Two Time Window Discretization Methods for the Traveling Salesman Problem

With Time Window Constraints

Xiubin Wang
Transportation and Logistics Research Center and
Department of Business Administration and Economics
University of Wisconsin, Superior
XWang1@facstaff.uwsuper.edu

Amelia Regan
Institute of Transportation Studies and
Department of Civil & Environmental Engineering
University of California, Irvine, 92626
aregan@uci.edu

ABSTRACT

In this paper, we discuss the convergence of two time window discretization methods for the traveling salesman problem with time window constraints. These methods are general and can be applied to other vehicle routing and scheduling problems with time windows. The first method provides a feasible solution to the problem while the second provides a lower bound for the minimization problem. Analysis shows that as the discretization interval tends towards zero the lower bounding method converges to the optimal solution while the method used to develop feasible solutions does not. This result suggests that improving the feasible solution by using smaller partition, though useful in many cases, is not always so.

November 2002
INTRODUCTION

In this paper we examine the convergence of two time-window discretization methods developed for the traveling salesman problem with time windows (TSPTW). The problem is defined as follows. A single mobile server (traveling salesman) is required to visit a set of $N$ nodes on a tour constructed on a fully connected network. The mobile server begins with an origin node $o$ and returns to the same node in the end of the tour. There is a cost associated with travel between any node pair. Each node has a time window in which it must be visited. The objective is to visit all nodes within their time windows at minimum cost. Time window discretization methods for time-constrained routing and scheduling problems were first developed some time ago. See for example (1, 2, 3, 4, and 5). The general idea is to replace the continuous variables in a time-constrained routing and scheduling problem (time windows for visit) by a set of discrete variables corresponding to each visit at a particular time. In a recent paper, the authors revisited this method and provided a new discretization method for obtaining a lower bound which can be used to evaluate the quality of each feasible solution (6).

These methods have wide applicability in practice and are also of interest theoretically. Although discretization methods have been reported to perform well, there has been no attempt in the literature to discuss their convergence. It is commonly believed that when a smaller interval is used to discretize the time window a better solution will be obtained. As a result, it is often taken for granted that a solution found using an interval that tends towards an infinitesimal value converges to the optimal. In fact, it is not always the case. In this paper, we make the following three points:

1. The traditional discretization method does not guarantee convergence to the optimal solution.
2. In the traditional discretization method there is not a unique maximum partition to provide feasible solutions.
3. The new discretization method recently introduced to provide a lower bound guarantees convergence to the optimal solution.

The TSPTW problem underlies a wide variety of applications. Therefore, the results for this problem can be easily extended to other routing and scheduling problems, such as multiple TSPTW problems (m-TSPTW) and pickup and delivery problems with time windows (PDP-TW) (see for example 7).

Before we introduce the two methods we begin with the problem formulation.
Notation

Let:

\[ N = \] the set of nodes excluding the starting and ending nodes,

\[ o = \] the starting node for the traveling salesman,

\[ d = \] the ending node for the traveling salesman

\[ a_i, b_i = \] the beginning and end of the time window for node \( i \)

\[ t_{i,j} = \] the time needed to travel from node \( i \) to node \( j \)

\[ c_{i,j} = \] the cost of travel from node \( i \) to the origin of node \( j \)

\[ T_i = \] the visit time at node \( i \)

\[ x_{i,j} = \] binary decision variable

The flow variable \( x_{i,j} \) is equal to 1 when the traveling salesman visits node \( j \) after node \( i \); it is equal to 0, otherwise.

Formulation

\[
\text{obj min} \quad \sum_{i \in N \cup \{o\}} \sum_{j \in N \cup \{d\} \setminus \{i\}} c_{i,j} x_{i,j} \quad (1.0a)
\]

\[
\sum_{j \in N \cup \{d\}} x_{i,j} = 1 \quad \forall i \in N \cup \{o\} \quad (1.1a)
\]

\[
\sum_{i \in N \cup \{o\}} x_{i,j} = 1 \quad \forall j \in N \cup \{d\} \quad (1.2a)
\]

\[
(T_j - T_i - t_{i,j}) x_{i,j} \geq 0 \quad \forall i \in N \cup \{o\}, j \in N \cup \{d\} \quad (1.3a)
\]

\[
a_i \leq T_i \leq b_i \quad \forall i \in N \cup \{d\} \cup \{o\} \quad (1.4a)
\]

\( x_{i,j} \) is binary; for all \( i \in N \cup \{o\}, j \in N \cup \{d\}, i \neq j \) \quad (1.5a)

Constraints (1.1a) require the mobile server to leave each node once. Constraints (1.2a) dictate that each node be entered once. Constraints (1.3a) enforce the temporal relationship of consecutive nodes. Constraints (1.4a) specify the time window requirement while constraints (1.5a) are the binary constraints.
**TIME WINDOW DISCRETIZATION**

Please note that a preprocessing step typically used for vehicle routing problems with time windows may be necessary. There may be some part of the time window, which is of no use to the assignment since the traveling salesman is not able to reach the node within that period. That part of the window is eliminated in order to reduce the size of the problem. Desrochers et al describe this method in their 1995 survey of this topic (8). From here on, we assume that the time windows have already been reduced according to that method.

