A Multi-Start Iterated Local Search Algorithm for the Bottleneck TSP

Project Review

Viknesh Rajaramon - COE18B060

under the supervision of

Dr. Venkatesh Pandiri

Department of Computer Science & Engineering
Indian Institute of Information Technology Design and Manufacturing, Chennai, Tamilnadu-600127, India

May 12, 2022

Seminar Outline

Motivation

- The Bottleneck Travelling Salesman Problem (BTSP) is a variant of the well-known Travelling Salesman Problem (TSP) where the objective is to find a Hamiltonian circuit that minimizes the maximum edge cost among its constituent edges.
- The BTSP finds application in the area of workforce planning and in minimizing make span in a two-machine flow shop with no-wait-in-process.
- Exact solution to the BTSP is NP-Hard. Hence, we try to obtain optimal solution to the BTSP by heuristic approaches.

Problem Statement

- Given an undirected edge-weighted complete graph $G = (V, E)$, where $V = \{1, 2, ..., n\}$ is the set of nodes, $E = \{(i, j) | i, j \in V\}$ is the set of edges, and a length or cost d_{ij} is associated with each edge $(i, j) \in E$, the BTSP seeks a Hamiltonian cycle that minimizes the length of the edge which is having the maximum length among its constituent edges.

Selected Literature Survey I

R. Ramakrishnan, Prabha Sharma, Abraham P. Punnen. 2009

An efficient heuristic algorithm for the bottleneck traveling salesman problem, Opsearch 46, 275-288.

G. S. Manku, 1996

A linear time algorithm for the bottleneck biconnected spanning subgraph problem, Information Processing Letters, 59: 1 7.

Z. H. Ahmed, 2010

A hybrid sequential constructive sampling algorithm for the bottleneck traveling salesman problem, IJCIR, 475-484.

Z. H. Ahmed, 2000

A sequential constructive sampling and related approaches to combinatorial optimization, Ph.D. Thesis, Tezpur University, Assam, India.

Z. H. Ahmed, 2013

A Hybrid Genetic Algorithm for the Bottleneck Travelling Salesman Problem, ACM Transactions on Embedded Computing Systems 12 (2013) Art. No. 9.

Iterated Local Search (ILS) I

- Iterated Local Search (ILS) is a single solution based meta-heuristic which iteratively improves the solution quality.
- The ILS mainly consists of four components: initial solution generation, local search, acceptance criteria and perturbation procedure.
- Starting from an initial solution, an iterative process ensues. During each iteration, first a local search algorithm is applied on the current solution to find a locally optimal solution.
- Depending on the acceptance criteria, this newly obtained locally optimal solution may replace the current solution.
- In order to escape from that locally optimal solution, a perturbation procedure is applied on the current solution to get a perturbed solution.

Iterated Local Search (ILS) II

- In essence, the local search procedures do the job of exploitation, whereas the perturbation procedure does the job of exploration.

Algorithm 1: Pseudo-code for basic ILS

Input: Set of parameters for ILS
Output: Best solution found
$T \leftarrow Initial\_Solution()$;
while *Termination condition not satisfied* **do**
  $T_1 \leftarrow Local\_Search(S)$;
  $T \leftarrow Acceptance\_Criteria(T, T_1, history)$;
  $T \leftarrow Perturbation\_Procedure(T)$;
return *best*;

Multi-Start Iterated Local Search (MS-ILS)

- The proposed multi-start iterated local search (MS-ILS) for the BTSP is an extension of ILS and restarts the ILS multiple times, each time starting with a new solution generated by our initial solution generation procedure.
- The motivation for choosing the multi-start mechanism is due to the fact that unproductive iterations may consume more time.

MS-ILS Components I

Solution Encoding and Fitness

- Within the MS-ILS, a solution is encoded by a linear permutation of the nodes where the first node always occupies the first position. By restricting the first node to the first position, the redundancy is removed in representation.
- NOTE : None of the MSILS components can modify the position of the first node.
- As BTSP is a minimization problem, the fitness function is defined as the multiplicative inverse of the objective function (considered as the maximum edge cost).

Initial Solution Generation

- The initial solution generation procedure starts by selecting two nodes uniformly at random and then an iterative process ensues.
- During each iteration, a node is selected uniformly at random and inserted in between the nodes associated with the edge of maximum cost.
- This procedure is repeated until all the nodes are inserted into the tour.

