Journal of the Eastern Asia Society for Transportation Studies, Vol.9, 2011

A Beam Search Heuristic for the Traveling Salesman Problem with Time Windows

Kun-Chih WU
PhD Student
Department of Industrial Engineering and Management
Yuan Ze University
135 Yuan-Tung Road, Chung-Li,
Taiwan 32003, R.O.C.
Fax: +866-3-4638907
E-mail: s968907@mail.yzu.edu.tw

Ching-Jung TING
Associate Professor
Department of Industrial Engineering and Management
Yuan Ze University
135 Yuan-Tung Road, Chung-Li,
Taiwan 32003, R.O.C.
Fax: +866-3-4638907
E-mail: ietingcj@saturn.yzu.edu.tw

Wei-Chun LAN
Graduate Assistant
Department of Industrial Engineering and Management
Yuan Ze University
135 Yuan-Tung Road, Chung-Li,
Taiwan 32003, R.O.C.
Fax: +866-3-4638907
E-mail: s975419@mail.yzu.edu.tw

Abstract: The traveling salesman problem with time windows (TSPTW) is a variety of the traveling salesman problem. In practice, temporal aspect is a necessary constraint with respect to the routing problems. TSPTW is a NP-hard problem, and computational time of exact algorithm increases exponentially as the number of customers increases. Hence, we present a beam search (BS) algorithm which is a heuristic based on breadth-first branch-and-bound without backtracking to solve the TSPTW. BS filters out worse nodes by a local evaluation and only keeps β (called beam width) nodes according to global evaluation at each level. In our BS, both one-step local evaluation and global evaluation are applied to estimate cost of nodes by inserting unvisited nodes of a given initial solution. The computational results on test instances from the literature show that our beam search can obtain good solutions with effective computational times for TSPTW.

Key Words: *traveling salesman problem with time windows, beam search, logistics*

1. INTRODUCTION

Physical distribution problem is a popular subject in past few decades, due to its mathematical properties and practical applications. Traveling salesman problem (TSP) is a classical problem in physical distribution system. With arising of practical applications, more constraints have been discussed to enrich this fundamental problem. In this study, we consider traveling salesman problem with time windows (TSPTW), which is an extension of traveling salesman problem with respect to temporal constraints. TSPTW can be defined formally on an undirected graph $G = (V, A)$, where V is the set of vertices and A is the set of edges. The vertex set has N vertices including a depot denoted by 0 and customers can be represented by $V\backslash\{0\}$. The set of edges A consists of pair of vertices, i.e. $A = \{(i, j): i, j \in V, i \neq j\}$. Each edge (i, j) is associated with a travel cost c_{ij} between vertices i and j. In TSPTW, one vehicle has to deliver goods through several customers. The objective of TSPTW is to find a Hamiltonian cycle of least cost, in which each customer can only be visited exactly once and the route must start and end at the depot. Furthermore, the time constraint restricts the time for visiting customer *i* within a given time window $[e_i, l_i]$, where e_i indicates the earliest service time and l_i is the latest service time. If the vehicle arrives at customer *i* earlier than e_i, the vehicle is required to wait till the earliest service time of customer *i*. Whenever the vehicle arrives at customer *i* later than the l_i, the tour is regarded as an infeasible solution. Savelsbergh (1985) had proved that the complexity of finding a feasible solution of TSPTW is NP-complete by transforming TSPTW to a 3-partition problem.

TSPTW has received more attention since 1980s though some earlier case studies had taken the temporal aspect into account. A great number of papers have been published on this problem. Several exact methods were applied to solve TSPTW to optimality, such as branch-and-bound and dynamic programming. Baker (1983) proposed a branch-and-bound algorithm to solve time-constrained traveling salesman problem, i.e. a less *lucid* name for TSPTW. In his branch-and-bound, a lower bound is obtained by relaxing absolute constraints of his time-constrained model. Dumas *et al*. (1995) applied a post elimination tests in the dynamic programming for TSPTW, and the elimination can reduce the state spaces successfully. Pesant *et al*. (1998) applied constraint programming embedded in a branch-and-bound algorithm, which has a depth search scheme with reduction of domain via constraint programming. The results showed that their method can solve the TSPTW more efficiently than Dumas *et al*. (1995) did.

