4i2.7875