MS-ILS Components II

Local Search

- Consists of two heuristics h_1 and h_2 each of which handle two cases.
- In Case 1, there exists a single maximum cost edge. The heuristics try to minimize the maximum edge cost. In Case 2, there can be multiple edges with maximum cost. The heuristics try to reduce the number of maximum cost edges by as much as possible.

Heuristic h_1 : Insertion between the nodes of maximum cost edge

- **Case 1** - Inserts a node between the nodes of maximum cost edge. Every node is tried for insertion and the best among all the resulting solutions is accepted.
- **Case 2** - Each maximum cost edge is considered one-by-one and Case 1 is applied. The best among all the resulting solutions is accepted. If the number of maximum cost edges get reduced then this heuristic terminates and h_2 starts.

Heuristic h_2 : Maximum cost edge centric 2-opt move

- Proposed heuristic is a modified version of 2-opt move, in which the maximum cost edge is always one of the two edges to be removed.
- **Case 1** - Every other edge is tried with the maximum cost edge for removal and two new edges are inserted at their place to minimize the maximum edge cost. The move that yields the maximum decrease in maximum edge cost is accepted.
- **Case 2** - Same manner as in h_1 and control is passed to h_1 as soon as the number of maximum cost edges got reduced.

MS-ILS Components III

Acceptance Criteria

- Our acceptance criteria compare the quality (fitness) of the solution generated by the local search procedure with the solution before applying this procedure.
- An improved fitness solution is accepted all the times. The equal fitness solution is accepted in cases where there are multiple edges with the maximum edge cost and there is a decrease in the number of edges having that cost.
- Upon failing all the above mentioned cases, the perturbation procedure is applied and it may return a better or worse fitness solution. The search process starts from this newly returned solution.

Perturbation Procedure

- The goal of the perturbation procedure is to escape from the present locally optimum solution by perturbing it, and providing a new starting solution to the local search to move the search to unexplored regions in the search space.
- As part of this procedure, destroy and repair mechanism is used. Each node from the tour is removed with a probability P_{pert}. All such removed nodes are added back to the tour in an iterative manner in the same manner as followed in the initial solution generation procedure.

Pseudo-codes

Algorithm 2: Pseudo-code for perturbing a solution

Input: A solution T

Output: A perturbed solution T_1

```
Function Perturbation_Procedure(T):
    foreach node c in tour of T do
        Generate a random number 0 ≤ p ≤ 1
        if p < P_pert then
            Add c to a set of unassigned nodes
        else
            Copy c to tour of T_1
    foreach node c in the set of unassigned nodes
     in some random order do
        Follow the procedure described in
         Initial_Solution() to insert c into tour of T_1
return T_1;
```

Algorithm 3: Pseudo-code for the proposed MS-ILS approach for the BTSP

Input: Set of parameters for the MS-ILS and a BTSP instance

Output: Best solution found

```
F(best) = −∞
for st = 1 to N_rst do
    T ← Initial_Solution();
    while Termination condition not satisfied do
        /* Apply heuristic h_1 */
        T_1 ← Apply_heuristic_h_1(T);
        if F(T_1) > F(T) then
            T ← T_1;
        else if F(T_1) > F(T) then
            if no. of edges with F(T) decreased then
                T ← T_1;
        /* Apply heuristic h_2 */
        T_1 ← Apply_heuristic_h_2(T);
        if F(T_1) > F(T) then
            T ← T_1;
        else if F(T_1) > F(T) then
            if no. of edges with F(T) decreased then
                T ← T_1;
        /* Dealing with the best solution and local
            optimum solution */
        if F(T) > F(best) then
            best ← T;
        else if F(T_1) < F(T) then
            T_1 ← Perturbation_Procedure();
return best;
```

Dataset Acquisition I

- TSPLIB instances each with different number of nodes are downloaded from the standard TSPLIB library.
- Each instance consists of data with different edge types (GEO, EUC_2D, etc). Each of these instances is converted to a $n \times n$ distance matrix where the distance between any two nodes is an integer.
- The diagonal elements in the $n \times n$ distance matrix is represented as ∞.

