International Journal of Science and Research (IJSR)
ISSN: 2319-7064
ResearchGate Impact Factor (2018): 0.28 | SJIF (2019): 7.583

Variants of *2*-opt Approach for the Generalized Traveling Salesman Problem

Luu Van Thanh

[1]International University, Vietnam National University – HCM City, School of Industrial Engineering and Management, Quarter 6, Linh Trung Ward, Thu Duc District, HCMC, Vietnam

Abstract: ***Generalized Traveling Salesman Problem (GTSP) is a well-known NP-hard problem. In a symmetric GTSP, the salesman must pass through a number of predefined subsets of customers, determining the order in which the subsets should be visited, and visiting exactly one customer in each subset while minimizing the sum of traveling costs of a completed undirected graph. This paper introduces a metaheuristic approach for solving this problem. The proposed algorithm is composed of two stages: (1) the constructive algorithm using the nearest neighbor heuristics (NN); and (2) the local improved algorithms consisting of combination of the well-known 2-opt search (2-opt classic), the adaptation of 2-opt with the NN (2-opt-NN), and the shortest path approach using Dijkstra's algorithm (2-opt-SP). The computational results on thirty-six TSPLIB problems with up to 442 nodes are presented wherein the problems up to 300 nodes have been solved with computational time shorter than previous results cited in the literature.***

Keywords: Combinatorial optimization, generalized traveling salesman problem, heuristics, local search.

1. Introduction

Generalized Traveling Salesman Problem (GTSP) is a generalization of the well-known traveling salesman problem (TSP) wherein the number of customers is divided into clusters. The salesman determines the order of the clusters to be visited, and chooses the customer to be visited in each cluster.

The *symmetric* GTSP can be defined as follows. Given an *undirected* graph $G = (N,E)$ where the set of nodes $N = \{1, ..., n\}$ is partitioned into m *mutually exclusive and exhaustive subsets* (clusters), C_k (for $i \neq j$, $C_i \cap C_j = \emptyset$; $N = C_1 \cup C_2 \cup \ ... \cup C_k$; $k = 1, ..., m\}$ and to each edge $(i, j) \in E$, $E = \{(i, j): i, j \in N\}$ is associated a distance (or cost) c_{ij}, such that $c_{ij} = c_{ji}$, determine the minimum-length (cost) m-edge cycle on G which includes *exactly one* node from each node-subset. The process presented is for the *symmetric* GTSP (*s*GTSP), in which the travel distances are symmetrical ($c_{ij} = c_{ji}$) and satisfy the triangle inequality. Discussion of *asymmetric* GTSP (*a*GTSP) is not included in this paper.

The GTSP is a NP-hard problem since it reduces to the TSP when each cluster contains exactly one node.

The GTSP is a well-known combinatorial optimization problem with a host of applications in planning & scheduling, material flow, warehousing, sequencing, post box and air-port selection, etc., [6].

2. Materials - Methods

In 1969, Srivastava *et al.*, and Henry-Labordere addressed *s*GTSP and *a*GTSP respectively and proposed a dynamic programming model for their solutions. Over ten years later, Laporte *et al.*, solved these problems by an integer programming. Fischetti *et al.*, (1997) developed a branch and cut algorithm to solve the symmetric instances for optimality. For large problems, there is a need for good heuristic methods to solve such problems in order to obtain nearly optimal solutions in a reasonable amount of computational time. Noon (1988) proposed several heuristics, the most promising of which is an adaptation of the well-known nearest-neighbor heuristic. Renaud and Boctor (1998) developed a heuristic called GI^3. Snyder and Daskin (2006) described a random-key genetic algorithm. Karapetyan and Gutin (2012) proposed an efficient local search algorithm for known and new neighborhoods. And very recently, a number of other meta-heuristics have also been proposed by Pintea *et al.*, (2017) with ant algorithm, K. Helsgaun (2015) with lin-kernighan-helsgaun algorithm, and Smith and Imeson (2017) with an effective large neighborhood search heuristic for solving the GTSP.

This paper proposes a meta-heuristic by using a local improvement algorithm called *k*-opt; more specifically, a *2*-opt local search method. However, additional efforts have been spent in fine tuning the search with some basic components of moves, random start, neighborhood (*NN*), and shortest path (*SP*) methodologies to construct more efficient optimization procedures.

3. Components

3.1. The *k*-opt approach.

k-opt attempts to improve a given tour by removing up to *k* edges that are not all adjacent and, replacing them with a corresponding number of new edges so that a new tour is generated. Note that, in the new tour, some segments of the original tour may be reversed.

The complexity of the *k*-opt procedure is $O(m^k)$ where m is the number of clusters. In other words, the neighborhood contains $O(m^k)$ solutions, and checking the objective value

of any solution requires a fixed amount of time. The *2*-opt involves the substitution of two edges, say {*i*, *i*+1} and {*j*, *j*+1}, with two other edges {*i*, *j*} and {*i*+1, *j*+1}.

This process is illustrated in Figure 3.1.

Note that the orientation of the path (*i*+1, *i*+2, …., *j*) is *reversed*. Such an exchange results in a local improvement if and only if the improvement of move (defined by δ) is greater than 0:

$\delta(i,j) = d[\sigma(i), \sigma(i+1)] + d[\sigma(j), \sigma(j+1)] - d[\sigma(i), \sigma(j)] - d[\sigma(i+1), \sigma(j+1)] > 0.$ (3.1).

A complete iteration of this *2*-opt heuristic consists of trying all the possible pairs of the current sequence and stopping if no improvement can be achieved during a complete iteration.

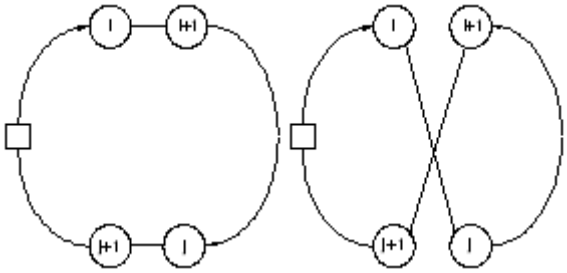


Figure 3.1: The *2*-opt procedure (*k* =2).

3.2. Moving procedures

This procedure consists of three specific moves of swapping, reversing, and inserting. These moves are demonstrated using a tour such as {0-1-2-3-4-5-6-7-8-9}.