TSPTW is a NP-Hard problem, and hence most research developed heuristics and meta-heuristics to deal with large size instances. Savelsbergh (1985) applied local search algorithms for TSPTW including *k*-opt, Or-opt, as well as an insertion heuristics to construct an initial solution. He mainly focused on reducing the computational complexity to check feasibility during local search, which reduced checking effort to $O(1)$. Because TSPTW is a subproblem of Vehicle Routing Problem with Time Windows (VRPTW), papers handled VRPTW have similar concept. Solomon (1987) also applied several heuristics with specified time windows restrictions, including nearest-neighbor, insertion, and sweep heuristics. Gendreau *et al*. (1998) proposed a generalized insertion heuristics (GENI), which inserts an unvisited node into a constructive tour by exchanging three inconsecutive edges. Calvo (2000) considered a two-phase heuristics. The first phase generates an initial solution based on assignment relaxation problem, and the second phase improves the solution via a two-level hierarchical local search. Ohlmann and Thomas (2007) performed a compressed-annealing heuristic for TSPTW, which handles infeasible solution by penalty function.

López-Ibáñez *et al*. (2009) and López-Ibáñez and Blum (2009) proposed a Beam-ACO algorithm for TSPTW, in which beam search is used to replace the construction solutions procedure in the framework of ACO. Beam-ACO constructs solutions based on the beam search and the pheromone matrix is still updated by ants. López-Ibáñez and Blum (2010) extended previous Beam-ACO via combining with local search, and a comprehensive comparison on test data from different papers. The result showed that Beam-ACO performed better than the compressed-annealing proposed by Ohlmann and Thomas (2007) in terms of robustness and low variability. Da Silva and Urrutia (2010) solved TSPTW by two phases, in which the first phase intended to find a feasible solution by VNS, and general VNS improved the feasible solution by VND in the second phase. Their approach successfully solved TSPTW with less effort in computational time and improving best known solution in some instances.

In this study, we propose a beam search (BS) algorithm for the Traveling Salesman Problem with Time Windows (TSPTW). BS was earliest developed in the field of artificial intelligent to solve the problems of speech recognition (Lowerre, 1976) and image understanding (Rubin, 1978). It was then applied in the field of combinatorial problem, e.g. Fox (1983) and Ow and Smith (1988) employed BS for scheduling problems. Ow and Morton (1988) later proposed a modified beam search, called filter beam search, to give a compromise between computational time and solution quality. In this paper, we proposed a different search structure of BS, which solved TSPTW based on an initial solution.

The remainder of this paper is organized as follows. In section 2, we describe the proposed beam search heuristic and the implementation steps of our proposed beam search heuristic. Computational results of tested instances are presented in section 3, followed by the conclusion in section 4.

2. THE BEAM SEARCH ALGORITHM

Beam search is an adopted heuristic method of the breadth-first search tree procedure, in which only most promising nodes are kept for further search. The key for beam search is to estimate which node has higher "possibility" to reach optimality. In the following subsections, we introduce the classic beam search first. The proposed beam search algorithm is then described, and their implementation details are provided.

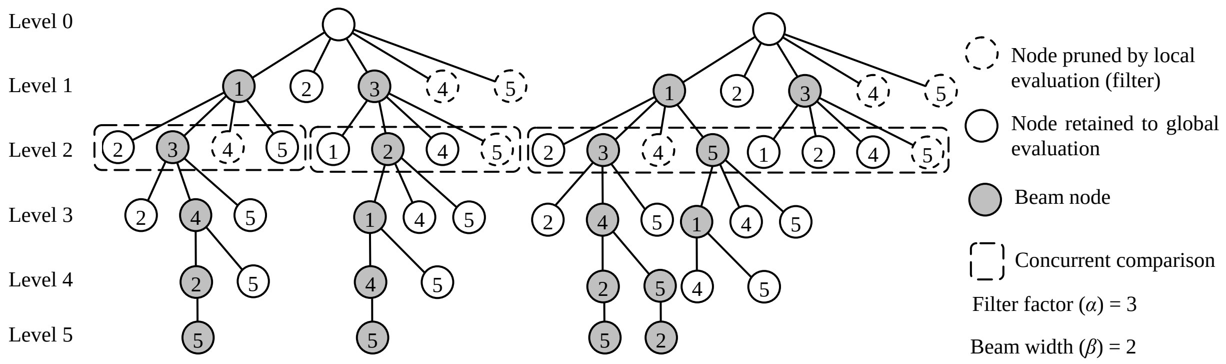
2.1 The Classic Beam Search

The beam search only explores few promising nodes via estimation; however the conventional search tree truncates nodes determinately by bounding constraint. The beam search follows breadth-first search scheme but does not involve any backtracking procedure. The nodes in the beam search are sprouted out level by level. In each level, only β nodes are retained for further search according to the evaluation of each node, where β is so-called *beam width*. It reduces a great deal of searching space via keeping only limited nodes. However, the optimality of solution can not be guaranteed by the beam search. The performance and effectiveness of a beam search algorithm greatly depends on the beam width and node evaluation process.