Then we partition the time window into a set of smaller sub windows with each sub-window corresponding to a sub node. We require that only one sub node of each original node be visited. Though the widths of the sub windows are not required to be uniform, to facilitate understanding in this paper, we specifically suggest one partitioning scheme as in (6). Suppose the pre-selected width for partitioning is $L$, and that the node has a window $[a_i, b_i]$. First, determine the number of sub-nodes for this iteration by taking the smallest integer that is greater than $(b_i-a_i)/L$ where $\text{ceil}(x)$ is the ceiling function that finds the smallest integer greater than $x$. Then partition the window into this many parts evenly. In addition, create one additional sub node for which the window is reduced to the starting time point of the original time window. As is easy to examine, the result obtained through this specific partitioning scheme can be easily extended to other schemes.

Then the formulation after time window partitioning can be modified as follows:

\[
\text{obj Min } \sum_{i \in \omega \cup \{o\}} \sum_{j \in \omega \cup \{d\}, \delta(i) \neq \delta(j)} c_{i,j} x_{i,j} \tag{1.0b}
\]

\[
\sum_{j \in \omega \cup \{d\}, \delta(j) \neq \delta(i)} x_{i,j} = 1 \quad \forall i \in \{o\} \cup \omega \tag{1.1b}
\]

\[
\sum_{i \in \omega \cup \{o\}, \delta(i) \neq \delta(j)} x_{i,j} = 1 \quad \forall j \in \omega \cup \{d\} \tag{1.2b}
\]

\[
\sum_{j \in \omega \cup \{d\}, \delta(j) = \xi} \sum_{i \in \omega \cup \{o\}, \delta(i) \neq \delta(j)} x_{i,j} = 1 \quad \forall \xi \in N \tag{1.2b'}
\]

\[
\sum_{i \in \omega \cup \{o\}, \delta(i) \neq \delta(j)} x_{i,j} - \sum_{m \in \omega \cup \{d\}, \delta(m) \neq \delta(j)} x_{j,m} = 0 \quad \forall j \in \omega \tag{1.3b}
\]

\[
x_{i,j}(T_i + t_{i,j} - T_j) \leq 0 \quad \forall i \in \omega, j \in \omega, \delta(i) \neq \delta(j) \tag{1.4b}
\]

\[
a_i^j \leq T_i \leq b_i^j \quad \forall i \in \omega \tag{1.5b}
\]

\[
x_{i,j} \text{ is binary } \forall i \in \omega \cup \{o\}, j \in \omega \cup \{d\}, i \neq j \tag{1.6b}
\]
where $\delta(i)$ denotes the node associated with sub-node $i$. $\omega$ is the set of all sub-nodes. Constraints (1.2'b) are coupling constraints that stipulate that exactly one sub-node associated each node will be served – these are a direct result of the discretization of the problem. In addition, $a_i'$ and $b_i'$ now represent the beginning and end of the sub window $i$.

**OVER-CONSTRAINED AND UNDER-CONSTRAINED METHODS**

In this section, we introduce the discretization methods used in (6) where the over-constrained method corresponds to the traditional discretization method which leads to a feasible solution and the under-constrained method provides a lower bound.

We first set up the network with links between sub nodes of different nodes. There are two ways, one over-constrained and the other under-constrained, to build the links. If we consider only the end points of the sub windows when we set up links between two sub nodes for the network, we obtain a network that ignores many feasible links. We refer to this method as the over-constrained method. If we consider the starting point of the first sub window and end point of the second sub window when we set up the link between two sub-nodes, then we end up with a network that includes some infeasible links. We refer to this method as the under-constrained method. The over-constrained method leads to a network from which we obtain a feasible solution while the under constrained method provides a lower bound for the solution. Please note that the over constrained method is just the conventional time window discretization method.

Figures 1 and 2 show the feasible links excluded and infeasible links included in the two methods.

As seen in Figure 1, it is not possible to reach node 1 after leaving at the latest time in the time window for node 2 or to reach node 2 after leaving at the latest point in the time window for node 1. However, as is shown above, it is possible to serve node 1 after serving node 2 in the early part of the time window for node 2.

