A Time-Dependent ATSP With Time Window and Precedence Constraints in Air Travel

Thanaboon Saradatta, Pisut Pongchairerks
Faculty of Engineering, Thai-Nichi Institute of Technology, Bangkok, Thailand.
pisut@tni.ac.th

Abstract—**This paper considers a time-dependent asymmetric travelling salesman problem with time window and precedence constraints, based on the real application of air transport. This problem is much more complicated than the classical asymmetric travelling salesman problem due to the properties of the airfare prices, the time window constraints and the precedence constraints. To solve this problem, this paper proposes a modified nearest neighbor algorithm and two local search algorithms.**

Index Terms—**Travelling Salesman Problem; TSP; Asymmetric Travelling Salesman Problem; ATSP; Local Search Algorithm; Air Transportation; Time Window; Time-Dependent; Precedence Constraint.**

I. INTRODUCTION

The classical travelling salesman problem (TSP) is to decide the roundtrip for a salesman to travel around a number of given cities with the objective of minimizing total distance. TSP involves with not only the salesman's application but also with other actual economic applications. In the past, TSP has usually been applied to the ground transport applications. Nowadays, it is however very usual to transport between countries by air. This statement makes guidance for this paper to consider an extension of TSP where the salesman travels around a number of given countries by air. The conditions of this extended TSP are summarized as follows:

1. The airfare prices from a country to another country in the same time period offered by different airlines may be different.
2. The airfare price to travel from a country to another country may not be same to the airfare price to travel in the reverse direction. (This makes the problem asymmetric.)
3. The airfare price to travel from a country to another country offered by an airline may change over time. (This makes the problem time-dependent.)
4. It is possible that there are no flights to travel from a country to another country.
5. A country may have to be visited within a pre-assigned time period. (This is a time window constraint.)
6. A country may have to be visited immediately after a predefined preceding country. (This is a precedence constraint.)
7. Each country must be visited once and the final destination is the starting country. (This is same to the condition of the traditional TSP.)

This extended TSP is more complicated than TSP due to the conditions (i) through (vi). Since the objective of problem is to minimize the total cost of the airfare prices, the best decision on the condition (i) is simply made by selecting the lowest-airfare-price airline among all airlines available for the same direct trip from a country to another in the same time period. With this decision for the condition (i), this problem becomes the Time-Dependent Asymmetric Travelling Salesman problem with Time Window and Precedence Constraints. It is called TD-ATSP-TWPC in this paper.

The published articles that have the contents relating to TD-ATSP-TWPC are given as follows. The air travel planning problem discussed in [1] and [2] is a variant of the shortest-path problem which requires finding the lowest-cost trip or roundtrip between two specific countries. The articles [1] and [2] also indicate that the air travel planning problem is far more complicated than the classical shortest-path problem due to the properties of airfare prices.

The article [3] considers the ATSP in the air transport application. However, its objective is to minimize the total distance. Thus, the problem in [3] does not face with all conditions relating to airfare prices. Also, it does not have the time window and precedence constraints. Thus, the major difficulty of the problem considered in [3] beyond the traditional ATSP is due to the condition (iv) only. The travelling tourist problem (TTP) discussed in [4] may be the most similar problem of TD-ATSP-TWPC, since TTP is the ATSP whose costs of routes change with time. However, TTP does not have the conditions (iv), (v) and (vi). The article [4] uses evolutionary Markov chain Monte Carlo and simulated annealing to solve TTP. Other variants of TSP similar to TD-ATSP-TWPC include ATSP with precedence constraints (i.e., sequence ordering problem) presented in [5], and the time-dependent TSP presented in [6].

The reviews of TSP and its variants are given in [7-10]. A review article about the time-window constrained routing problems is shown in [11]. In literature, there are a number of algorithms developed for solving TSP or its variants. These algorithms can be classified into three types as follows:

1. Simple construction heuristics, such as Nearest Neighbor algorithm [7, 12].
2. Exact algorithms, such as Branch-and-Cut algorithms [5 , 13].
3. Meta-heuristic algorithms, such as Local Search algorithms [14, 15], a Variable Neighborhood Search algorithm [16].