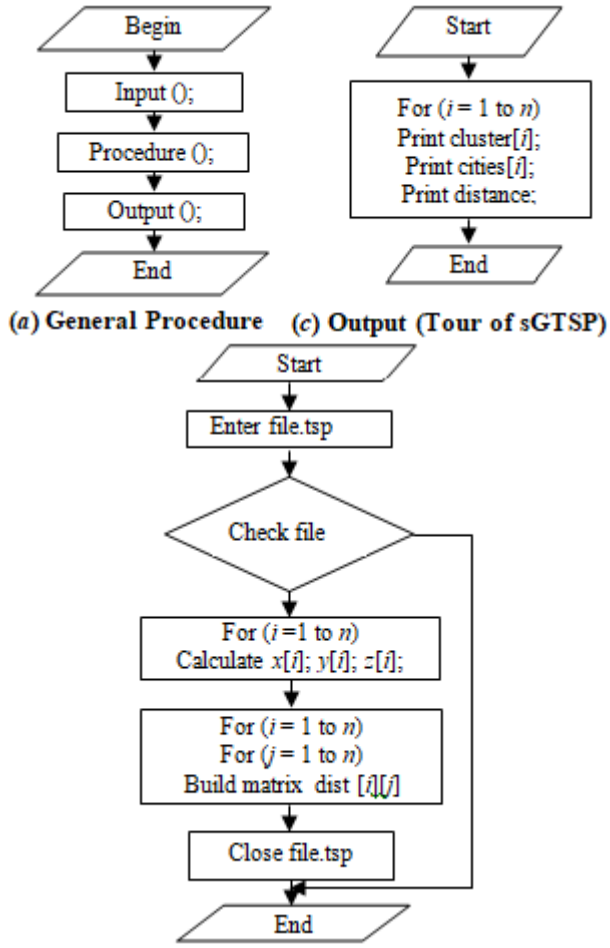
- *Swapping* two paths means interchanging the places of two sequences in the tour. For instance, swapping the path {**3-4-5}** and {**7-8**} results in {0-1-2-**7-8**-6-**3-4-5**-9-0}.
- *Reversing* a path means changing the sequence of nodes in a segment of the tour, e.g. reversing the path {**3-4-5-6}** gives the route {0-l-2-**6-5-4-3**-7-8-9}.
- *Inserting* a node/cluster means inserting a number in each location of the tour. For instance inserting 5 into the above example will result the sequence of {0-**5**-l-**5**-2-**5**-3-**5**-4-5-6-**5**-7-**5**-8-**5**-9}.

3.3. Random start

Each run of the proposed algorithm will start at a node randomly picked from the set of nodes for the problem. The quality of solution will depend on the node where the solution starts. The use of multiple trials increases the chance of getting closer to the optimum solution.

4. The Proposed Algorithms

4.1 The general outline of the complete algorithm is depicted in figure 4.1 (*a*), (*b*), and (*c*).



(*a*) General Procedure **(*c*) Output (Tour of sGTSP)**

(*b*) Input (construct distance matrix)

Figure 4.1: Main steps of the procedures.

4.2 The NN heuristic algorithm

The *NN* procedure is summarized as follows (Figure A.1 in the appendix 1).
The input: The matrix of distances (Figure 4.1.*b*).
The output: The initial tour of *s*GTSP (Figure 4.1.*c*).
Step 1: Start randomly at cluster *k*, and city *i* that belongs to the cluster *k*. Mark the cluster *k* = 1 (*k*=1: visited; *k*=0: not visited yet).
Step 2: Among non-visited clusters find the nearest node with city *i* and add this node to the tour. Mark this cluster.
Step 3: Continue this procedure until all clusters are marked. This will form the initial *m*-edge tour.
Step 4: Construct the tour and print out this tour. Stop the *NN* procedure.

4.3 The 2-opt classic algorithm

This procedure includes two parts: (1) the *NN* heuristics as defined in figure A.1; (2) *2*-opt classic heuristics as in figure A.2.
The input**:** the initial *m*-edge tour of the *s*GTSP with the distance matrix (Figure 4.1.*b*).
The output: an improved tour of the *s*GTSP.
Step 1: Construct the initial *m*-edges tour of the *s*GTSP by *NN* heuristics (Figure A.1).
Step 2: Use the *2*-opt move with the *m*-edges tour. Assume "*Non-fixed cluster, fixed candidate cities*", the problem is

now a TSP. Perform the local search move around node *i* (*i* = 1 to *m*-2) and node *j* (*j* = *i*+2 to *m*) (Figure A.2), and calculate the δ following the equation (3.1): $\delta = d(\sigma_{i-1}, \sigma_i) + d(\sigma_j, \sigma_{j+1}) - d(\sigma_{i-1}, \sigma_j) - d(\sigma_i, \sigma_{j+1})$ (Figure A.3).
Step 3: If $\delta \leq 0$: go to step 2 with the new value of *i*, *j*. If $\delta > 0$: perform the move *i*, *j* (Figure A.4). A new tour is constructed with the improved distance.
Step 4: Go to step 2 with the new tour which now becomes the current tour. The procedure is continued until no more improvement is made. Stop and print out the current tour with its distance.

4.4 The 2-opt–NN algorithm

This algorithm includes two parts: (1) the *NN* heuristics as defined in figure A.1; (2) the *2*-opt-*NN* heuristics as shown in figure A.5.

The input is the initial *m*-edge tour of the *s*GTSP with the distance matrix (Figure 4.1.*b*).

The output is an improved tour.
Step 1: This step includes all steps of section 4.2 (Figure A.1). This step constructs the initial *m*-edge tour of the *s*GTSP by *NN* heuristics.
Step 2: Use the *2*-opt-*NN* moves with the initial *m*-edge tour. The problem is now a *s*GTSP. Perform a local search move in cluster *i* (*i* = 1 to *m*-1) and cluster *j* (*j* = *i*+1 to *m*) (Fig. A.5), and calculate δ following the equation (3.1) with a modification as: $\delta = (dist\ 1 - dist\ 2)$ (Figure A.6).
Step 3: If $\delta \leq 0$: go to step 2 with the new value of *i*, *j*. If $\delta > 0$: perform the move for these *i*, *j*. (Figure A.7). This procedure is continued with value of *i* increases by 1, while *j* decreases by 1. The new tour of the *s*GTSP is constructed with the improved distance.
Step 4: Go to step 2 with the new tour as the current tour. Continue the procedure until there is no new improvement. Stop and print out the current tour with its distance.

4.5 The 2-opt–SP algorithms

This procedure consists of two parts: (1) The *NN* heuristics as defined in figure A.1; (2) The *2*-opt-*SP* heuristics (figure A.8). The input**:** the current tour of *s*GTSP and weighted distance matrix (Figure A.9).

The output: An improved tour of the *s*GTSP.

Step 1: This step includes all steps of section 4.2 (Figure A.1). This step constructs the initial *m*-edge tour of the *s*GTSP by *NN* heuristics.
Step 2: Use the *2*-opt-*SP* moves with the initial *m*-edge tour. The problem becomes a *s*GTSP. Use Dijstra's algorithm with cluster *i* (*i* = 1 to *m*) and cluster *j* (*j* = *i*+2 to *m*), (Figure A.8) to calculate the improvement $\delta = (dist\ 1 - dist\ 2)$ (Figure A.10).
Step 3: If $\delta \leq 0$, go to step 2 with the new value of *i*, *j*. If $\delta > 0$, perform the move of these *i*, *j* (Figure A.11). This procedure is continued with value of *i* increases by 1, while *j* decreases by 1. The new tour of the *s*GTSP is constructed with the improved distance.
Step 4: Go to step 2 with the new tour as the current tour. Continue the procedure until there is no more improvement. Stop and print out the current tour with its distance.

5. Results and Discussions

The four proposed algorithms are implemented in C^{++} and tested on an Intel ® core™ 2Duo CPU E7500@ 2.93GHz, 2.87GB of RAM running under window XP on 36 TSPLIB problems. The abridged data are shown in table 5.1 and table 5.2 (full data are in the appendix 2).