The evaluation for deciding promising nodes can be obtained by heuristics, mathematical models or estimation rules. Filter beam search proposed by Ow and Morton (1988), which contains two different types of evaluations: local evaluation functions and global evaluation functions. The first one only considers a one-step of the immediate branch cost for the beam node, and the latter one considers the total cost of the whole solution. The local evaluation is applied before global evaluation, because it takes lower computational complexity. In addition, it gives a rough evaluation like a filter to prune out some nodes with worse estimated cost. In this step, only α descendents of one beam node are retained, so there are totally $\alpha\beta$ nodes to be retained. Then, these nodes are evaluated by global evaluation, which is a caution estimation and time-consuming heuristic. Note that α could be less than β, because β is chosen from $\alpha\beta$ nodes retained from all descendents.

There are two types of beam search implementation in terms of branching procedure: dependent beam search and independent beam search. The independent beam search means that the search is independent between different beam nodes, an example for β =2 is shown in

Figure 1(a), in which one descendent of each beam node is chosen. The independent beam search compares estimation only within the descendents of the same beam node, instead of the comparison for those between different beam nodes. Therefore, the selection from descendents of a particular beam node would not be affected by descendents of other beam nodes. This way can guarantee that current β beam nodes only create β beam nodes for next level. On the other hand, the dependent beam search pools all the descendents of all beam nodes together, compares those descendent nodes and chooses β nodes with the best estimation to be the beam nodes for next level (see Figure 1(b)). In this case, the generated descendents at the same level need to be evaluated and considered together, and then these nodes must be compared to each other. In this paper, we apply dependent beam search which always chooses best β nodes from all generated nodes in the same level for further search.



(a) Independent beam search (b) Dependent beam search

Figure 1 Illustrations of filter beam search

2.2 The Beam Search for TSPTW

In this paper, we consider a dependent beam search with filter. However, unlike the conventional beam search applying a constructive heuristic for evaluation, the evaluation of our beam search is based on an initial *reference solution*. The root of tree in the beam search is always the depot. Then beam search first generates all descendants at first level, and evaluates those nodes by global evaluation. The evaluation estimates the travel cost and feasibility of each nodes based on an insertion heuristics to the reference solution. The best β nodes are kept as beam nodes to generate descendants in next level. If infeasible nodes exist, the node with less violation has higher priority to be selected. On the other hand, if there are more than needed β with the same evaluation value, the roulette wheel approach is performed to break ties.

The reference solution can be obtained simply by an arbitrary constructive heuristic. An initial reference solution is generated by adding each customer into an empty tour one by one. The unvisited customer has minimum e_i is selected first. The heuristic is terminated until all customers are added into the tour. This heuristic cannot guarantee the feasibility of the generated solution. However, the initial reference solution is not necessary to be feasible for our beam search, because the comparison between each evaluated nodes forces beam search to select a node with less feasibility violation.

The reference solution is improved by repeated insertions during the exploration of beam search. In the following subsections, the implementation details of the proposed beam search heuristic is provided, and the procedure is described as follows:

Step 1. Determine the filter width (α) and beam width (β), and obtain an initial reference solution.

Step 2. Generate all descendants at the first level, and set L=1.

Step 3. Evaluate each descendant by a global evaluation, which estimate the travel cost by repetitive insertions.

Step 4. Select the best descendants as beam nodes according to the evaluation value of each node and then follow search scheme of dependent beam search. If there are more than needed β nodes have same evaluation value, the beam nodes are selected by the roulette wheel approach to break ties.

Step 5. Generate descendants from the beam nodes and then filter out unnecessary descendants by local evaluation, and only α nodes have best evaluation value are kept.

Step 6. If $L<N$, then $L= L+1$ and go to step 7, otherwise go to step 8.

