Efficient Feasibility Testing for Dial-a-Ride Problems

Brady Hunsaker
Martin Savelsbergh

Department of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0205

Abstract

Dial-a-Ride systems involve dispatching a vehicle to satisfy demands from a set of customers who call a vehicle operating agency requesting that an item be picked up from a specific location and delivered to a specific destination. Dial-a-ride problems differ from other routing and scheduling problems in that they typically involve service related constraints. It is common to have maximum wait time constraints and maximum ride time constraints. In the presence of maximum wait time and maximum ride time restrictions, it is not clear how to efficiently determine, given a sequence of pickups and deliveries, whether a feasible schedule exists. We demonstrate that this, in fact, can be done in linear time.

1 Introduction

Dial-a-Ride systems involve dispatching a vehicle to satisfy demands from a set of customers who call a vehicle operating agency requesting that an item be picked up from a specific location and delivered to a specific destination. Such demand-responsive transportation systems have been in operation in several metropolitan areas. The Dial-a-Ride problem arises in several practical applications such as the transportation of elderly and disabled persons, shared taxi services and courier services.

Dial-a-ride systems, and more general pickup and delivery problems, have been studied extensively in the literature, see for example, Stein [10], Psaraftis [3, 4, 5, 6, 7], Sexton and Bodin [8, 9] Desrosiers et al. [1], Jaw et al. [2].

In the typical dial-a-ride problem, a taxi company has the task of transporting customers from their origin to their destination. Different customers may share the ride during their journey, but at any given time, the taxi may carry at most a limited number of passengers. The goal is to minimize the total distance traveled by the taxi in transporting all the passengers. Dial-a-ride problems differ from other routing and scheduling problems in that they typically involve service related constraints. It is common to have maximum wait time constraints, which limit the waiting time at a stop before departing, and maximum ride time constraints, which limit the time between pickup and delivery. Such constraints significantly complicate the construction of high quality feasible schedules. In the literature, these complicating constraints are usually relaxed.

Practical problems are mostly solved heuristically either by construction-based algorithms or improvement-based algorithms. Many of these heuristics proceed by starting with a vehicle route and performing insertions either to improve the route or to add a customer to the route. For such methods, it is key that the feasibility of an insertion can be verified efficiently. Note that inserting a customer adds two new stops to a route - one corresponding to the pickup, the other corresponding to the delivery. In the presence of maximum wait time and maximum ride time restrictions, it is not obvious how to verify the feasibility of an insertion and it is even less clear how to accomplish this quickly. As mentioned before, existing heuristics avoid this issue by making the simplifying assumptions that vehicles are not allowed to wait while customers are on board, and that there are no constraints on the maximum ride time permitted.

In this technical note, we demonstrate that it is possible to verify the feasibility of a given route, even in the presence of maximum wait time and maximum ride time constraints, in linear time.

2 Problem Definition

We model the dial-a-ride problem as follows. We have a number of packages $i \in I$, each with a pickup location i^+ with time window $[e_{i^+}, \ell_{i^+}]$, delivery location i^- with time window $[e_{i^-}, \ell_{i^-}]$, and size d_i. We must start all routes from a central facility with location 0 during the time interval $[e_0, \ell_0]$. Let $J = \{i^+, i^- : i \in I\} \cup \{0\}$ be the set of all points. The capacity of a vehicle is Q, and the maximum allowed waiting time at any point is W. Let t_{i^+,i^-} be the time to drive from i^+ to i^- directly. The maximum ride time constraint states that the time between pickup of i and delivery of i must be no more than $\alpha t_{i^+,i^-}$, where $\alpha > 1$ is a constant. To summarize, we have four types of feasibility constraints:

capacity At all times, $\sum_{i \text{ in vehicle}} d_i \leq Q$.

time windows Each pickup and delivery must occur within its time window.

waiting time At no stop may a vehicle wait more than W minutes before departing.

drive time The time from pickup to delivery of package i must be no greater than $\alpha t_{i^+,i^-}$.

3 Verifying Route Feasibility

Let $|I| = n$, so that there are $2n + 1$ stops on the route including the initial location. We will show that we can check feasibility in $O(n)$ time.