Observe that the under constrained network might permit infeasible links. If the traveling salesman leaves node 1 at the end of its time window it will have no way to reach node 2 within its time window. In fact, this network may contain cycles.

In the over constrained method we replace constraints (1.4b) and (1.5b) with the following:

$$x_{i,j}(b_i' + t_{i,j} - b_j') \leq 0$$  \hspace{1cm} (1.7b)

While in the under constrained method we replace constraints (1.4a) and (1.5a) with the following:

$$x_{i,j}(a_i' + t_{i,j} - b_j') \leq 0$$  \hspace{1cm} (1.8b)
Constraints (1.7b) define the links for the network of over constrained method. Constraints (1.8b) define links of network from under constrained method. However the network generated using the under constrained method is likely to contain cycles as can be seen in figure 2.

Observation

Every solution from the over-constrained method is feasible to the original problem while the under-constrained method contains all the feasible solutions to the original problem.

Observe that the optimal solution $Z_{1}^{IP}$ to the linear relaxation of the under constrained formulation, the integer solution $Z_{1}^{IP}$ to the under constrained method, the solution $Z_{m}^{IP}$ to the over constrained formulation and the global optimal integer solution $Z_{o}^{IP}$ can be placed in the following order:

$$Z_{1}^{LP} \leq Z_{1}^{IP} \leq Z_{o}^{IP} \leq Z_{m}^{IP}$$

In general, the bigger the time windows, the bigger the gap between $Z_{1}^{IP}$ and $Z_{o}^{IP}$, and between $Z_{o}^{IP}$ and $Z_{m}^{IP}$. Under constrained method provides a way to evaluate the solution quality of the feasible solution from over constrained method. It is generally believed that a partitioning width small enough leads to optimal solution. This is true in some cases and false in others. The following provides a discussion about the convergence of both the over-constrained and under-constrained methods. Analysis shows that under constrained method leads to the optimal solution when partitioning width tends to zero.

CONVERGENCE

In this section we analyze the convergence of the methods.

Proposition I

In the over-constrained method, there is a particular partitioning width that guarantees the optimal solution.

Proof

Two cases must be considered. Case I: Suppose there are $n$ nodes in the optimal assignment, out of which there are $m$ nodes served at a time point different from the starting or ending points of their windows. Please note that the starting or ending point of the time window are also the starting or ending point of some corresponding sub-window according to the partitioning method adopted. As a result, for the $(n-m)$ nodes for which the optimal visit time is the beginning or ending of their time windows, these optimal
points are always considered in the solution space defined by over-constrained method. Therefore, the partitioning scheme needs only to select the right partitioning width in order to have the optimal visit times for the other $m$ nodes considered in the over-constrained method. An optimal visit time is considered in the solution when it coincides with the end point of a sub-window. We note that each of these $m$ nodes is divided into two parts, called divisions, by the optimal visit time at this node. There must exist a partitioning width applicable to all the windows such that each division is in an integer multiple of this width. By partitioning with this width, the optimal visit times are guaranteed to coincide with the end points of the sub-windows and therefore be considered for an assignment. As a result, the optimal solution is guaranteed.

In fact, this width can be easily obtained. Let us suppose division $i$ has a length of $\frac{m_i}{n_i}$, where $m_i, n_i$ are positive integers with no common factors and $i$ is the index of all the divisions. (Again, we assume that the lengths of all time windows are rational numbers.) A partition that guarantees the optimal solution is $\frac{\text{com}(m_i)}{\prod n_i}$, where $\text{com}(m_i)$ is the maximum common factor of the integer values $m_i$ and the denominator is an integer that equals to the product of all $n_i$.

Therefore, any partition that is $1/k$ times the value obtained above, where $k$ is an integer greater than zero, guarantees that the over-constrained method contains the optimal solution.

Case II: If the optimal assignment serves all nodes at the end points of the time windows, any partition leads to an optimal solution in the over-constrained method.

(End of proof)

Typically small partitioning width leads to large number of sub windows and therefore an exploded network. Of interest is whether there is a maximum partition in order to guarantee an optimal solution in the over-constrained method.

**Observation**

Under the partitioning scheme specified above, there is not a maximum partitioning width to guarantee optimal solution by over-constrained method.