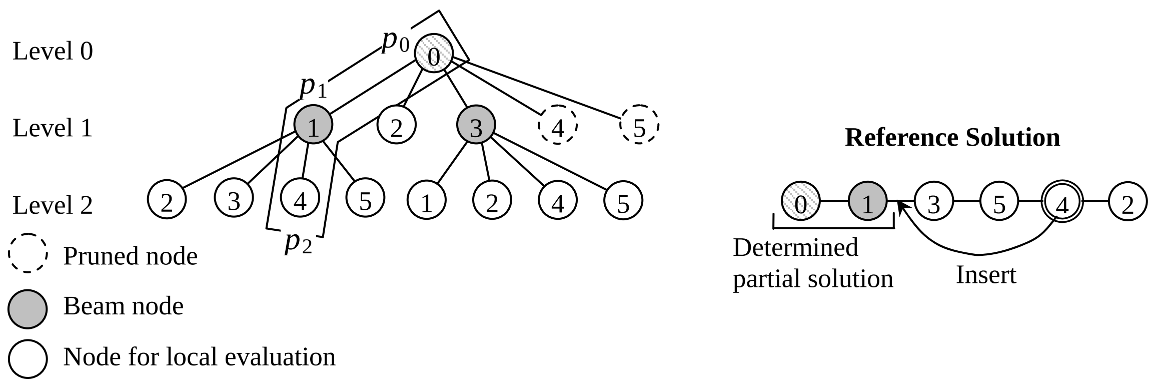
Step 7 Evaluate the kept descendants by a global evaluation, and go to step 4.

Step 8. Terminate the search procedure, and output the best solution in the beam nodes.

2.2.1 The Local Evaluation

The proposed beam search follows the concept of local search, where an initial solution is necessary to estimate the travel cost of each node. The initial solution is named *reference solution* hereafter. Whenever the beam search sprouts out nodes at level L, the partial solution till the level L are determined permanently due to forbiddance of backtrack. Given a reference solution $R = \{p_0, p_1, p_2, \ldots, p_N,\}$, a beam node at level L has a partial determined sequence $\{p_0, p_1, p_2, \ldots, p_{L-1}\}$, where p_i specifies the customer at order i of the current tour. The p_L is generated and determined according to beam node. The cost of p_L is then estimated by a local evaluation via inserting one customer from $\{p_L, p_{L+1}, \ldots, p_N\}$ to the level L. In the local evaluation, a one-step estimation for immediate cost is considered. Only one customer selected to level L is considered, and the remainder unvisited customers in reference solution would stay in place.

An example is shown in Figure 2. Two determined partial solutions of beam nodes are {0, 1} and {0, 3} and eight candidate nodes are generated by beam nodes. Given the reference solution R = {0, 1, 3, 5, 4, 2} and determined partial solution {0, 1}, the example shows that customer 4 is chosen to locate at level 2, then the estimation performed by inserting customer 4 into the 3rd place of reference solution as shown in Figure 2 (b). The evaluation is simply to calculate the travel cost and to verify the feasibility of the new reference solution R' = {0, 1, 4, 3, 5, 2}.



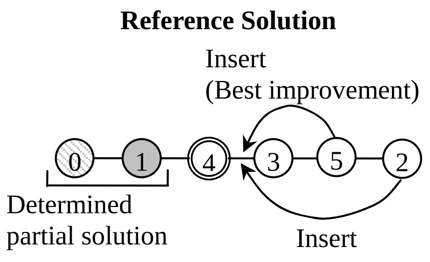
(a) A partial solution generated by a beam node (b) Evaluation by insertion