Dataset Acquisition II

```
NAME: burma14
TYPE: TSP
COMMENT: 14-Staedte in Burma (Zaw Win)
DIMENSION: 14
EDGE_WEIGHT_TYPE: GEO
EDGE_WEIGHT_FORMAT: FUNCTION
DISPLAY_DATA_TYPE: COORD_DISPLAY
NODE_COORD_SECTION
   1  16.47       96.10
   2  16.47       94.44
   3  20.09       92.54
   4  22.39       93.37
   5  25.23       97.24
   6  22.00       96.05
   7  20.47       97.02
   8  17.20       96.29
   9  16.30       97.38
  10  14.05       98.12
  11  16.53       97.38
  12  21.52       95.59
  13  19.41       97.13
  14  20.09       94.55
EOF
```

Figure: **burma14** instance from the standard TSPLIB library

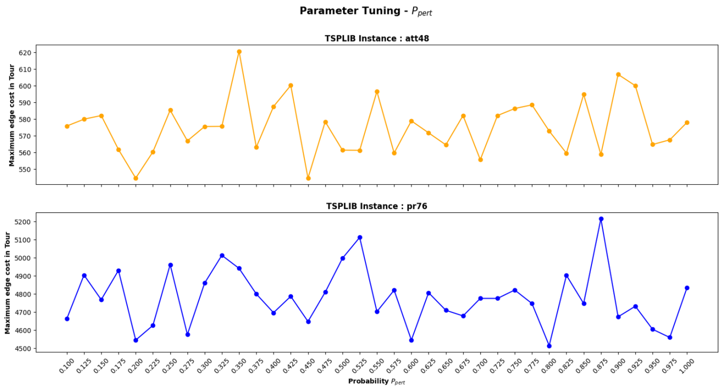
Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	∞	153	510	706	966	581	455	70	160	372	157	567	342	398
2	153	∞	422	664	997	598	507	197	311	479	310	581	417	376
3	510	422	∞	289	744	390	437	491	645	880	618	374	455	211
4	706	664	289	∞	491	265	410	664	804	1070	768	259	455	310
5	966	997	744	491	∞	400	514	902	990	1261	947	418	499	636
6	581	598	390	265	400	∞	168	522	634	910	593	19	635	239
7	455	507	437	410	514	168	∞	389	482	757	439	163	284	232
8	70	197	491	664	902	522	389	∞	154	406	133	508	124	355
9	160	311	645	804	990	634	482	154	∞	276	43	623	273	498
10	372	479	880	1070	1261	910	757	406	276	∞	318	898	358	761
11	157	310	618	768	947	593	439	133	43	318	∞	582	633	464
12	567	581	374	259	418	19	163	508	623	898	582	∞	315	221
13	342	417	455	499	635	284	124	273	358	633	315	275	∞	247
14	398	376	211	310	636	239	232	355	498	761	464	221	247	∞

Figure: Distance matrix for **burma14** instance

Parameter Tuning

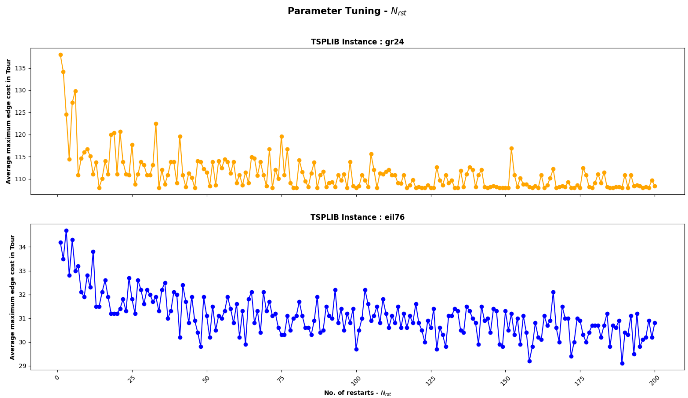
- Three parameters - P_{pert}, N_{rst} and $RUNS$ need to be optimized to obtain better results.
- P_{pert} denotes the probability with which a node is removed during the perturbation procedure.
- N_{rst} represents the number of times the algorithm is restarted after being stuck in local optimal solution.
- $RUNS$ denotes the number of times the algorithm is run on the same instance, each time with a random seed value.