For each problem, the algorithms are run five times to examine their performance and variation of results from trial to trial. The best performance of the five trials is used as the solution (UB) and compared to the optimal solution (O^*) reported by Fischetti *et al.*, [10]. The quality of the solution for each problem over the proposed algorithms is calculated: gap (%) = $100*(UB\text{-}O^*)/O^*$.

Table 5.1: Data of four proposed algorithms

36 Problems	O^*	The *NN*			The *2*-opt-classic			The *2*-opt-*NN*			The *2*-opt-*SP*		
		UB	gap	Time	UB	gap	Time	UB	gap	Time	UB	gap	Time
EIL 51	174	189	8.62	0.01	191	9.77	0.01	178	2.30	0.02	174	0.00	0.00
ST 70	316	350	10.76	0.00	335	6.01	0.00	326	3.16	0.02	320	1.27	0.01
EIL 76	209	240	14.83	0.01	217	3.83	0.02	214	2.39	0.01	209	0.00	0.02
PR 76	64925	79114	21.85	0.00	69882	7.63	0.02	66162	1.91	0.03	64925	0.00	0.03
…….													
PR 226	64007	69241	8.18	0.01	66035	3.17	0.03	64722	1.12	0.08	64007	0.00	1.72
…….													
PCB 442	21657	28996	33.89	0.05	24817	14.59	0.03	23804	9.91	3.47	23329	7.72	44.76
Average			**16.92**	**0.02**		**10.53**	**0.02**		**5.62**	**0.50**		**3.17**	**7.20**

From table 5.1, it is clear that the *2*-opt-*SP* outperforms the other solution heuristics. On the average it gives solutions within 3.17% of the optimum in an average of 7.2 seconds of CPU. The second good result is given by the *2*-opt–*NN*, which produced an average percentage increase over the optimum of 5.62% in an average time of 0.5 CPU seconds.

Table 5.2: Data of the GI^3, the *G-NN*, the *GA* and the *2*-opt-*SP*

36 Problems	O^*	The GI^3			The *G-NN*			The *GA*			The *2*-opt-*SP*		
		UB	gap	Time	UB	gap	Time	UB	gap	Time	UB	gap	Time
EIL 51	174	174	0.00	0.30	174	0.00	0.40	174	0.00	0.10	174	0.00	0.00
ST 70	316	316	0.00	1.70	316	0.00	0.80	316	0.00	0.20	320	1.27	0.01
EIL 76	209	209	0.00	2.20	209	0.00	1.10	209	0.00	0.20	209	0.00	0.02
PR 76	64925	64925	0.00	2.50	64925	0.00	1.90	64925	0.00	0.20	64925	0.00	0.03
……													
PR 226	64007	64007	0.00	25.50	65395	2.17	67.60	64007	0.00	1.00	64007	0.00	1.72
……													
PCB 442	21657	22936	5.91	567.70	21704	0.22	838.40	22025.8	1.70	10.10	23329	7.72	44.76
Average			**1.05**	**82.61**		**1.48**	**171.56**		**0.13**	**1.63**		**3.17**	**7.20**

Table 5.2 compares the *2*-opt-*SP* with other composite heuristics presented in literature including the GI^3 by Renaud and Boctor [7]; the *G-NN* by Noon [3] and the *GA* by Snyder *et al.*, [9]. It shows that the *GA* produces the best results within 0.13% of the optimum in an average of 1.63 seconds of CPU. Although the solutions produced by the GI^3 are almost as good with 1.05 % of the optimum, its average computation time is 82.61 CPU seconds. It is over 11 times slower compared with the *2*-opt-*SP* (7.2 CPU seconds).

Table 5.3: Summary of performance measures for 36 TSPLIB problems

Measure	The NN	The 2-opt-classic	The 2-opt-NN	*The 2-opt-SP*
Number of optimum solutions	0	0	0	4
Average of gap (%)	16.92	10.53	5.62	3.17
Min % increase above the optimum	3.24	2.19	0.90	0.00
Max % increase above the optimum	33.89	19.32	13.32	9.90
Average computational time	0.02	0.02	0.50	7.20
Min computation time	0.00	0.00	0.01	0.00
Max computation time	0.05	0.05	5.27	87.38

Table 5.3 summarizes the results given in the previous tables and shows that the *2*-opt-*SP* produced the smallest average percentage increase above the optimum (3.17%), and in four occasions it reached the optimum. This table also shows that the *2*-opt–*NN* produces good results. Generally, the average computational time of the *2*-opt-*SP* is comparable with the *GA* (Figure 5.1).

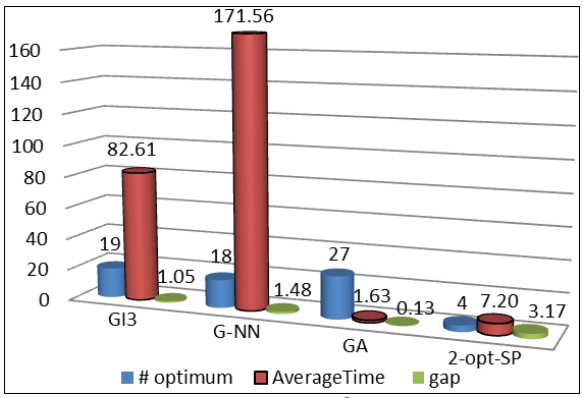


Figure 5.1: Comparison of the GI^3, the *G-NN*, the *GA*, and the *2*-opt-*SP*.

Specially, the *2*-opt-*SP* produced the shortest computational time with the problems up to 300 nodes (0.86 CPU seconds) (Figure 5.2).

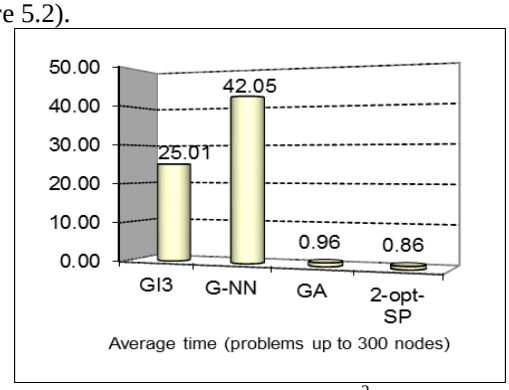


Figure 5.2: Time comparison of the GI^3, the *G-NN*, the *GA*, and the *2*-opt-*SP*.

6. Conclusions and Recommendations

This paper investigated the symmetric generalized traveling salesman problem. The GTSP provides an attractive way of modeling a wide range of situations.

The most efficient local searches in this research are the adaptations of the well-known *2*-opt with the neighborhood search (called the *2*-opt-*NN*), and the shortest path approach (called the *2*-opt-*SP*) for solving the *s*GTSP. The assessment of the quality of the solution has been made by treating the proposed algorithms as a "black-box" routine for 36 TSPLIB problems, for which optimal solutions are available. Three other composite heuristics as well as the optimal solution were used as the basis for comparison. It has been shown that with problems up to 300 nodes and when there is a need for a fast solution, the *2*-opt-*SP* requires the smallest average CPU seconds and may turn out to be the preferred approach.