In this paper, Section 2 presents the statement of TD-ATSP-TWPC. Section 3 presents the instances for TD-ATSP-TWPC. Section 4 proposes a modified nearest neighbor algorithm, while Section 5 proposes the two local search algorithms for TD-ATSP-TWPC. Section 6 evaluates the performances of the three proposed algorithms. Finally, Section 7 provides a conclusion.

II. Statement of TD-ATSP-TWPC

TD-ATSP-TWPC consists of a salesman and N given countries. These N countries include country 1, country 2,…, country N. The salesman has to visit all N countries within N weeks; in addition, he/she must visit only a single country per week. The last visited country (i.e., the country visited in the week N) must be the same to the starting country (i.e., the country he starts). If there are at least two flights available to travel from country i to country j in week k, the airline with the lowest airfare price will be selected for travelling from country i to country j in week k. Let c_{ijk} be the lowest airfare price over the airfare prices offered by all available airlines to travel from country i to country j in week k, where i, j and k = 1, 2,…, N, and $i \neq j$. In addition, c_{ijk} is possibly unequal to c_{jik}. It is also possible that there are no available flights to travel from country i to country j in week k. Moreover, this problem has time window constraints and the precedence constraints, which are described below.

- For time window constraints of TD-ATSP-TWPC, let country $w_k \in \{1, \dots, N\}$ be the country which must be visited in week k (where $k = 1, 2,\dots, N - 1$). The w_N is not included here, because the week N is pre-assigned for visiting the starting country. Any week k which has not been pre-assigned for a specific country will have w_k = null. For example, if $w_1 = 3$, $w_3 = 4$ and other w_k are null, this means country 3 must be visited in week 1, and country 4 must be visited in week 3.
- For precedence constraints of TD-ATSP-TWPC, let country $a_q \in \{1, \dots, N\}$ be the country which must be visited immediately before country $b_q \in \{1, \dots, N\}$ for the same q, where $q = 1, 2,\dots, Q$ and $a_q \neq b_q$. Let Q be the number of all pairs of countries a_q and their immediate successive countries b_q. For example, If $Q = 2$, $a_1 = 3$, $b_1 = 5$, $a_2 = 4$ and $b_2 = 2$, country 3 must be visited immediately before country 5, and country 4 must be visited immediate before country 2.

In this paper, the $a_1, a_2,\dots, a_Q, b_1, b_2,\dots, b_Q, w_1, w_2,\dots, w_{N-1}$ are not same to one another. The objective of TD-ATSP-TWPC is to minimize the total cost of travelling around these N countries. The total cost is the sum of all airfare prices used to complete the roundtrip.

III. Proposed Problem Instances

This paper generates six instances based on actual data. In these instances, the locations of countries are taken from the locations of airports in those countries as given below:

1. Suvarnabhumi International Airport in Bangkok, Thailand.
2. Changi Airport in Singapore.
3. Kuala Lumpur International Airport in Kuala Lumpur, Malaysia.
4. Narita International Airport in Tokyo, Japan;
5. Incheon International Airport in Seoul, South Korea.
6. Beijing Capital International Airport in Beijing, China.
7. Hong Kong International Airport in Hong Kong;
8. Sydney International Airport in Sydney, Australia;
9. JFK International Airport in New York, United States of America.
10. London Heathrow Airport in London, United Kingdom.
11. Charles de Gaulle Airport in Paris, France.
12. Leonardo da Vinci Airport in Rome, Italy.
13. Frankfurt Airport in Frankfurt, Germany.
14. Madrid-Barajas Airport in Madrid, Spain;
15. Auckland Airport in Auckland, New Zealand.
16. Dublin Airport in Dublin, Ireland.
17. Toronto Pearson International Airport in Toronto, Canada.
18. Athens International Airport in Athens, Greece.
19. Istanbul Ataturk Airport in Istanbul, Turkey.
20. Zurich Airport in Zurich, Switzerland.