Figure 2 Illustrations of local evaluation

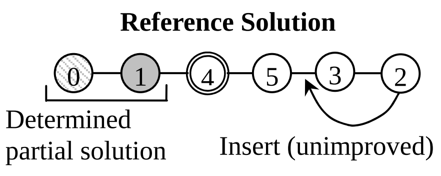
2.2.2 The Global Evaluation

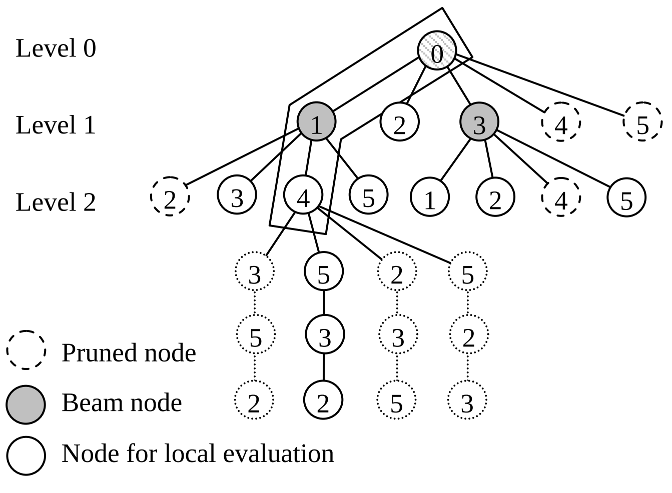
The global evaluation gives a better estimation of total travel cost, which modifies the order of the customers after p_L by repeated insertion. Only α nodes retained as well as the new reference solutions obtained by local evaluation are performed in the global evaluation. The global evaluation is first to insert each undetermined customer to the place L+1, and then choose the insertion with least cost to be the new reference solution. Next, according to the new reference solution, the remainder unvisited customers is inserted to the place L+2, and so on. Whenever the insertion gains a better reference solution, the current reference is updated by the better one. If no feasible position occurs during the insertion procedure, the current reference is updated by the one with less violation. The procedure terminates until every node from L+1 to N is processed, and the total travel cost of final reference solution is the final result for global evaluation.

Continue the example in Figure 2; Figure 3 illustrates the global evaluation procedure. Suppose three retained descendents of the first beam node are {0, 1, 3}, {0, 1, 4} and {0, 1, 5}, where the descendent {0, 1, 2} is filtered out in local evaluation step. Then, the global evaluation for the descendent {0, 1, 4} is carried out as follows. Given new reference solution R' = {0, 1, 4, 3, 5, 2}, the customer 5 and 2 are temporarily inserted into the 4th place of reference solution, respectively (see Figure 3 (a)). Suppose the travel cost evaluation shows that the customer 5 is inserted to 4th place to provide a better cost, then updating the reference solution to be R'' = {0, 1, 4, 5, 3, 2}. Next, try to insert the customer 2 into the 5th place, and evaluate the total cost again as shown in Figure 3 (b). Suppose the travel cost is not improved, then the best estimation for descendent node {0, 1, 4} by global evolution is the total travel cost of reference solution R'' = {0, 1, 4, 5, 3, 2}. Figure 3 (c) demonstrates that the global evaluation for node {0, 1, 4} actually scanned four different complete solutions, when the solution are temporarily created during insertion procedure. Finally, the best solution becomes new reference solution with the best estimated cost.



(a) Insertion to the 4th place of solution





(b) Insertion to the 5th place of solution (c) The solutions are explored by global evaluation

Figure 3 Illustrations of global evaluation

2.2.3 Implementation Details of Evaluation

According to time windows constraint, the time slot for visiting a customer includes three parts: travel time, waiting time and service time. The travel time and waiting time are varied depending on the solution sequence. If the feasibility is verified by straightforward computing the exact arrival time of each customer, its computational complexity would be $O(N)$. To reduce the computational complexity, we estimate travel time and departure time based on the idea of Savelsbergh (1985) and Solomon (1987) instead of working out accurate travel time. The following notations are used in further interpretation:

A_i	:	the arrival time at customer *i*.
B_i	:	the starting service time at customer *i*, and $B_i = max(A_i, e_i)$.
c_{ij}	:	the travel time between customers *i* and *j*.
D_i	:	the departure time at customer *i*, and $D_i = B_i + s_i$.
N	:	the total number of customers, including the depot.
p_i	:	the customer is visited at order *i*.
s_i	:	the service time at customer *i*.
W_i	:	the waiting time at customer *i*, $W_i = B_i - A_i$.

The local evaluation and the global evaluation are both performed by insertion. The insertion method switches one customer from an initial visiting order *k* to the forward order *L* by inserting the customer p_k after customer $p_{L\text{-}1}$. After insertion, the arrival time of segment between p_L and $p_{k\text{-}1}$ will shift backward, and the arrival time of segment between p_{k+1} and p_N may shift forward or backward (see Figure 4). If arrival time of any customer shifts forward and becomes later than its latest service time (l_i), then the solution turns out to be infeasible. Hence the shift time has to be taken into account to verify the feasibility. Let δ_1 indicate the estimated time shift of prior segment and δ_2 indicate the estimated time shift for latter segment. Then δ_1 and δ_2 can be obtained by equations (1) and (2), respectively.