For example, delivery routes for companies like DHL and UPS are quickly changing every day. Companies cannot take too much time to find optimal solution for their tours. They need fast algorithms to recalculate tours in a few seconds. In such cases, the *2*-opt-*SP* can be the best choice.

Further research is required to study more combinations of GTSP local searches, in which we could develop more *k*-opt procedures with $k > 2$. It is expected that there are yet additional opportunities to significantly improve the performance of GTSP.

Moreover, it seems worthwhile to combine some meta-heuristics as *GA*, guided local search, ant colony approaches are also recommended for investigation.

7. Acknowledgment

I would like to express my heartfelt gratefulness to Prof. Farhad Azadivar, lecturer in the department of Mechanical Engineering, University of Massachusetts Dartmouth for his invaluable guidance, and enthusiasm for reading various drafts of the research.

References

[1] B. Bontoux, C. Artigues, and D. Feillet, « Memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem," *Computers and Operations Research 37*, 1844-1852, 2010.
[2] Camelia-M. Pintea, Petrică C. Pop and Camelia Chira, "The generalized traveling salesman problem solved with ant algorithms," *Complex Adapt Syst Model*, DOI 10.1186/s40294-017-0048-9, 2017.
[3] Ch. E. Noon, "The generalized traveling salesman problem," *Ph.D. Dissertation*, University of Michigan, 1988.
[4] D. Ben-Arieh, G. Gutin, M. Penn, A. Yeo, and A. Zverovitch, "Process planning for rotational parts using the generalized traveling salesman problem," *IJPR 41;* 2581-2596, 2003.
[5] D. Karapetyan, and G. Gutin, "Efficient local search algorithms for known and new neighborhoods for the generalized traveling salesman problem," *EJOR 219*, 234-251, 2012.
[6] G. Laporte, A. Asef-Vaziri, and C. Sriskandarajah, "Some applications of the generalized traveling salesman problem," *Journal of the Operation Research Society 47*, 1461-1467, 1996.
[7] J. Renaud, and F. F. Boctor, "An efficient composite heuristics for the symmetric generalized traveling salesman problem," *EJOR 108*, 571-584, 1998.
[8] K. Helsgaun, "Solving the equality generalized traveling salesman problem using the lin–kernighan–helsgaun algorithm," *Mathematical Programming Computation*, vol. 7, 269–287, 2015.
[9] L. V. Snyder, and M. S. Daskin, "A random-key genetic algorithm for the generalized Traveling salesman problem," *EJOR 174*, 38-53, 2006.
[10] M. Fischetti, J. J. S. Gonzalez, and P. Toth, (1997). A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research 45*, 378-394.
[11] O. Nalivajevs, D. Karapetyan, "Conditional Markov Chain Search for the Generalized Travelling Salesman Problem for Warehouse Order Picking," 2019.
[12] P. Beullens, L. Muyldermans, D. Cattrysse, and D. V. Oudheusden, "A guided local search heuristic for the capacitated arc routing problem," *EJOR 147*; 629-643, 2003.
[13] S. L. Smith and F. Imeson, "An effective large neighborhood search heuristic for the generalized traveling salesman problem," *Computers & Operations Research*, vol. 87, 1–19, 2017.

Author Profile

The author received the engineer degree in chemical engineering from Ho Chi Minh City University of Technology in 1996, the diploma study in industrial engineering from the Asian Institute of Technology (AIT) - Thailand in 2001, and industrial management master degree at the Katholieke Universiteit Leuven (KUL) - Belgium in 2003. From 1996 to 2009, he has gained a lot of experiences from multinational companies. He successfully completed an MRP model for the KAO Vietnam - a Japanese company. And he developed a hybrid optimization approach on forecasting-inventory-planning model for the S.C. Johnson & Son – a American company, further gave insight into MRPII application. Since 2010, he was a lecturer in the International University, Vietnam National University – HCMC and others as the Asian Institute of Technology Center in Vietnam (AITCV), HOASEN university. His research interest includes combinatorial optimization in logistics and supply chain, production planning, operations management, and engineering economy. Mr. Author's awards and honors includes the prize of Ho Chi Minh City Plastic & Rubber Association on thesis research, the Petro Vietnam award for diploma study in industrial engineering at the AIT, and the full scholarship from the Belgian Technical Cooperation to complete the master program in industrial management at the KUL.