Parameter Tuning - P_{pert}



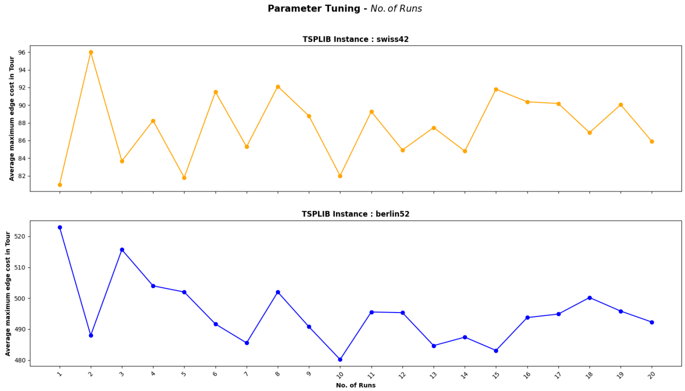
Both the TSPLIB instances **att48** and **pr76** obtain a average maximum edge cost of 544.40 and 4543.60 respectively when $P_{pert} = 0.2$.

Parameter Tuning - N_{rst}



Both the TSPLIB instances **gr24** and **eil76** obtain a average maximum edge cost of 108.00 and 29.70 respectively when $N_{rst} = 100$.

Parameter Tuning - *RUNS*



Both the TSPLIB instances **swiss42** and **berlin52** obtain a average maximum edge cost of 82.00 and 480.20 respectively when $RUNS = 10$.

Results on few Test Instances

- The proposed MS-ILS approach is executed on each test instance 10 times independently, each time starting with a different random seed.
- For all the test instances, the following parameters are used - MS-ILS is restarted 100 times, i.e., N_{rst} =100, $P_{pert} = 0.2$.
- The MS-ILS terminates when there is no improvement in the solution continuously for 100 iterations.
- Below table shows the comparison between the proposed MS-ILS algorithm and already existing algorithms. We can clearly see that the proposed MS-ILS algorithm finds the optimal solution (in average) for all the 15 instances whereas HGA, HSCS and BST algorithms find the solution for 5, 3 and 10 instances respectively.
- On the basis of both solution quality and computational time, the proposed MS-ILS algorithm is found to be better than BST, HSCS, and HGA algorithms.

Results I

Table: Results of Different Algorithms for some Standard TSPLIB Instances

Instance	n	BST		HSCS		HGA		MS-ILS($h_1 + h_2$)	
		Average	Time	Average	Time	Average	Time	Average	Time
burma14	14	**418.00**	742.38	422.18	60.57	**418.00**	61.80	**418.00**	0.02
ulysses16	16	**1504.00**	834.15	**1504.00**	68.06	**1504.00**	69.44	**1504.00**	0.07
gr17	17	**282.00**	727.00	**282.00**	59.31	**282.00**	60.52	**282.00**	0.03
gr21	21	**355.00**	826.46	**355.00**	67.43	**355.00**	68.80	**355.00**	0.03
ulysses22	22	**1504.00**	938.42	1519.04	76.56	**1504.00**	78.12	**1504.00**	0.13
fri26	26	**93.00**	533.36	93.50	43.51	93.50	44.40	**93.00**	0.04
brazil58	58	**2149.00**	1264.68	2508.54	103.18	2483.70	105.28	**2149.00**	0.13
gr96	96	3491.00	2931.54	4098.90	239.17	4098.90	244.04	**2807.00**	0.36
pr107	107	7053.00	3694.10	7387.40	301.39	7387.40	307.52	**7050.00**	0.40
bier127	127	**7486.00**	3765.21	7957.80	307.19	7957.80	313.44	**7486.00**	0.43
gr137	137	4282.00	4409.57	5153.63	359.76	5102.60	367.08	**2132.00**	0.86
brg180	180	9000.00	5997.15	9000.00	489.29	9000.00	499.24	**3500.00**	0.96
d198	198	1511.00	9824.34	1712.40	801.53	1712.40	817.84	**1380.00**	1.06
gr202	202	**2230.00**	8996.92	2393.70	734.03	2393.70	748.96	**2230.00**	0.98
d493	493	**2008.00**	81359.08	2045.25	6637.80	2025.00	6772.84	**2008.00**	4.82
Overall		3188.00	8456.29	3423.63	689.92	3415.08	703.95	**2536.62**	0.69
NBV		10		3		5		**15**	

Results II

Table: Results of MS-ILS(h_1), MS-ILS(h_2), MS-ILS($h_1 + h_2$)