The common information used in all six instances is given as follows:

1. The salesman must start his roundtrip from Thailand.
2. The salesman must visit exactly one country per week.
3. The lowest airfare price to travel from the country i to the country j in week k (i.e., c_{ijk}) and the airline, which offers this lowest airfare price are given in [17].

The lowest airfare prices in [17] are shown in Baht, where $1 = 33.7 Baht during the period of collecting data. All flights in weeks 1 through 20 were taken off on 7 Jun 2015, 14 Jun, 21 Jun, 28 Jun, 5 Jul, 12 Jul, 19 Jul, 26 Jul, 2 Aug, 9 Aug, 16 Aug, 23 Aug, 30 Aug, 6 Sep, 13 Sep, 20 Sep, 27 Sep, 4 Oct, 11 Oct and 18 Oct, respectively. These six instances are classified into two sets based on the similarities on the number of all countries, the number of countries whose their visited week has been pre-assigned, and the number of all pairs of the countries and their immediate successive countries. Set 1 includes Instances 1 through 3, and Set 2 includes instances 4 through 6.

Each instance in Set 1 considers only 15 countries (i.e., $N = 15$) including (1) Thailand, (2) Singapore, (3) Malaysia,…., and (15) New Zealand. The salesman thus must visit all 15 countries within 15 weeks; moreover, he must visit each country per week. Each instance in Set 1 has only one country, which has a pre-assigned visited week as well as only one pair of a country and its immediate successive country. The details of Instances 1 through 3 are given as follows. In Instance 1, the salesman must visit Singapore in week 2, and he must visit France immediately after Italy (equivalent to he must visit Italy immediately before France). In Instance 2, the salesman must visit South Korea in week 6, and he must visit Malaysia immediately after Japan. In Instance 3, the salesman must visit Japan in week 9, and he must visit China immediately after Germany.

Each instance in Set 2 considers 20 countries (i.e., $N = 20$) including all countries listed above. The salesman of each instance in Set 2 must visit all 20 countries within 20 weeks; and, he must visit each country per week. Each instance in Set 2 has two countries, which have the pre-assigned visited weeks and two pairs of countries and their immediate successive countries. In Instance 4, the salesman must visit United Kingdom in week 6 and Malaysia in week 9; moreover, he must visit Japan immediately before Australia as well as visiting France immediately before Hong Kong. In Instance 5, the salesman must visit Germany in week 6 and Australia in week 11; moreover, he must visit China immediately before USA as well as visiting South Korea immediately before Malaysia. In Instance 6, the salesman must visit France in week 8 and South Korea in week 13; moreover, he must visit Australia immediately before New Zealand as well as visiting Hong Kong immediately before United Kingdom.

IV. Modified Algorithm

The nearest neighbor algorithm (NN) [7] is the most well-known heuristic for TSP and its variants. Thus, it should be used to compare with the local search algorithms proposed in the next section. This section hence modifies the original NN algorithm to be able to solve TD-ATSP-TWPC. This modified NN is hereafter called MNN. The steps of MNN are given as follows.

Step 1: Let the cost of a direct trip from a country to another country is the lowest airfare price among all airfare prices offered by all available airlines. Assign the starting country, and let the starting country be the current country (i.e., the country where the salesman locates now). Let $k = 1$.