Appendices

Appendix 1: The Proposed Algorithms

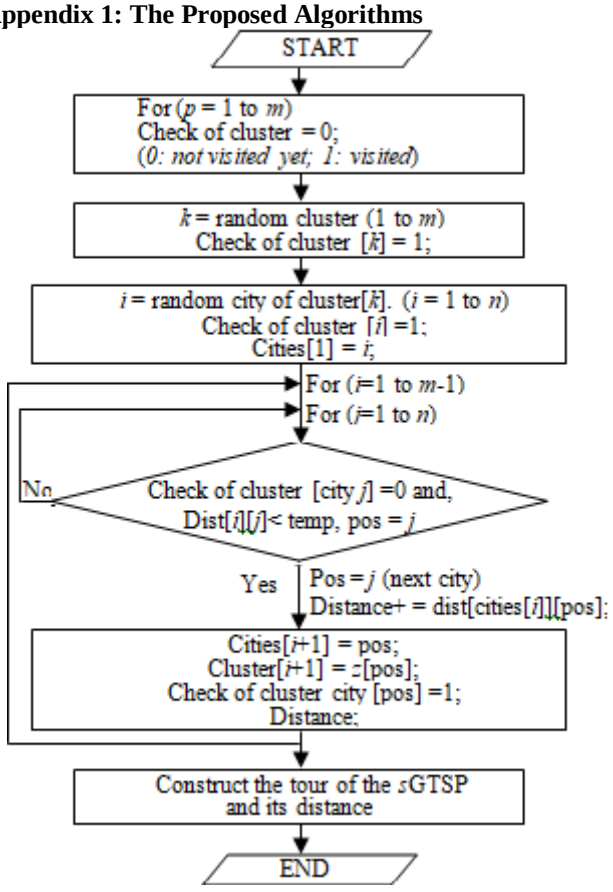


Figure A.1 The *NN* procedure

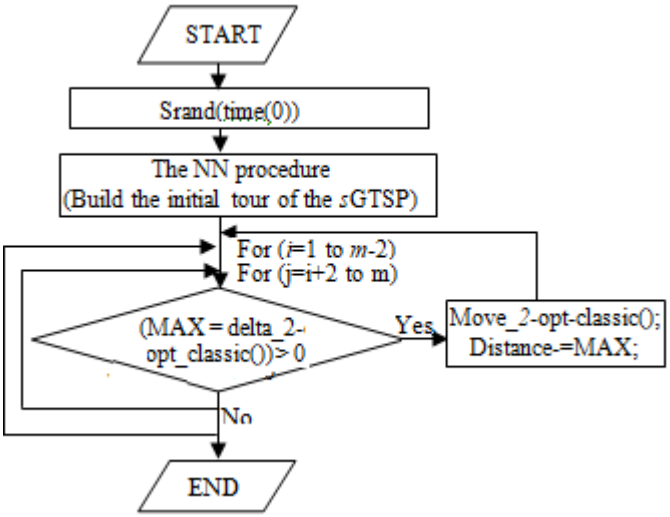


Figure A.2: The random start + *2*-opt-classic procedure

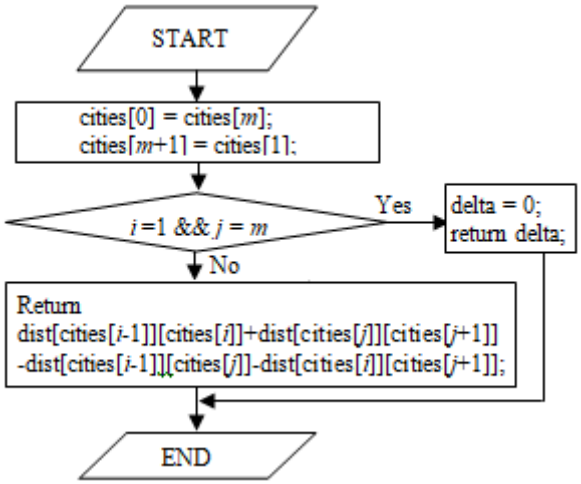


Figure A.3 The δ calculation for the *2*-opt-classic

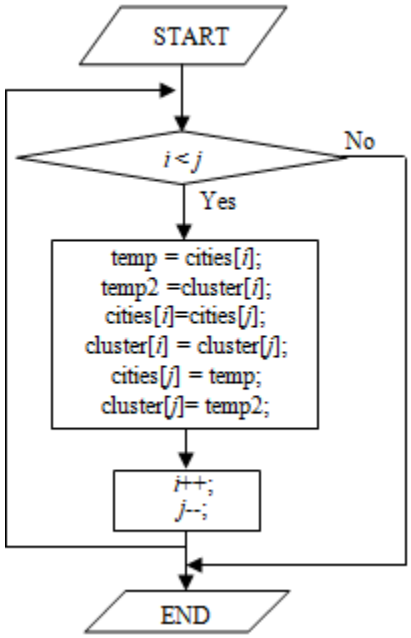


Figure A.4 The *2*-opt-classic moves

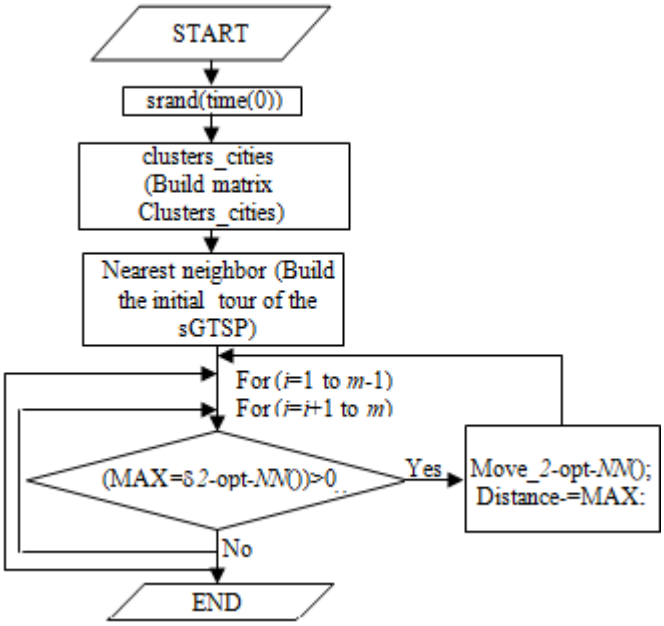


Figure A.5 The *2*-opt-*NN* procedure

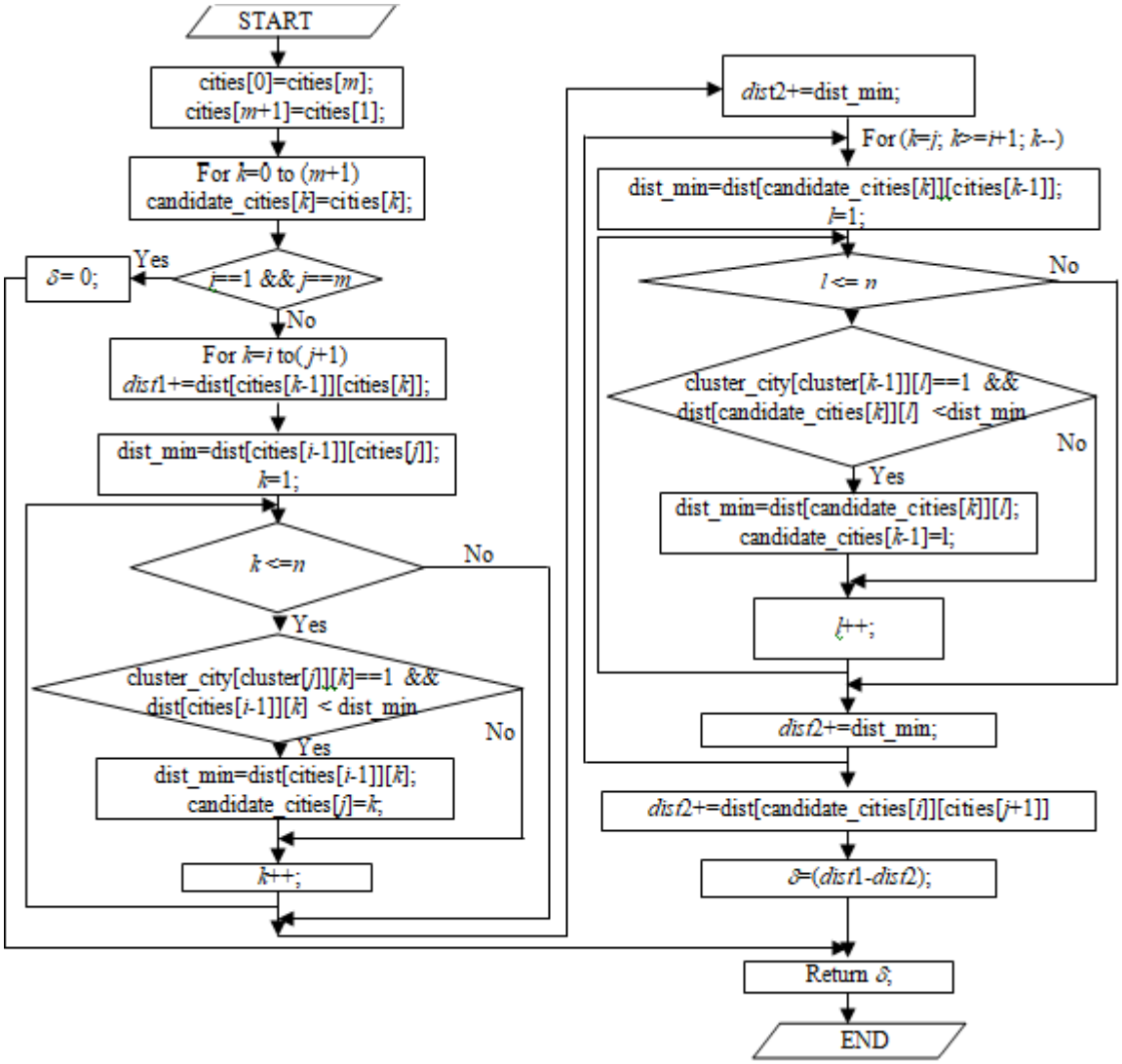


Figure A.6 The δ calculation for the *2*-opt-*NN*

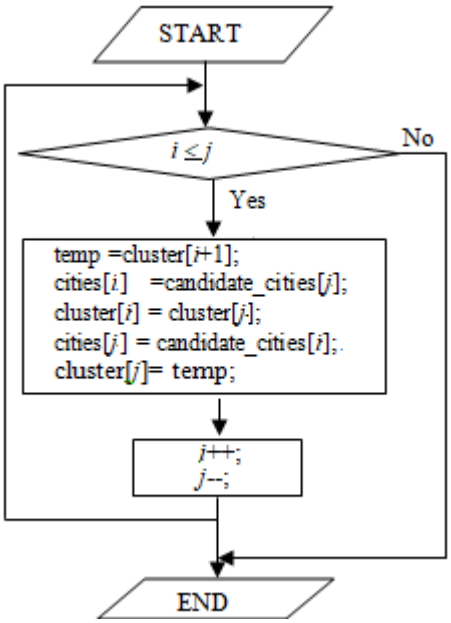


Figure A.7 The *2*-opt-*NN* moves

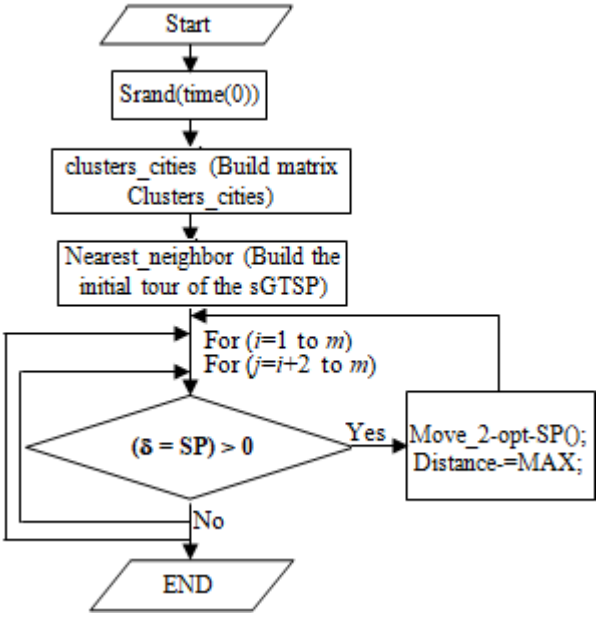


Figure A.8 The *2*-opt-*SP* procedure

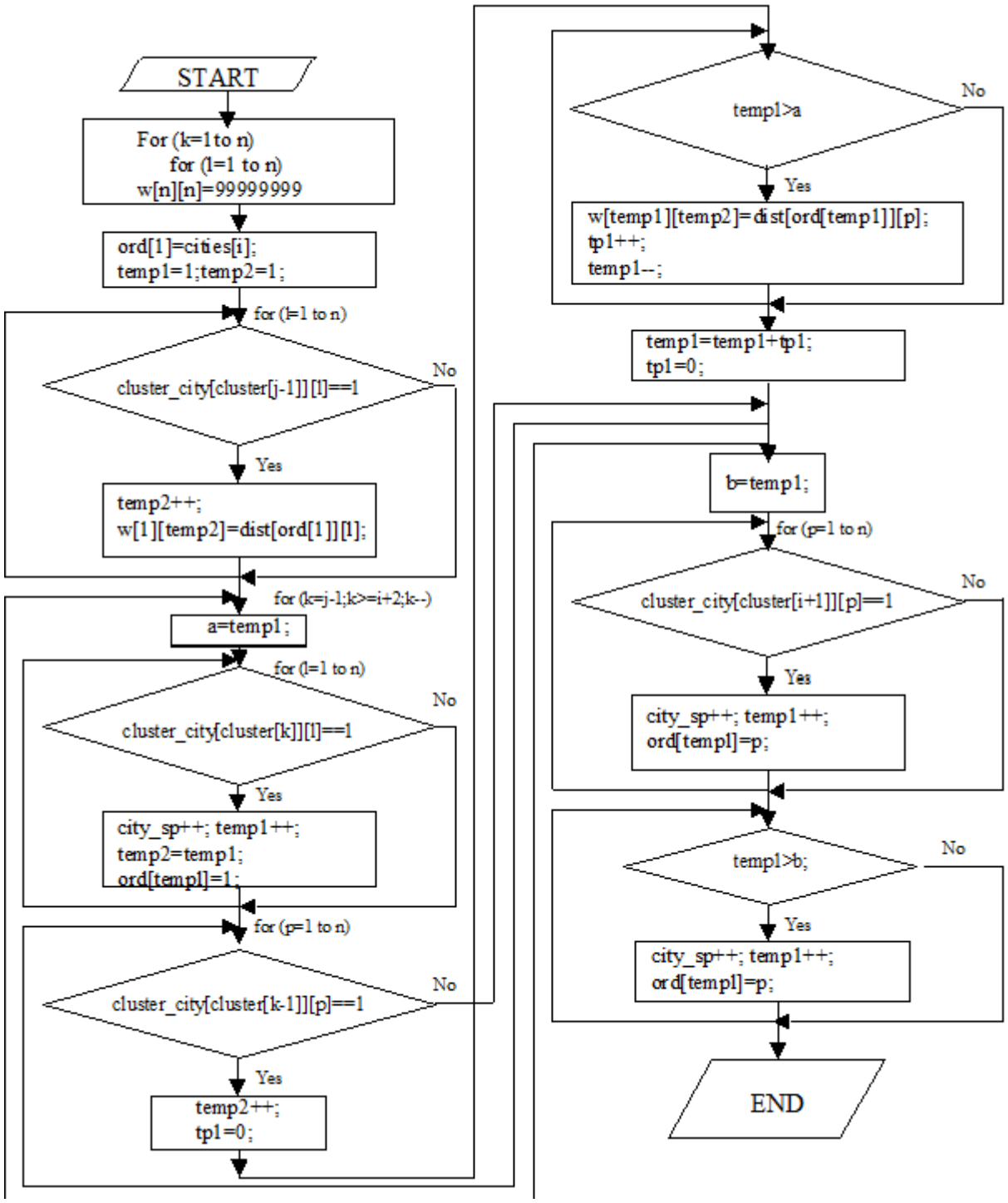


Figure A.9 The weight distance matrix

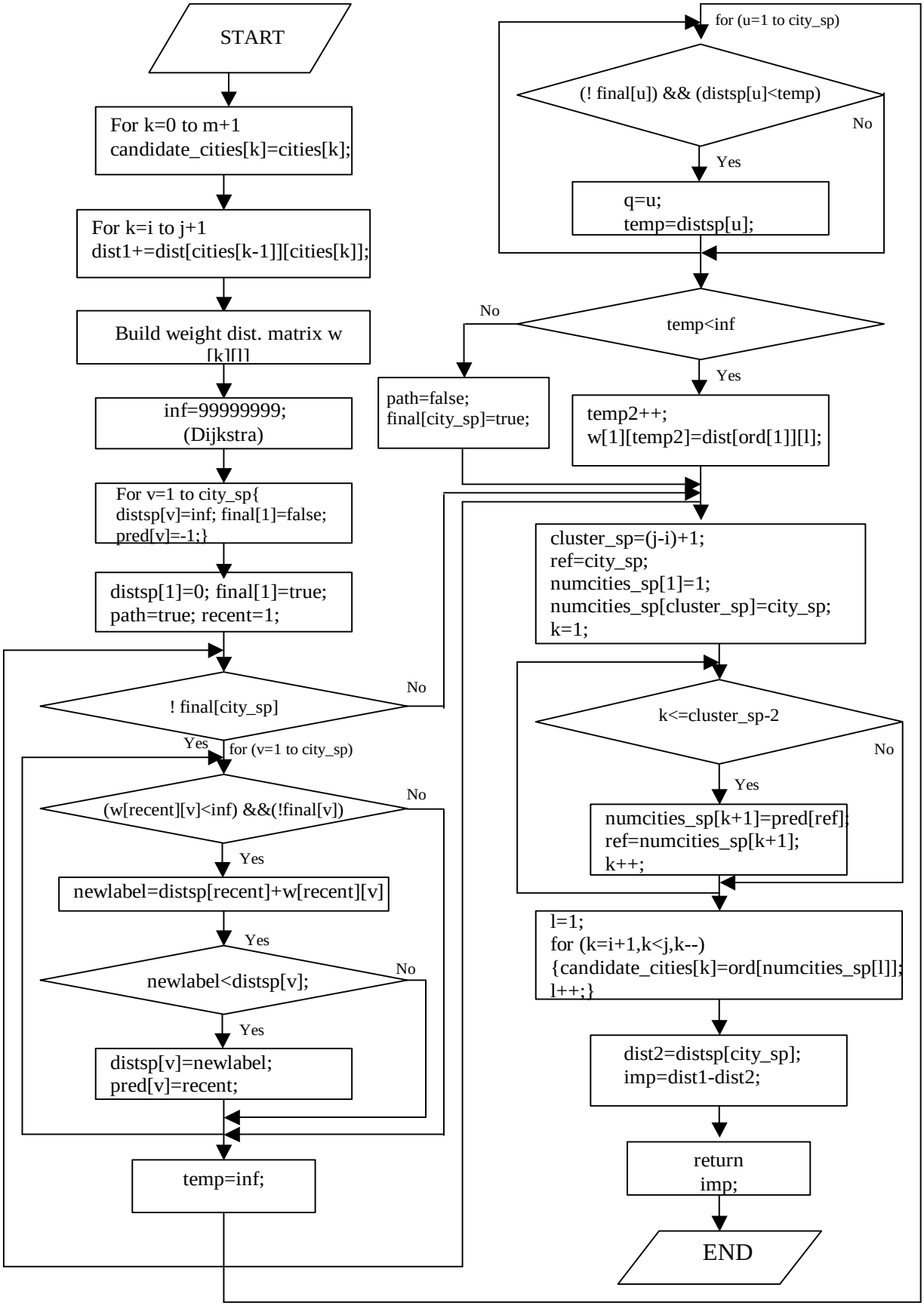


Figure A.10 The Dijstra's algorithm

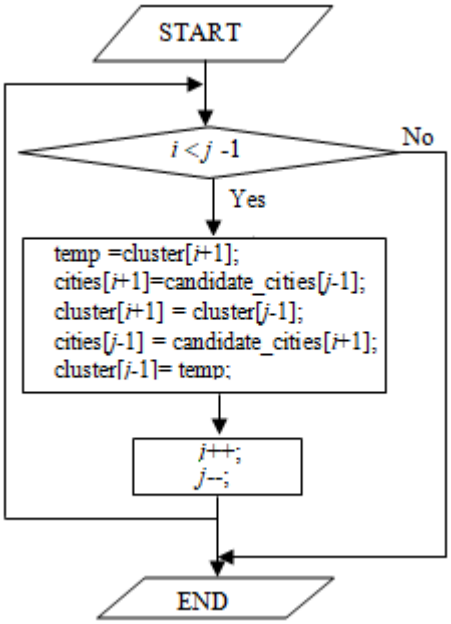


Figure A.11: The *2*-opt-*SP* moves

Appendix 2: Computational data.

Table 5.1: *Data of four proposed algorithms*

36 Problems	O^*	The *NN*			The *2*-opt-classic			The *2*-opt-*NN*			The *2*-opt-*SP*		
		UB	gap	Time	UB	gap	Time	UB	gap	Time	UB	gap	Time
EIL 51	174	189	8.62	0.01	191	9.77	0.01	178	2.30	0.02	174	0.00	0.00
ST 70	316	350	10.76	0.00	335	6.01	0.00	326	3.16	0.02	320	1.27	0.01