Instance	n	MS-ILS(h_1)				MS-ILS(h_2)				MS-ILS($h_1 + h_2$)			
		Best	Worst	Average	Time	Best	Worst	Average	Time	Best	Worst	Average	Time
gr21	21	**355.00**	**355.00**	**355.00**	0.03	**355.00**	**355.00**	**355.00**	0.02	**355.00**	**355.00**	**355.00**	0.03
eil76	76	30.00	34.00	33.20	0.21	30.00	34.00	32.40	0.14	**27.00**	**33.00**	**31.10**	0.26
gr120	120	365.00	403.00	385.30	0.46	313.00	417.00	374.90	0.30	**220.00**	**370.00**	**277.20**	0.52
tsp225	225	178.00	193.00	186.40	1.16	185.00	196.00	190.00	0.92	**169.00**	**192.00**	**182.00**	1.39
d493	493	**2008.00**	**2008.00**	**2008.00**	3.70	**2008.00**	2254.00	2055.70	3.19	**2008.00**	**2008.00**	**2008.00**	4.82
att532	532	1027.00	1088.00	1045.70	5.34	1009.00	1060.00	1035.00	4.46	**742.00**	**1008.00**	**890.50**	8.33
u724	724	1169.00	1221.00	1195.70	7.89	1145.00	**1213.00**	1188.50	7.18	**1004.00**	1220.00	**1133.00**	17.20
rat783	783	**216.00**	228.00	222.40	9.87	219.00	227.00	221.70	8.94	217.00	**224.00**	**220.00**	18.73
nrw1379	1379	1035.00	1071.00	1059.40	32.07	1045.00	**1062.00**	1053.20	35.87	**1028.00**	1070.00	**1049.10**	66.73
fl1577	1577	970.00	988.00	978.70	49.97	964.00	**986.00**	978.80	42.47	**958.00**	988.00	**972.40**	98.71
vm1748	1748	8936.00	9288.00	9185.50	64.27	8996.00	9237.00	9138.90	59.71	**6820.00**	**7258.00**	**7000.30**	130.57
rl1889	1889	7883.00	8064.00	7997.40	68.41	7885.00	8140.00	8024.40	75.94	**6797.00**	**7868.00**	**7422.80**	170.28
Overall		2014.33	2078.42	2054.39	20.28	2012.83	2098.42	2054.04	19.93	**1695.42**	**1882.83**	**1795.12**	43.13
NBV		3	2	2		2	4	1		**11**	**9**	**12**	

- Out of 82 instances, the MS-ILS($h_1 + h_2$) got the best values for 79 and 80 instances for the best and average objective function values respectively. The MS-ILS(h_2) got the best values for 25 and 13 instances for the best and average objective function values respectively. The MS-ILS(h_1) got the best values for 19 and 8 instances for the best and average objective function values respectively.

Wilcoxon Signed Rank Test

	MS-ILS($h_1 + h_2$)							MS-ILS(h_2)					
	NWT / Total	W^+	W^-	Z	Z_c	*Significant*		*NWT / Total*	W^+	W^-	Z	Z_c	*Significant*
MS-ILS(h_2)	71/82	2553	3	-7.306	-2.576	Yes	**MS-ILS**(h_1)	74/82	2355	0	-5.212	-2.576	Yes
MS-ILS(h_1)	74/82	2775	0	-7.475	-2.576	Yes							

- Two tailed Wilcoxon signed rank test has been used to check whether the performances of the three MS-ILS variants differ significantly. The significance criteria were set to 1% (i.e., $p - value \leq 0.01$).
- W^+ represents the sum of ranks for cases where the top of the table approach (MS-ILS($h_1 + h_2$)/MS-ILS(h_2)) outperforms its competitor on the left side of the table.
- W^- represents the sum of ranks for cases where the top of the table approach (MS-ILS($h_1 + h_2$)/MS-ILS(h_2)) under performs its competitor on the left side of the table.
- Since there are more than thirty instances ($NWT > 30$), the test statistic Z is utilized.
- If $Z \leq Z_c$, the performance of the two MS-ILS variations under consideration differs significantly; otherwise, the difference is insignificant.
- The results of all three MS-ILS variants (MS-ILS(h_1), MS-ILS(h_2), and MS-ILS($h_1 + h_2$) are statistically significant from each other, as shown in the above table.