Step 2: Select the country where the salesman must visit next by following these steps:

- Step 2.1: Check that if there is a country that must be visited in week k due to a time window constraint. If so, assign this country as the next country, and then go to Step 3. Otherwise, go to Step 2.2.
- Step 2.2: Check that if there is a country that must be visited immediately after the current country due to a precedence constraint. If so, assign this country as the next country, and then go to Step 3. Otherwise go to Step 2.3.
- Step 2.3: Select the next country by following Steps 2.3.1 through 2.3.4:
  - Step 2.3.1: Let L be a List of all possible countries which can be visited in week k. Construct L by adding every as-yet-unvisited country which can be visited by one or more flights from the current country into the list.
  - Step 2.3.2: Delete every country having a predefined preceding country from the list L.
  - Step 2.3.3: If there is a country which must be visited in week k + 1 due to a time window constraint, delete every country having no flights departing to this country and also delete every country having a predefined successive country from the list L.
  - Step 2.3.4: Select the country which can be visited from the current country with the lowest cost among all countries in the list L as the next country. Then, go to Step 3.

Step 3: Let the salesman move from the current country to the next country selected in Step 2; then, update the new current country. If k is less than N, increase k by 1, and then repeat from Step 2. If k equals N, let the salesman move back to the starting country; and the construction of the roundtrip is now completed.

V. Proposed Local Search Algorithms

The solution representation used in this paper is modified from the traditional solution representation for TSP widely used in a number of articles, e.g. [18]. The local search algorithms proposed here represent their solutions (i.e., TD-ATSP-TWPC roundtrips) by the permutations. Each permutation is the sequence of $N - 1$ integers, including 2, 3,…, N. The interpretation for each permutation is given as follows: the number i located in the *k-th* position in the permutation means that the country i will be visited in week k, where $i = 2, 3,…, N$ and $k = 1, 2,…, N - 1$. The number 1 is not included into the permutation, because the country 1 is always set as the starting country for every instance. It is also known that the starting country (i.e., the country 1) will be visited in week N. In this paper, Thailand is the starting country for every instance. An example of decoding from a permutation into a solution is given as follows: for a 5-country instance, the permutation (4, 2, 5, 3) means the roundtrip that the salesman departs from the country 1 to visit the country 4 in week 1, then departs from the country 4 to visit the country 2 in week 2, then departs from the country 2 to visit the country 5 in week 3, then departs from the country 5 to visit the country 3 in week 4, and he finally departs from the country 3 to visit the country 1 in week 5.

The two local search algorithms proposed in this paper are given in Sections 5.1 and 5.2 based on the special swap and insert operators, respectively. These operators select countries randomly with some conditions while the traditional operators [19] select countries randomly without conditions. The additional conditions enable the algorithms to avoid or reduce generating infeasible neighbor solutions.

In each algorithm, the user must input the values for c_{ijk} ($i, j, k = 1, 2,…, N$ and $i \neq j$), w_k ($k = 1, 2,…, N - 1$), a_q and b_q ($q = 1, 2,…, Q$) before using the algorithm. The permutations P_0 and P_1 represent the roundtrips S_0 and S_1, respectively. The coding and decoding procedures used in the algorithm are already explained in this section. For each algorithm, S_0 is the current best solution; it will then be the final solution after the algorithm is stopped. In the proposed algorithms, every lowest airfare price to travel from a country to another country that violates one or more constraints will be set to a large amount of money, says 90 million Baht, as a penalty cost [20].

A. *Local Search Algorithm with Swap Operator*

The proposed local search algorithm, which uses the swap operator is hereafter called LS-SWAP. The steps of LS-SWAP are given as follows.

Step 1: If there are no flights to travel from the country i to the country j in week k, let c_{ijk} = 90 million Baht, for i, j and $k = 1, 2,…, N$, and $i \neq j$.

Step 2: Randomly generate a feasible roundtrip for TD-ATSP-TWPC; let S_0 be this roundtrip. Code S_0 into the permutation P_0.

Step 3: Let $t = 0$.