EIL 76	209	240	14.83	0.01	217	3.83	0.02	214	2.39	0.01	209	0.00	0.02
PR 76	64925	79114	21.85	0.00	69882	7.63	0.02	66162	1.91	0.03	64925	0.00	0.03
RAT 99	497	578	16.30	0.01	563	13.28	0.01	511	2.82	0.03	499	0.40	0.06
KRO A100	9711	11171	15.03	0.01	10615	9.31	0.02	10385	6.94	0.02	10507	8.20	0.03
KRO B100	10328	11136	7.82	0.01	11136	7.82	0.02	10836	4.92	0.02	10335	0.07	0.02
KRO C100	9554	11842	23.95	0.02	10974	14.86	0.03	10827	13.32	0.03	10035	5.03	0.05
KRO D100	9450	10383	9.87	0.00	10127	7.16	0.01	10138	7.28	0.02	9528	0.83	0.03
KRO E100	9523	10469	9.93	0.01	10469	9.93	0.03	10078	5.83	0.03	9633	1.16	0.05
RD 100	3650	4199	15.04	0.00	3937	7.86	0.01	3796	4.00	0.03	3795	3.97	0.01
EIL 101	249	273	9.64	0.02	268	7.63	0.02	264	6.02	0.01	251	0.80	0.03
LIN 105	8213	8829	7.50	0.01	8760	6.66	0.02	8439	2.75	0.03	8232	0.23	0.05
PR 107	27898	28803	3.24	0.00	28509	2.19	0.01	28150	0.90	0.01	27990	0.33	0.06
PR 124	36605	40000	9.27	0.01	39520	7.96	0.03	37993	3.79	0.03	36805	0.55	0.11
BIER 127	72418	85262	17.74	0.00	79168	9.32	0.01	75059	3.65	0.05	78329	8.16	0.11
PR 136	42570	51026	19.86	0.01	47349	11.23	0.03	45111	5.97	0.05	44305	4.08	0.23
PR 144	45886	47905	4.40	0.01	47541	3.61	0.02	46468	1.27	0.05	45890	0.01	0.28
KRO A150	11018	13091	18.81	0.01	12664	14.94	0.01	11854	7.59	0.05	11298	2.54	0.19
KRO B150	12196	13786	13.04	0.01	13504	10.72	0.03	13026	6.81	0.06	12419	1.83	0.25