Time Complexity - Heuristic h_1 I

- A tour on k nodes is denoted by $H^k=\{c_1, c_2, c_3, \ldots, c_k\}$.
- Choose a random node r from the remaining $n - k$ nodes and place it between the nodes with the highest edge cost in H^k.
- Replace H^k with the new tour H^{k+1}, and repeat until a tour H^n is achieved. This method takes $O(n^3)$ time to implement in a simple way.
- The time complexity can be decreased to $O(n^2)$ by efficiently storing the maximum ans second maximum edge cost and retrieving it in $O(1)$ time.
- The average time complexity of the heuristic h_1 may thus be validated as $O(n^2)$.

Time Complexity - Heuristic h_1 II

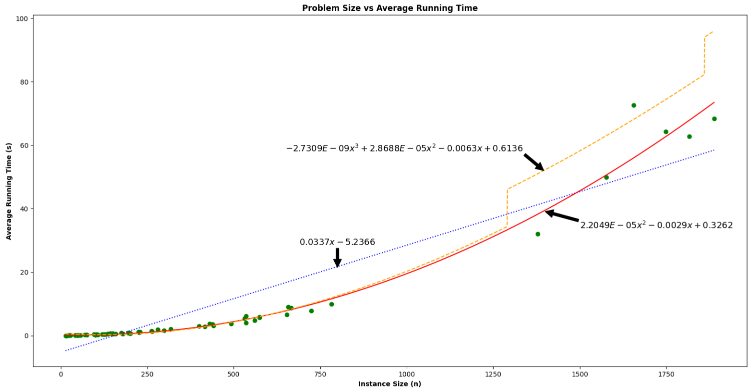


Figure: Problem Size vs Average Running Time for Heuristic h_1

- Above figure shows that the average time complexity from the computational results obtained is also $O(n^2)$.

Time Complexity - Heuristic h_2 I

- In general, a single two-opt move incurs $O(n^2)$ cost in the worst case.
- In our algorithm, since we choose the one of the edges to be the maximum edge in the tour, we need to select one edge from the remaining $(n-1)$ edges, which is $O(n)$.
- After the two-opt move, updating the edge with maximum edge cost is $O(n)$.
- Therefore, the average time complexity of a single two-opt move is reduced $O(n)$. Since the two-opt move is done n-times for n edges, the overall average time complexity of heuristic h_2 becomes $n * O(n) = O(n^2)$.

Time Complexity - Heuristic h_2 II

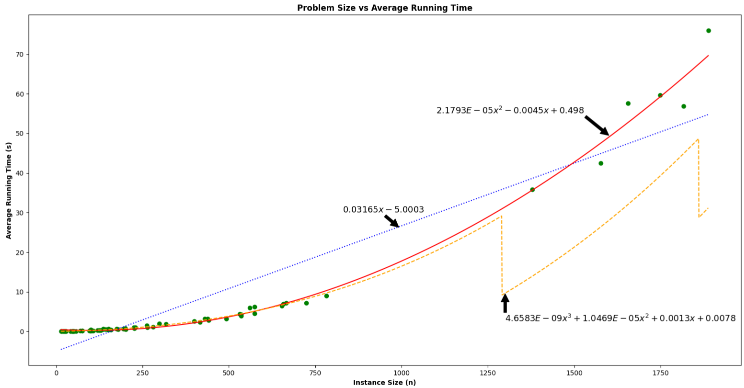
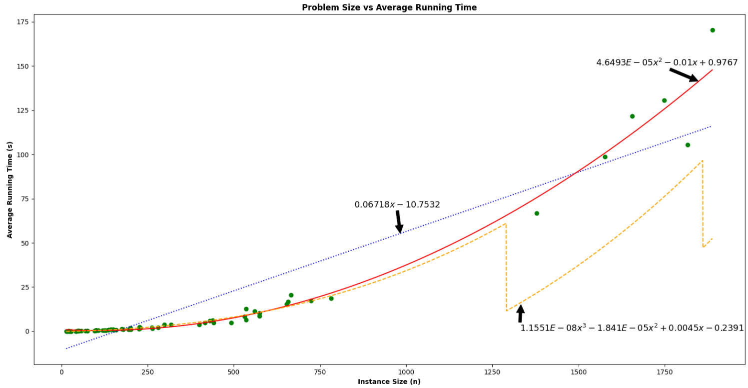


Figure: Problem Size vs Average Running Time for Heuristic h_2

- Above figure shows that the average time complexity from the computational results obtained is also $O(n^2)$.

Time Complexity - Heuristic $h_1 + h_2$



- Above figure shows how closely the quadratic fit approximates the running time. This is consistent with the known experimental findings for the LK-heuristic on the average complexity of $O(n^{2.2})$.