Step 4: Generate a neighbor permutation P_1 and a neighbor solution S_1 by the following steps:

- Step 4.1: Randomly select a number $u \in \{2, 3,…, N\}$ under these conditions:
  - u cannot equal any of w_k for $k = 1,…, N - 1$.
  - u cannot equal any of b_q for $q = 1,…, Q$.
- Step 4.2: Randomly select a number $v \in \{2, 3,…, N\}$ under these conditions:
  - v cannot equal any of w_k for $k = 1,…, N - 1$.
  - v cannot be any of b_q for $q = 1,…, Q$.
  - v cannot equal u.
- Step 4.3: Generate P_1 based on the following conditions:
  - If u and v are both unequal to any of a_q for $q = 1,…, Q$, generate P_1 by switching the positions between the number u and the number v in P_0.
  - Otherwise, generate P_1 by switching the positions between u and v in P_0 and also switching the positions between the number located immediately after u and the number located immediately after v in P_0.
- Step 4.4: Decode P_1 into S_1.

Step 5: If the total cost of S_1 is less than or equal to the total cost of S_0, then let S_0 equal to S_1 as well as letting P_0 equal to P_1, and repeat from Step 3; otherwise, increase t by 1 and go to Step 6.

Step 6: If t equals to $N(N - 1)$, stop. Otherwise, repeat from Step 4.

B. *Local Search Algorithm with Insert Operator*

The proposed local search algorithm, which uses the insert operator is hereafter called LS-INSERT. The steps of LS-INSERT are given as follows.

Step 1: Let c_{ijk} = 90 million Baht (for i, j and $k = 1, 2,…, N$, and $i \neq j$) if one or more following conditions are met.

- If there are no flights to travel from the country i to the country j in week k.
- If $i = a_q$ and $j \neq b_q$ for the same q, where $q = 1,…, Q$.
- If $i \neq a_q$ and $j = b_q$, for the same q, where $q = 1,…, Q$.

Step 2: Randomly generate a feasible roundtrip for TD-ATSP-TWPC; let S_0 be this roundtrip. Code S_0 into the permutation P_0.

Step 3: Let $t = 0$.

Step 4: Generate a neighbor permutation P_1 and a neighbor solution S_1 by the following steps:

- Step 4.1: Randomly select a number $u \in \{2, 3,…, N\}$ under these conditions:

```
        - u cannot equal any of $w_k$ for $k = 1,…, N – 1$.
        - u cannot equal any of $b_q$ for $q = 1,…, Q$.
Step 4.2: Randomly select a number $v \in \{2, 3,…, N\}$ under these conditions:
        - v cannot equal any of $a_q$ for $q = 1,…, Q$.
        - v cannot equal u.
        - If $u = a_q$, then v cannot equal $b_q$ for the same q where $q = 1,…, Q$.
Step 4.3: Generate $P_1$ by removing u from its old position in $P_0$, and then inserting u immediately after v in $P_0$. After the $P_1$ has been constructed, if any of $w_k$ ($k = 1,…, N – 1$) is not located in the k-th position in $P_1$, this $P_1$ must be repaired by removing this $w_k$ from the current position and then inserting it into the k-th position in $P_1$.
Step 4.4: Decode $P_1$ into $S_1$.
Step 5: If the total cost of $S_1$ is less than or equal to the total cost of $S_0$, then let $S_0$ equal to $S_1$ as well as letting $P_0$ equal to $P_1$, and repeat from Step 3; otherwise, increase t by 1 and go to Step 6.
Step 6: If t equals to $N(N – 1)$, stop. Otherwise, repeat from Step 4.
```

In both LS-SWAP and LS-INSERT, the S_0 is always a feasible solution for TD-ATSP-TWPC, since the initial S_0 is feasible. However, S_1 can be an infeasible solution for TD-ATSP-TWPC. In LS-SWAP, S_1 can be infeasible only due to the condition that there are no flights to transport between two countries. In LS-INSERT, S_1 can be infeasible due to the conditions of no flights and the precedence constraints. Every infeasible S_1 solution generated by LS-SWAP and LS-INSERT will return the large cost (i.e., 90 million Baht in this paper); and hence it cannot be selected as the next S_0.