We show this observation by example. In the following example as in figure 3, an optimal solution requires the traveling salesman to serve nodes 1 and 2 and then return to the depot. Both windows have unit length. In this example, there is a unique optimal solution. The optimal arrival time at node 1 is at its one-third point of the window while the arrival time at the second node is at its last point of the window. Under the
partitioning scheme, any partition within the interval \([1/3,1/2)\) leads to the optimal assignment since all these widths lead to three partitions of the first window. In this example, there is no maximum partitioning width for the optimal solution though an upper limit exists to this optimal partitioning width, which equals to \(1/2\) unit in this example.

We observe from intuition that the over constrained method results in loss of available time by equivalently forcing the mobile server to wait until the last time point in each sub window before starting out. Therefore, the total sum of lost time affects the optimality of the solution. By reducing the partitioning width to an infinitesimal value, the discrete problem approaches the continuous one and sum of the lost time tends to be zero. Therefore, the feasible solution tends intuitively towards optimality. However, the optimal solution is not always guaranteed in the over constrained method even by those partitions approaching infinitesimal widths. Here we provide a conclusion as follows.

**Proposition II**

Not all partitioning methods approaching infinitesimal values guarantee an optimal solution in the over constrained method.

**Proof**

Here we identify two possible cases. In the first case the optimal visit time for each of the nodes is unique as shown by the example in figure 4, where the widths of the windows are one time unit. The optimal visit time at node 1 is assumed to be exactly the middle of the window. The mobile server is not able to arrive at node 1 before the midpoint of its window. In this case, only a partition of \(1/k\) times of \(1/2\) unit with \(k\) being an integer greater than zero guarantees optimal solution. Otherwise, any lost time resulting from under constrained method, regardless how small it is, causes the optimal solution to be missed. For example, a partitioning series \(\{3^{-n}, n \to +\infty\}\), tends towards an infinitesimal width, but does not guarantee the optimal solution since at every partition in this series, the optimal visit time at node 1 is lost and hence cannot serve node 2 at its latest time point.

The second case is that the optimal schedule can be shifted, which means that there is not one unique optimal visit time at each node. In this case, there are an infinite number of optimal schedules, which forms a time span \([S_i, E_i]\) for each of the window. Now we prove that by selecting a partition sufficiently small, an optimal solution is always obtainable in this case by the over constrained method. We define a fraction of the window, \([a_i, b_i]\), for each node \(i\) such that \(a_i = b_{i-1} + t_{i-1}\) and \(S_i < a_i < b_i < E_i\). The windows \([a_i, b_i]\) are shown in Fig. 6. We can easily see that any schedule is feasible and optimal if the visit time of the node falls within the small window \([a_i, b_i]\) of that node.
Next we demonstrate that a partition sufficiently small can guarantee a visiting time considered in each of the small window \([a_i, b_i]\). Select a partition that is less than or equal to the minimum of the widths for the windows \([a_i, b_i]\). By partitioning with this width, we can always guarantee for each of the nodes a sub window whose last time point falls within the window \([a_i, b_i]\) since each window \([a_i, b_i]\) is wider than the partitioning width. Therefore, by selecting a partitioning width sufficiently small, the optimal solution is always guaranteed.

In contrast to the convergence property of the over-constrained method, the under constrained method clearly shows convergence.

(End of proof)

**Proposition III**

Solutions obtained by the under-constrained method converge to the optimal integer solution if the partitioning width tends towards an infinitesimal value.

**Proof**

Please note that the solution space defined by under constrained method includes all the feasible solutions to the original problem. However, the under constrained method might lead to infeasible solutions. Any infeasible solution can be excluded by adopting a partitioning width smaller enough. Therefore when the partitioning width tends towards an infinitesimal value, the solution space defined by the under constrained method converges to the solution space of the original problem.

Suppose an assignment resulting from under-constrained method is node1-node2…-node \(n\). Suppose that this assignment first becomes infeasible since \(m^{th}\) node after examination, where \(m \leq n\). In other words,

\[ T_{i-1} + t_{i-1,i} \leq b_i \forall i, i < m \]

Note that \(b_i\) is the end point of the time window for node \(i\). \(T_i\) is the earliest visit time at node \(i\) according to the assignment from the under-constrained method. \(t_{i,j}\) is the travel time from node \(i\) to node \(j\). Each \(T_i\) must satisfy the following,

\[ T_i \geq T_{i-1} + t_{i-1,j} \]

\[ a_i \leq T_i \leq b_i \]

With the \(m^{th}\) node, the infeasibility can be expressed in the following way,

\[ T_m = T_{m-1} + t_{m-1,m} \quad \text{and} \quad T_m > b_m \]
Obviously there is a gap between the actual arrival time and the latest allowable visit time at node \( m \). Now let us provide some explanation why this gap is ignored in the under-constrained method.