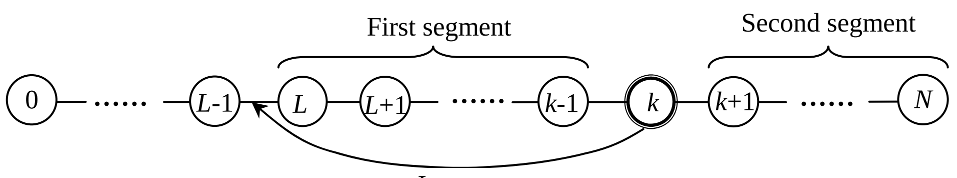


Figure 4 Illustration of time shift segments after insertion

$$\delta_1 = \max\{c_{P_{L-1},P_k},\ e_{P_k} - D_{P_{L-1}}\} + s_{P_k} + c_{P_k,P_L} - c_{P_{L-1},P_L} \tag{1}$$

$$\delta_2 = \delta_1 - \sum_{i=L}^{k-1} W_{p_i} + D_{P_{k\text{-}1}} + c_{P_{k\text{-}1},P_{k+1}} - A_{P_{k+1}} \tag{2}$$

To verify the feasibility, the shift in time needs to be compared with allowable time shift of those two segments. The allowable backward time shift of each customer is the maximum time difference that the arrival time can be postponed, and it is named *feasible forward shift* and denoted by τ_i hereafter. If shift in time is larger than the allowable time shift of any customer, the solution is regarded as infeasible. Given a reference solution, the feasible forward shift τ_i for a customer *i* can be estimated by equation (3).

$$\tau_i = \begin{cases} \sum_{j=L}^{i} W_{p_j} + \left(l_{p_i} - B_{p_i}\right) & \text{if } L \le i < k \\ \sum_{j=k+1}^{i} W_{p_j} + \left(l_{p_i} - B_{p_i}\right) & \text{if } k < i \end{cases} \tag{3}$$

The time shift δ_1 cannot exceed the feasible forward shift of any customer, hence an upper bound $\tau^1_{\min}$ of δ_1 can be estimated by equation (4) and an upper bound $\tau^2_{\min}$ of δ_2 can be estimated by equation (5) by taking minimum of all τ_i in those two segments.

$$\tau^1_{\min} = \min_{L \le i \le k-1} \{\tau_i\} \tag{4}$$

$$\tau^2_{\min} = \min_{k+1 \le i \le N} \{\tau_i\} \tag{5}$$

Then, equations (6) and (7) are applied to confirm the estimated feasibility after insertion.

$$\delta_1 \le \tau^1_{\min} \tag{6}$$

$$\delta_2 \le \tau^2_{\min} \tag{7}$$

The reduced travel time of an insertion can be computed by equation (8). The reference solution of current node is improved when Δd is greater than zero. If an infeasible solution is found, the maximum violation v_{max} among every customer is estimated by equation (9).

$$\Delta d = c_{P_{L-1}, P_L} + c_{P_{k-1}, P_k} + c_{P_k, P_{k+1}} - c_{P_{L-1}, P_k} - c_{P_k, P_L} - c_{P_{k-1}, P_{k+1}} \tag{8}$$

$$v_{\max} = \max\{\delta_1 - \tau^1_{\min},\ \delta_2 - \tau^2_{\min}\} \tag{9}$$

In the local evaluation, if travel times of all the generated nodes from the same beam node are computing aggregately, the summation in equation (2) can be counted accumulatively and computational complexity for each node is $O(1)$. According to similar aspect, the computational complexity for global evaluation of a filtered node is $O(N)$. The evaluation procedure is therefore able to be performed faster.