VI. Numerical Experiment

To evaluate the performances of LS-SWAP and LS-INSERT, each algorithm will be run 10 times. Each run uses different initial permutation randomly generated. LS-SWAP and LS-INSERT are coded on C# and run on a personal computer of Intel(R) Core(TM) i5-2430M CPU @ 2.40 GHz with a 4 GB RAM.

In this paper, a solution of each algorithm is a roundtrip generated by the algorithm and a solution value is the total cost of the roundtrip generated by the algorithm. Table 1, for each instance, shows the solution value given by MNN, the best found solution value over 10 runs (Best) given by each local search algorithm, the percentage of improvement of the best found solution value over 10 runs given by each local search algorithm from the solution value given by MNN (% Improve), the average solution value found over 10 runs (Avg) given by each local search algorithm, and the average computational time per run in seconds (Avg Time) of each local search algorithm. All costs are in Baht (let US$1 = 33.7 Baht). Note that MNN will be run only one time per instance, since it is a deterministic algorithm.

Table 1 shows that LS-SWAP performs best on Instances 5 and 6, while LS-INSERT performs best on Instances 1 through 4. LS-SWAP and LS-INSERT both perform better than MNN on all six instances. The average % improvement of the best found solution value over 10 runs of LS-SWAP from the solution value of MNN on all six instances is 16.4%, while the average % improvement of the best found solution value over 10 runs of LS-INSERT from the solution value of MNN on all six instances is 15.1%. The average % improvement of the best found solution value over 10 runs of LS-SWAP from the average % improvement of the best found solution value over 10 runs of LS-INSERT is 1.1%.

This paper then tests the population means of % Improvements by stating the five pairs of H0 versus H1 based on the results from Table 1. Let % Improve of Algorithm A from Algorithm B for each instance, where A and B are any algorithms, refers to the % improvement of the best found solution value of Algorithm A from the best found solution value of Algorithm B. For LS-SWAP and LS-INSERT, the best found solution is the best solution found over 10 runs. For MNN, the best found solution is the solution over a single run, since it is a deterministic algorithm. The five pairs of H0 and H1 are given as follows:

1. H0: the population mean of % Improves of LS-SWAP from MNN for all instances is zero versus H1: this population mean is greater than zero.
2. H0: the population mean of % Improves of LS-INSERT from MNN for all instances is zero versus H1: this population mean is greater than zero.
3. H0: the population mean of % Improves of LS-SWAP from LS-INSERT for all instances is zero versus H1: this population mean is greater than zero.
4. H0: the population mean of % Improves of LS-INSERT from LS-SWAP for the instances using the conditions of Set 1 is zero versus H1: this population mean is greater than zero.
5. H0: the population mean of % Improves of LS-SWAP from LS-INSERT for the instances using the conditions of Set 2 is zero versus H1: this population mean is greater than zero.

To test the hypotheses above, the five hypothesis tests are conducted by using the significance level of 0.20. The results of these five hypothesis tests are shown in Table 2. The results of the hypothesis tests for (I) and (II) are that the mean of % Improves of LS-SWAP from MNN and the mean of % Improves of LS-INSERT from MNN are both significantly greater than zero. As the conclusion, on average, LS-SWAP and LS-INSERT both outperform MNN with the significance level of 0.20. The hypothesis test for (III) fails to reject H0. It concludes that there are no enough evidences to support that LS-SWAP outperforms LS-INSERT on average with the significance level of 0.20. However, the result from the hypothesis test for (IV) concludes that, on average, LS-INSERT outperforms LS-SWAP for the instances using the conditions of Set 1 with the significance level of 0.20. On the contrary, the result from the hypothesis test for (V) concludes that, on average, LS-SWAP outperforms LS-INSERT for the instances using the conditions of Set 2 with the significance of 0.20.