PR 152	51576	56852	10.23	0.01	55483	7.58	0.01	52722	2.22	0.05	51820	0.47	0.50
U 159	22664	28330	25.00	0.01	26443	16.67	0.01	24483	8.03	0.05	23813	5.07	0.36
RAT 195	854	1128	32.08	0.01	1019	19.32	0.02	951	11.36	0.31	884	3.51	0.95
D 198	10557	12369	17.16	0.01	11450	8.46	0.03	10843	2.71	0.14	10637	0.76	3.03
KRO A200	13406	17225	28.49	0.02	15937	18.88	0.03	14768	10.16	0.11	14059	4.87	1.16
KRO B200	13111	17544	33.81	0.02	15480	18.07	0.01	14115	7.66	0.16	13805	5.29	1.59
TS 225	68340	76395	11.79	0.01	75887	11.04	0.03	72090	5.49	0.06	70251	2.80	0.50
PR 226	64007	69241	8.18	0.01	66035	3.17	0.03	64722	1.12	0.08	64007	0.00	1.72
GIL 262	1013	1227	21.13	0.02	1186	17.08	0.01	1106	9.18	0.48	1057	4.34	2.92
PR 264	29549	34697	17.42	0.02	32686	10.62	0.03	30858	4.43	0.34	31555	6.79	4.63
PR 299	22615	29271	29.43	0.02	25463	12.59	0.03	24477	8.23	0.69	24009	6.16	7.66
LIN 318	20765	26520	27.71	0.02	24256	16.81	0.03	22332	7.55	1.42	22433	8.03	13.63
RD 400	6361	8045	26.47	0.05	7573	19.05	0.01	7105	11.70	2.94	6991	9.90	47.00
FL 417	9651	10662	10.48	0.05	9954	3.14	0.03	9870	2.27	1.80	9866	2.23	39.93
PR 439	60099	71002	18.14	0.05	66325	10.36	0.05	64195	6.82	5.27	64205	6.83	87.38
PCB 442	21657	28996	33.89	0.05	24817	14.59	0.03	23804	9.91	3.47	23329	7.72	44.76
Average			**16.92**	**0.02**		**10.53**	**0.02**		**5.62**	**0.50**		**3.17**	**7.20**