2.2.4 Implementation Details of Node Comparison

In order to compare the nodes, a penalty equals to the product of 10^6 and violation v_{max}. Let x_{ref} be the reference solution of a node, the evaluation value includes total travel time and penalty of maximum violation of the reference solution according to equation (10).

$$E(x_{ref}) = \sum_{i=0}^{N-1} c_{p_i, p_{i+1}} + c_{p_N, p_0} + 10^6 \cdot v_{\max} \tag{10}$$

Furthermore, the roulette wheel approach is performed to break ties. Let $f_L(\cdot)$ be the evaluation function for the travel time from depot to current level L. Then the modified evaluation function $Y(x_{ref})$ is shown in equation (11), which indicates the difference between the evaluation value and current travel time at level L. The normalized value for roulette wheel approach is shown as the equation (12) with respect to the parameter γ. The smaller the modified evaluation function value is, the larger the normalized value will be.

$$Y(x^i_{ref}) = E(x^i_{ref}) - f_L(x^i_{ref}) \tag{11}$$

$$Z_i = (Y_{\max} - Y(x^i_{ref}) + 1)^{\gamma} \tag{12}$$

where x^i_{ref} indicates the reference solution of the ith candidate node and Y_{max} represents the maximum evaluation value among all pooled nodes at the same level, i.e. Y_{max} =

$max\{Y(x^i_{ref})|\forall i\}$. The probability of a node to be selected as a beam node is determined by equation (13).

$$P_i = \frac{Z_i}{\sum_i Z_i} \tag{13}$$

The evaluation function shown in equations (11)-(13) indicate that a node with less cost in the future travel has higher probability to be selected.

3. NUMERICAL EXPERIENCE

The proposed beam search were coded in Borland C++ 6.0 and run on a PC with Intel Core 2 Due CPU E8400 3.0 GHz, RAM 1.99GB under Windows XP operations system. The beam search was tested in an instance set proposed by Dumas *et al.* (1995). The instance set based on symmetric Euclidean and the customers are distributed uniformly in the interval [0, 50] in the coordinate plane. Travel times between two customers equal to its distances. The time windows are generated randomly according to the service time of each customer in the TSP tour, in which a parameter *w* determines the range for time windows. We only selected 12 instances arbitrarily, where number of nodes ranges from 20 to 150 and *w* ranges from 20 to 60. The optimal travel distance of each instance is solved by Dumas *et al.* (1995). The selected instances are shown as Table 1.

Table 1 The selected test instances in Dumas *et al.* (1995)

instance	*N*	instance	*N*
n20w20.004	20	n80w20.003	80
n20w40.003	20	n80w20.005	80
n40w20.004	40	n100w20.003	100
n40w40.003	40	n100w40.002	100
n60w20.001	60	n150w20.003	150
n60w60.005	60	n150w20.005	150

We conducted a preliminary analysis to decide on filter factor α, beam width β and the parameter γ. The analysis results suggested the best parameters on $\alpha = 3$, $\beta = 9$ and $\gamma = 1.08$. In order to break tie for those nodes with the same estimation cost, the beam nodes are selected by roulette wheel approach. Because the roulette wheel approach is performed in a random manner, each instance was run five times to confirm its average performance. The computational results are presented in tables 2 by using the recommended parameters. The column headings in the table are defined as follows: instance name, the optimal travel cost, the gap of initial reference solution, the minimum gap, the average gap and average computational time in seconds. Statistics are averages over the five runs for each instance. The gap is computed as equation (14).

$$Gap = \frac{best - optimal}{optimal} \times 100\% \tag{14}$$

Table 2 shows the results obtained by the beam search. For each instance, the numbers in bold indicates that the best results of five runs achieves the optimal value. It can be observed that the initial reference solutions are infeasible except second and third instances, but our beam search still can be applied on those instances. Because the evaluation procedures can either reduce infeasibility or even get feasible solution by applying penalties. Though our beam search cannot guarantee to obtain feasible solution, the results show that all solutions of our testing instances can satisfy the time windows constraints. Moreover, eight out of twelve instances can obtain the optimal solution, while the largest gap is at 0.88% among the rest four instances. The average gaps are all within 1.68%. As expected, the CPU time increases with time window width and with problem size. The average computational time is less than one second except the largest size instances (*N* = 150).