The recommendations based on the above results are that LS-INSERT is proper to use for easy instances while LS-SWAP is proper to use for hard instances. An easy instance refers to an instance that has at most one pair of a country and its immediate successive country as well as having at most one country whose its visited week is pre-assigned. A hard instance refers to an instance that has at least two pairs of countries and their immediate successive countries as well as at least two countries whose their visited weeks are pre-assigned. Although the instances in Set 1 and Set 2 are also different in number of all countries, it is believed that the number of all countries has no much effect on the instance difficulty compare to the effects from the number of time window constraints and the number of precedence constraints.

The main reason that LS-SWAP, on average, is better than LS-INSERT on hard instances is because the insert operator has a higher possibility than the swap operator to generate an infeasible solution for S1 for each use of operator. The swap operator can generate an infeasible solution for S1 only in the case that there are no flights to travel from a country to its next country in the generated roundtrip, while the insert operator can generate an infeasible solution for S1 in the case that there are no flights to travel from a country to its next country as well as the case that the country which has a pre-assigned successive country cannot be visited immediately before its successive country.

Table 1
Results taken from MNN, LS-SWAP and LS-INSERT

Instance	MNN	LS-SWAP				LS-INSERT			
		Best	% Improve	Avg	Avg Time	Best	% Improve	Avg	Avg Time
1	114,616	111,193	3.0	149,509	0.033	102,280	10.8	133,068	0.041
2	121,674	116,309	4.4	145,173	0.039	115,122	5.4	160,814	0.043
3	173,739	141,966	18.3	179,276	0.036	133,329	23.3	164,028	0.039
4	297,508	200,610	32.6	236,436	0.081	196,620	33.9	245,355	0.087
5	224,823	194,682	13.4	225,415	0.072	208,989	7.0	236,990	0.067
6	203,281	149,065	26.7	196,294	0.071	182,081	10.4	223,917	0.088

Table 2
Results of one-sample *t*-test in performance competitions

Variable	Sample Size	Mean	Std. Dev.	*t*	*p-value*
% Improve of LS-SWAP from MNN	6	16.40	11.87	3.38	0.010
% Improve of LS-INSERT from MNN	6	15.13	11.14	3.33	0.010
% Improve of LS-SWAP from LS-INSERT	6	1.12	9.89	0.28	0.397
% Improve of LS-INSERT from LS-SWAP on Set 1	3	5.03	3.62	2.41	0.069
% Improve of LS-SWAP from LS-INSERT on Set 2	3	7.63	10.08	1.31	0.160

VII. Conclusion

This paper considers the time-dependent asymmetric travelling salesman problem with time window and precedence constraints or TD-ATSP-TWPC based on the actual application of air travel. Three algorithms are proposed in this paper, namely MNN, LS-SWAP and LS-INSERT. MNN is the modified nearest neighbor algorithm for solving TD-ATSP-TWPC especially. LS-SWAP and LS-INSERT are the local search algorithms developed for TD-ATSP-TWPC, based on the modified swap and insert operators respectively. The swap and insert operators developed in this paper randomly select countries with some additional conditions in order to enable the algorithms to reduce the chance to generate infeasible neighbor solutions. LS-SWAP and LS-INSERT both perform very well in terms of solution quality as well as CPU time. Based on the analysis, LS-INSERT is more recommended for easy instances, i.e., the instances with at most one pair of a country and its immediate successive country as well as at most one country whose the visited week is pre-assigned. On the other hand, LS-SWAP is more recommended for hard instances, i.e., the instances with at least two pairs of countries and their immediate successive countries as well as at least two countries whose the visited weeks are pre-assigned.