Conclusion and Future Scope

- On standard TSPLIB instances, the proposed heuristic method (viz. MS-ILS($h_1 + h_2$)) performed well when compared with two other heuristics (viz. MS-ILS(h_1) and MS-ILS(h_2)). There was no significant difference in the performance of MS-ILS(h_1) and MS-ILS(h_2).
- Compared to other existing algorithms, MS-ILS($h_1 + h_2$) proved to be the best in terms of both solution and computational time.
- As a future work, we intend to build a population-based meta-heuristic method for the BTSP by combining it with MS-ILS components to enhance the result.

References I

[1] Paul C Gilmore and Ralph E Gomory. "Sequencing a one state-variable machine: A solvable case of the traveling salesman problem". In: *Operations research* 12.5 (1964), pp. 655–679.

[2] E Gabovic, A Ciz, and A Jalas. "The bottleneck traveling salesman problem". In: *Trudy Vy cisl. Centra Tartu. Gos. Univ* 22 (1971), pp. 3–24.

[3] George L Vairaktarakis. "On Gilmore–Gomory's open question for the bottleneck TSP". In: *Operations Research Letters* 31.6 (2003), pp. 483–491.

[4] Ming-Yang Kao and Manan Sanghi. "An approximation algorithm for a bottleneck traveling salesman problem". In: *Journal of Discrete Algorithms* 7.3 (2009), pp. 315–326.

[5] Ravi Ramakrishnan, Prabha Sharma, and Abraham P Punnen. "An efficient heuristic algorithm for the bottleneck traveling salesman problem". In: *Opsearch* 46.3 (2009), pp. 275–288.

[6] Gurmeet Singh Manku. "A linear time algorithm for the Bottleneck Biconnected Spanning Subgraph problem". In: *Information processing letters* 59.1 (1996), pp. 1–7.

[7] Zakir H Ahmed. "A hybrid sequential constructive sampling algorithm for the bottleneck traveling salesman problem". In: *International Journal of Computational Intelligence Research* 6.3 (2010), pp. 475–484.

References II

[8] Zakir Hussain Ahmed. "A sequential constructive sampling and related approaches to combinatorial optimization". PhD thesis. Tezpur University, Assam, India, 2000.

[9] Zakir Hussain Ahmed. "A hybrid genetic algorithm for the bottleneck traveling salesman problem". In: *ACM Transactions on Embedded Computing Systems (TECS)* 12.1 (2013), pp. 1–10.

[10] Helena Ramalhinho Lourenço, Olivier C Martin, and Thomas Stützle. "Iterated local search: Framework and applications". In: *Handbook of metaheuristics*. Springer, 2019, pp. 129–168.

[11] Pandiri Venkatesh, Gaurav Srivastava, and Alok Singh. "A multi-start iterated local search algorithm with variable degree of perturbation for the covering salesman problem". In: *Harmony Search and Nature Inspired Optimization Algorithms*. Springer, 2019, pp. 279–292.

[12] Frank Wilcoxon, SK Katti, and Roberta A Wilcox. *Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test*. American Cyanamid Company, 1963.

[13] Santosh N Kabadi and Abraham P Punnen. "The Bottleneck TSP". In: *The traveling salesman problem and its variations*. Springer, 2007, pp. 697–736.

References III

[14] R Gary Parker and Ronald L Rardin. "Guaranteed performance heuristics for the bottleneck travelling salesman problem". In: *Operations Research Letters* 2.6 (1984), pp. 269–272.

[15] Robert S Garfinkel and KC Gilbert. "The bottleneck traveling salesman problem: Algorithms and probabilistic analysis". In: *Journal of the ACM (JACM)* 25.3 (1978), pp. 435–448.

[16] Dorit S Hochbaum and David B Shmoys. "A unified approach to approximation algorithms for bottleneck problems". In: *Journal of the ACM (JACM)* 33.3 (1986), pp. 533–550.

[17] Pandiri Venkatesh, Alok Singh, and Rammohan Mallipeddi. "A Multi-Start Iterated Local Search Algorithm for the Maximum Scatter Traveling Salesman Problem". In: *2019 IEEE Congress on Evolutionary Computation (CEC)*. 2019, pp. 1390–1397. DOI: 10.1109/CEC.2019.8790018.

Thank You