Table 2 Results of our beam search

instance	**opt.**	**initial gap**	**min gap**	**avg. gap**	**avg. time (s)**
n20w20.004	396	0.51*	**0.00**	0.00	0.006
n20w40.003	317	9.78	**0.00**	0.00	0.006
n40w20.004	404	3.47	**0.00**	0.00	0.052
n40w40.003	474	25.53*	**0.00**	0.00	0.076
n60w20.001	551	6.72*	**0.00**	0.00	0.123
n60w60.005	569	38.49*	0.88	0.88	0.131
n80w20.003	667	9.00*	**0.00**	0.00	0.379
n80w20.005	748	5.21*	**0.00**	0.00	0.646
n100w20.003	762	11.42*	**0.00**	0.10	0.849
n100w40.002	653	34.92*	0.45	1.68	0.931
n150w20.003	834	12.47*	0.36	0.41	3.803
n150w20.005	846	25.53*	0.12	0.19	4.168
average		15.25	0.15	0.27	0.931

*infeasible solution

4. CONCLUSION

We have proposed a beam search heuristic for the traveling salesman problem (TSPTW) in this paper. Both local evaluation and global evaluation based on insertion are applied in this paper, in which an initial solution is needed. To reduce computational complexity, estimation functions for checking feasibility are used in evaluation step, where the feasibility of solutions can be checked with computational effort of O(1). Regardless of the feasibility of the initial solution, our beam search can either fix the infeasible solution or improve the solution. Our beam search is tested on 12 instances obtained from the literature. The results show that 8 instances can achieve optimal solution in small computational effort, and the gap of remainder instances is less than 1.0 percent. The results show that our beam search can obtain good solutions with effective computational times on TSPTW.

REFERENCES

Baker, E. K. (1983) An exact algorithm for the time-constrained traveling salesman problem, **Operations Res earch, Vol. 31, No5**, 938-945.

Calvo, R. W. (2000) New heuristic for the traveling salesman problem with time windows, **Transportation Science, Vol. 34, No. 1**, 113-124.

Da Silva, R. F. and Urrutia, S. (2010) A General VNS heuristic for the traveling salesman problem with time windows, **Discrete Optimization, Vol. 7, No. 4**, 203-211.

Dumas, Y., Desrosiers, J., Gelinas, E. and Solomon, M. M. (1995) An optimal algorithm for the traveling salesman problem with time windows, **Operations Research, Vol. 43, No. 2**, 367-71.

Fox, M. S. (1983) **Constraint-Directed Search: A Case Study of Job-Shop Scheduling**, Ph.D. Dissertation, Computer Science Department, Carnegie-Mellon University.

Gendreau, M., Hertz, A., Laporte, G. and Stan, M. (1998) A Generalized Insertion Heuristic for the Traveling Salesman Problem with Time Windows, **Operations Research, Vol. 43, No. 3**, 330-346.

Lowerre, B.T. (1976) **The HARPY Speech Recognition System**, Ph.D. Dissertation, Department of Computer Science, Carnegie-Mellon University at Pittsburgh, USA.

López-Ibáñez, M. and Blum, C. (2009) Beam-ACO based on stochastic sampling: A case study on the TSP with time windows, **Lecture Notes in Computer Science, Vol. 5851,** 59-73.

López-Ibáñez, M. and Blum, C. (2010) Beam-ACO for the traveling salesman problem with time windows, **Computers & Operations Research, Vol. 37, No. 9**, 1570-1583.

López-Ibáñez, M., Blum, C., Thiruvady, D., Ernst, A.T. and Meyer, B. (2009) Beam-ACO based on stochastic sampling for makespan optimization concerning the TSP with time windows, **Lecture Notes in Computer Science, Vol. 5482**, 97-108.

Ohlmann, J. W., and Thomas, B. W. (2007) A compressed-annealing approach to the travelling salesman problem with time windows, **INFORMS Journal on Computing, Vol. 19, No. 1**, 80-90.

Ow, P. S. and Morton, T. E. (1988) Filtered beam search in scheduling, **International Journal of Production Research, Vol. 26, No. 1**, 35-62.

Ow, P. S. and Smith, S. F. (1988) Viewing scheduling as an opportunistic problem-solving process. **Annals of Operations Research, Vol. 12, No. 1**, 85-108.

Pesant, G., Gendreau, M., Potvin, J-Y. and Rousseau, J-M. (1998) An exact constraint logic programming algorithm for the traveling salesman problem with time windows, **Transportation Science, Vol. 32, No. 1**, 12-28.

Rubin, S. (1978) **The ARGOS Image Understanding System**, Ph.D. Dissertation, Department of Computer Science, Carnegie-Mellon University at Pittsburgh, USA.

Savelsbergh, M. W. P. (1985) Local search in routing problems with time windows, **Annals of Operations Research, Vol. 4, No. 1**, 285-305.

Solomon, M. M. (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints, **Operations Research, Vol. 35, No. 2**, 254-265.