References

[1] Marcken, C. D. 2003. Computational Complexity of Air Travel Planning. *Public Notes on Computational Complexity*. Retrieved October, 22, 2015 from http://www.demarcken.org/carl/papers/

[2] Robinson, S. 2002. Computer Scientists Find Unexpected Depths in Airfare Search Problem. *SIAM NEWS*. 35(6). Retrieved November, 14, 2015 from http://www.msri.org/people/members/sara/articles/ airfares.pdf

[3] OpenFlights. 2015. *The Air-Traveling Salesman*. Retrieved October, 22, 2015 from https://sites.google.com/site/ travellingcudasalesman/

[4] Touyz, J. 2013. The Travelling Tourist Problem: A Mixed Heuristic Approach. Retrieved October, 22, 2015 from http://issuu.com/jgetouyz/docs

[5] Ascheuer, N., Jünger, M., and Reinelt, G. 2000. A Branch & Cut Algorithm for the Asymmetric Traveling Salesman Problem with Precedence Constraints. *Computational Optimization and Application*. 17(1): 61–84.

[6] Picard, J.-C. and Queyranne, M. 1978. The Time-Dependent Traveling Salesman Problem and Its Application to the Tardiness Problem in One-Machine Scheduling. *Operations Research*. 26(1): 86–101.

[7] Applegate, D. L., Bixby, R. E., Chvátal, V., and Cook, W. J. 2007. *The Traveling Salesman Problem: A Computational Study*. New Jersey: Princeton University Press.

[8] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., and Shmoys, D. B. 1995. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. New York: John Wiley & Sones.

[9] Gutin, G. and Punnen, A. P. 2002. *The Traveling Salesman Problem and Its Variations*. US: Springer.

[10] Cook, W. J. 2012. *In Pursuit of the Traveling Salesman*. New Jersey: Princeton University Press.

[11] Solomon, M. M. and Desrosiers, J. 1988. Time Window Constrained Routing and Scheduling Problem. *Transportation Science*. 22(1): 1-13.

[12] Kizilateş, G. and Nuriyeva, F. 2013. On the Nearest Neighbor Algorithms for the Traveling Salesman Problem. *Advances in Computational Science, Engineering and Information Technology*. 225: 111-118.

[13] Hernández-Pérez, H. and Salazar-González, J. J. 2004. A Branch-and-Cut Algorithm for a Traveling Salesman Problem with Pickup and Deliver. *Discrete Applied Mathematics*. 145(1): 126-139.

[14] Voudouris, C. and Tsang, E. 1999. Guided Local Search and Its Application to the Traveling Salesman Problem. *European Journal of Operational Research*. 113(2): 469-499.

[15] Misevičius, A., Ostreika, A., Šimaitis, A., and Žilevičius, V. 2007. Improving Local Search for the Traveling Salesman Problem. *Information Technology and Control*. 36(2): 187-195.

[16] Piriyaniti, I. and Pongchairerks, P. 2013. Variable Neighbourhood Search Algorithms for Asymmetric Travelling Salesman Problems. *International Journal of Operational Research.* 18(2): 157-170.

[17] Saradatta, T. and Pongchairerks, P. 2015. Instances for Time-Dependent ATSP with Time Window and Precedence Constraints in Air Travel. Retrieved November, 20, 2015 from https://drive.google.com/folderview?id=0B2XqS3TSsvP7UFFEa1E0 UlpWbHc&usp=sharing

[18] Ray, S.S. and Bandyopadhyay, S. 2007. Genetic Operators for Combinatorial Optimization in TSP and Microarray Gene Ordering', Applied Intelligence. 26(3): 183-195.

[19] Guo, P. and Wenming, C. 2014. A General Variable Neighborhood Search for Single-Machine Total Tardiness Scheduling Problem with Step-Deteriorating Jobs. *Journal of Industrial and Management Optimization*. 10(4): 1071-1090.

[20] Smitch, A. E. and Coit, D. W. 1997. Constraint-Handling Techniques - Penalty Functions. In Baeck, T., Fogel, D. and Michalewicz, Z. (Eds) *Handbook of Evolutionary Computation (C 5.2)*. Bristol: Oxford University Pres.