Table 5.2 Data of the GI^3, the *G-NN*, the *GA* and the *2*-opt-*SP*

36 Problems	O^*	The GI^3			The *G-NN*			The *GA*			The *2*-opt-*SP*		
		UB	gap	Time	UB	gap	Time	UB	gap	Time	UB	gap	Time
EIL 51	174	174	0.00	0.30	174	0.00	0.40	174	0.00	0.10	174	0.00	0.00
ST 70	316	316	0.00	1.70	316	0.00	0.80	316	0.00	0.20	320	1.27	0.01
EIL 76	209	209	0.00	2.20	209	0.00	1.10	209	0.00	0.20	209	0.00	0.02
PR 76	64925	64925	0.00	2.50	64925	0.00	1.90	64925	0.00	0.20	64925	0.00	0.03
RAT 99	497	497	0.00	5.00	497	0.00	7.30	497	0.00	0.70	499	0.40	0.06
KRO A100	9711	9711	0.00	6.80	9711	0.00	3.80	9711	0.00	0.40	10507	8.20	0.03
KRO B100	10328	10328	0.00	6.40	10328	0.00	2.40	10328	0.00	0.40	10335	0.07	0.02
KRO C100	9554	9554	0.00	6.50	9554	0.00	6.30	9554	0.00	0.30	10035	5.03	0.05
KRO D100	9450	9450	0.00	8.60	9450	0.00	5.60	9450	0.00	0.40	9528	0.83	0.03
KRO E100	9523	9523	0.00	6.70	9523	0.00	2.80	9523	0.00	0.80	9633	1.16	0.05
RD 100	3650	3653	0.08	7.30	3653	0.08	8.30	3650	0.00	0.30	3795	3.97	0.01
EIL 101	249	250	0.40	5.20	250	0.40	3.00	249	0.00	0.20	251	0.80	0.03
LIN 105	8213	8213	0.00	14.40	8213	0.00	3.70	8213	0.00	0.30	8232	0.23	0.05
PR 107	27898	27898	0.00	8.70	27898	0.00	5.20	27898	0.00	0.40	27990	0.33	0.06
PR 124	36605	36762	0.43	12.20	36605	0.00	12.00	36605	0.00	0.60	36805	0.55	0.11
BIER 127	72418	76439	5.55	36.10	79431	9.68	7.80	72418	0.00	0.50	78329	8.16	0.11
PR 136	42570	43117	1.28	12.50	44930	5.54	9.60	42570	0.00	0.50	44305	4.08	0.23
PR 144	45886	45886	0.00	16.30	45886	0.00	11.80	45886	0.00	0.30	45890	0.01	0.28
KRO A150	11018	11018	0.00	17.80	11018	0.00	22.90	11018	0.00	1.30	11298	2.54	0.19
KRO B150	12196	12196	0.00	14.20	12196	0.00	20.10	12196	0.00	1.00	12419	1.83	0.25
PR 152	51576	51820	0.47	17.60	52506	1.80	10.30	51576	0.00	1.50	51820	0.47	0.50
U 159	22664	23254	2.60	18.50	23296	2.79	26.50	22664	0.00	0.60	23813	5.07	0.36
RAT 195	854	854	0.00	37.20	865	1.29	86.00	854.0	0.00	0.70	884	3.51	0.95
D 198	10557	10620	0.60	60.40	10620	0.60	118.80	10557	0.00	1.20	10637	0.76	3.03
KRO A200	13406	13406	0.00	29.70	14110	5.25	53.00	13406	0.00	2.70	14059	4.87	1.16
KRO B200	13111	13111	0.00	35.80	13111	0.00	135.20	13111.6	0.00	1.40	13805	5.29	1.59
TS 225	68340	68756	0.61	89.00	68340	0.00	117.80	68352.0	0.02	2.40	70251	2.80	0.50
PR 226	64007	64007	0.00	25.50	65395	2.17	67.60	64007	0.00	1.00	64007	0.00	1.72
GIL 262	1013	1064	5.03	115.40	1032	1.88	122.70	1020.6	0.75	1.90	1057	4.34	2.92
PR 264	29549	29655	0.36	64.40	31241	5.73	147.20	29549	0.00	1.30	31555	6.79	4.63
PR 299	22615	23119	2.23	90.30	23069	2.01	281.80	22638.8	0.11	6.10	24009	6.16	7.66
LIN 318	20765	21719	4.59	206.80	21787	4.92	317.00	20893.6	0.62	3.50	22433	8.03	13.63
RD 400	6361	6439	1.23	386.10	6614	3.98	1137.10	6436.4	1.19	3.50	6991	9.90	47.00
FL 417	9651	9932	2.91	427.10	9754	1.07	1341.00	9655.6	0.05	2.40	9866	2.23	39.93
PR 439	60099	62215	3.52	611.00	62514	4.02	1238.90	60258.4	0.27	9.10	64205	6.83	87.38
PCB 442	21657	22936	5.91	567.70	21704	0.22	838.40	22025.8	1.70	10.10	23329	7.72	44.76
Average			**1.05**	**82.61**		**1.48**	**171.56**		**0.13**	**1.63**		**3.17**	**7.20**