First assume that the sub-node used in the under-constrained method that leads to this assignment is \( N_i \) associated with node \( i \). Each of the sub-nodes has a window \([a_i, b_i]\) with a width \( \theta_i \). Let \( D_i = T_i^r - a_i \), where \( T_i^r \) is the arrival time at sub node \( N_i \) from the starting point of the sub window \( N_{i-1} \). Please note that if \( T_i^r \) coincides with a dividing point in the window, the sub-node whose lasted point is \( T_i^r \) is used to obtain \( D_i \). The following conclusion follows. An assignment infeasible at node \( m \) exists if and only if the following inequality holds,

\[
\sum_{i=1}^{m-1} D_i \geq T_m - b_m
\]  

(3)

Following this conclusion, we can have a way to get rid of an infeasible assignment. It is self-evident that \( \theta_i \geq D_i \), then we have \( \sum_{i=1}^{m} \theta_i \geq \sum_{i=1}^{m} D_i \). By adopting sufficiently smaller partitioning width, the value of \( \sum_{i=1}^{m} \theta_i \) can be reduced to a level as follows,

\[
\sum_{i=1}^{m-1} \theta_i \leq T_m - b_m
\]  

(4)

such that the inequality \( (4) \) does not hold. Any infeasible assignment after the \( m^{th} \) node can be eliminated in the similar way. In conclusion, by adopting a partitioning width smaller enough, we can eliminate any infeasible assignment. Therefore, a series of partitioning width leading to infinitely small values have solutions convergent to the optimal.

(End of proof)

This point is illustrated in figure 7 where assignment node1-node2-node3 is infeasible at node 3, missing the deadline by a gap \( g \). The assignment is obtained when the three shadowed sub-nodes are considered. The assignment is infeasible because when a link is added between two sub-nodes in under constrained method, the server is equivalently empowered to move backward in time by a small value \( D_i \) (corresponding to subnode \( N_i \)). The infeasible assignment is made only when the sum of the \( D_i \) before the infeasible node is large enough. By reducing the partitioning width \( \theta_i \), this infeasible assignment can be excluded.

From the analysis above, it can be seen for a given partitioning series, even when the series tends infinitely small, the over constrained method does not guarantee convergence
to the optimal solution while under-constrained method does. However, our intuition is that a smaller partitioning width typically contributes significantly to the improvement in optimality in the over constrained method. The reason is that many problems encountered in practice have schedules that can be shifted. In these cases, a sufficiently small partitioning width always leads to an optimal solution under over constrained method. Even in the case where the optimal schedule cannot be shifted, our argument is that smaller width helps to improve solution quality by improving partial solution where the partial schedule can be shifted. This might be the reason this method works well in practice.

CONCLUSION

In this paper we discuss the convergence property of two time window discretization methods for the general traveling salesman problem with time window constraints. Two discretization methods are studied. The over constrained method provides a feasible solution while the under constrained method provides a lower bound. Examination shows that the over constrained method does not converge to the optimal solution in some cases while the under constrained method always does when the partition tends towards an infinitesimal value. We also identify condition under which an infinitely small partitioning width guarantees optimal solution to the original problem under over constrained method. However, even in the case where over constrained method does not converge to the optimal solution, by adopting a smaller partitioning width, it could help improve the solution quality as in many applications because it might help improve partial solution where partial schedule can be shifted. We also show that there is not a maximum partition for the optimal solution in the over constrained method. Though only one partitioning scheme is examined, the results here easily extend to other schemes.

However, the over constrained method always provides a feasible solution while the under constrained method must continue execution before it converges to the optimal solution. Therefore the over constrained method is in general more practical to implement. However, under constrained method may provide a tight bound to the over constrained method since it converges to the optimal solution when the partitioning width tends infinitesimally small.

ACKNOWLEDGEMENT

This research was partially supported by the University of California Transportation Center (UCTC) and by the US National Science Foundation (NSF) under Grant No. CMS-9875675. The authors gratefully acknowledge this generous support. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the UCTC or the NSF.
REFERENCES


List of Figures

Figure 1) Over-constrained network

Figure 2) Under-constrained network

Figure 3) A unique optimal solution

Figure 4) The optimal assignment with unique schedule

Figure 5) The optimal assignment with a schedule that can be shifted

Figure 6) The optimal and latest possible schedules

Figure 7) Infeasible assignment from the under constrained method
Figure 1) Over-constrained network
Figure 2) Under-constrained network
Figure 3) A unique optimal solution
Figure 4) The optimal assignment with unique schedule
Figure 5) Optimal assignment with a schedule that can be shifted
Figure 6) The optimal and latest possible schedules
Figure 7) Infeasible assignment from under constrained method