To check capacity, we can step through the route, maintaining the current load q_j at each point j. Updating q_j at each point takes constant time using the formulas $q_{j+1} = q_j + d_i$ if $j = i^+$ and $q_{j+1} = q_j - d_i$ if $j = i^-$. Then we simply check that $q_j \leq Q$. Since there are $O(n)$ stops, the total time to check is $O(n)$.

To check the remaining scheduling constraints, we will step through the route several times, each time computing arrival and departure times at each stop. We will maintain the property that no feasible solution can have an arrival or departure time earlier than the computed arrival or departure time in the current solution.

Let the arrival and departure time at stop j be given by A_j and D_j, respectively. In the first pass, we will account for the pickup and delivery time windows and the maximum waiting time constraints. We start with $D_0 = e_0$, which clearly satisfies the property that no feasible solution can have an earlier departure time at 0. Next, we iteratively compute arrivals $A_{j+1} = D_j + t_{j,j+1}$ and departures $D_{j+1} = \max\{A_{j+1}, e_{j+1}\}$. Again, these computations clearly maintain our earliest feasible time property.

In addition, we keep track of the latest possible departure time at each stop such that time window constraints and waiting time constraints are satisfied for the current stop and all earlier stops in the route. Denote this value as L_j for stop j. Computing L_j can be done iteratively as $L_0 = \ell_0$ and $L_j = \min\{L_{j-1} + t_{j-1,j} + W, \ell_j\}$.

On this first pass, if we ever find that $A_j > \ell_j$, then we are arriving too late and the route is infeasible. The proof of this is easy: by construction, no feasible solution can arrive at stop j earlier than A_j, yet by definition no feasible solution can arrive at stop j later than ℓ_j.

In addition, if we ever have $A_j + W < e_j$, then we must wait at this stop too long, and the route *might* be infeasible. The reason that it may still be feasible, however, is that we may be able to transfer some of the waiting to earlier stops on the route, which is why we kept track of L_j. In fact L_{j-1} is the latest possible departure time at stop point $j - 1$ that maintains time window and waiting time feasibility on all earlier stops. Thus, the route is still feasible if and only if $L_{j-1} + t_{j-1,j} + W \geq e_j$. In the event that the route is still feasible, we set $A_j = e_j - W$ and $D_j = e_j$. Note that we chose the earliest time for A_j possible in order to maintain the earliest feasible time property. We can then proceed to the next stop.

This first pass through the tour checks completely for time windows and waiting time, and it requires a constant number of computations and comparisons at each stop. The time complexity so far is $O(n)$.

Note that the arrival and departure times computed in the first pass may not represent a feasible solution for two reasons. First, if we ever had to spread waiting time backwards using an L_j value, then all arrival

and departure times prior to stop j need to be recomputed (though they will be feasible). Second, we have not taken maximum ride time constraints into account yet. In order to do this we now pass through the stops in reverse order, correcting arrivals and departures and checking ride time constraints as we go.

Specifically, we start with the final arrival time A_{2n}, which is necessarily correct. Then we work backwards by computing $D_{j-1} = A_j - t_{j-1,j}$ for each stop. This may be different from our previous value for D_{j-1}. We need to keep A_{j-1} as close as possible to the previous value, however, in order to maintain the earliest feasible time property. Since the longest we can wait at a stop is W, we set $A_{j-1} = \max\{A_{j-1}, D_{j-1} - W\}$. In addition, we keep track of the total waiting time $\hat{W}$ from j to the end. This value is initialized as $\hat{W} = 0$ and updated as $\hat{W} = \hat{W} + (D_j - A_j)$.

As we are computing these values, we also check ride time constraints. Whenever we are at a pickup point i^+, we can check the ride time for package i, which is $D_{i^-} - D_{i^+}$. If this is less than or equal to the maximum ride time $\alpha t_{i^+,i^-}$, then we continue in our backwards pass. If $D_{i^-} - D_{i^+} > \alpha t_{i^+,i^-}$, however, then the route appears infeasible, but it may still be feasible. The one chance for feasibility is that there is enough waiting time on the route between stops i^+ and i^- that we can shift to stops before i^+. This would reduce the ride time for package i and may achieve feasibility.

By the earliest feasible time property, we cannot adjust D_{i^-} to an earlier time, so our one hope is to adjust D_{i^+} to a later time. The amount by which we need to adjust it if we hope to achieve feasibility is $\Delta \equiv (D_{i^-} - D_{i^+}) - \alpha t_{i^+,i^-}$. Before checking if there is enough waiting time to shift, we must first check that $D_{i^+} + \Delta \leq L_{i^+}$. If this is not the case, then we cannot introduce the necessary delay at i^+ and still maintain feasibility at earlier stops, so the route is infeasible. If it is the case that $D_{i^+} + \Delta \leq L_{i^+}$, then we still need to check that we have enough waiting time. To do this, we could sum all the waiting time between i^+ and i^-, but doing so would result in $O(n^2)$ worst-case time. To avoid this, we note that the waiting time between i^+ and i^- is no more than $\hat{W}$. Consequently, if $\Delta > \hat{W}$, then the route is infeasible.

If $\Delta \leq \hat{W}$, then the route may be feasible. The earliest feasible departure time at i^+ is $D_{i^+} = D_{i^+} + \Delta$. We have "used" Δ of the waiting time to make this change, however, so we must also update $\hat{W} = \hat{W} - \Delta$. We then continue our backwards pass as before.

If this backwards pass through the route terminates without a declaration of infeasibility, then we must confirm that our attempts to repair drive times were valid. There are two reasons that this may not be so: there may not have been enough waiting time between a pickup and delivery (remember that we just used $\hat{W}$ as an upper bound on the true value), or the delay in a departure may cause a time window to be violated at a later stop.

We can check these possibilities with a final pass through the route in the forward direction. In this pass, we start with D_0 and update $A_j = D_{j-1} + t_{j-1,j}$ and $D_j = \max\{D_j, A_j\}$. If we ever find $D_j > \ell_j$, then the route is infeasible since D_j satisfies the earliest feasible time property.

In addition, when we are at a delivery point i^-, we check that $D_{i^-} - D_{i^+} \leq \alpha t_{i^+,i^-}$. If this is true, then the drive time constraint for package i is satisfied and we continue. If the inequality is not satisfied, then the route is infeasible. To prove this, note that in the backwards pass D_{i^+} was adjusted so that the drive time constraint would be satisfied if D_{i^-} remained the same. So the fact that the constraint is not satisfied means that D_{i^-} is not the same as it was during the last pass. This means that *no* departure time D_j between i^+ and i^- remained the same as it was during the last pass, since that would have put us back "on track" to reach the old value of D_{i^-}. Since the new departure times were computed as $D_j = \max\{D_j, A_j\}$, this means that $D_j = A_j$ for all stops between i^+ and i^-. Thus, there is no waiting time between i^+ and i^-, and the drive time constraint is violated in a way that cannot be fixed. This proves the route is infeasible.

If the final pass through the route finishes without declaring infeasibility, then the values A_j and D_j give a feasible solution by construction. In fact, they give the earliest feasible solution! In each of the three passes, only a constant number of computations and comparisons were done at each stop, so the total time is $O(n)$.

Therefore, we have shown that we can check a given route for feasibility in $O(n)$ time (though we do require $O(n)$ space to store the values of A_j, D_j, and L_j).

The algorithm is compactly stated below.

1. First pass
   (a) Set $q = 0$, $D_0 = e_0$, $L_0 = \ell_0$, and $j = 1$.
   (b) If $j = i^+$, set $q = q + d_i$. If $j = i^-$, set $q = q - d_i$.
   (c) If $q > Q$, then infeasible. Stop.
   (d) Set $L_j = \min\{L_{j-1} + t_{j-1,j} + W, \ell_j\}$.
   (e) Set $A_j = D_{j-1} + t_{j-1,j}$ and $D_j = \max\{A_j, e_j\}$.
   (f) If $A_j > \ell_j$, then infeasible. Stop.
   (g) If $A_j + W < e_j$, then need to adjust for waiting time. If $L_j < e_j$, then infeasible; stop. Otherwise, set $A_j = e_j - W, D_j = e_j$.
   (h) If $j < 2n$, set $j = j + 1$ and return to (b).
2. Second pass
   (a) Set $\hat{W} = 0$ and $j = 2n - 1$.
   (b) Set $D_j = A_{j+1} - t_{j,j+1}$ and $A_j = \max\{A_j, D_j - W\}$.
   (c) If $j = i^+$:
      i. Set $\Delta = (D_{i^-} - D_{i^+}) - \alpha t_{i^+,i^-}$.
      ii. If $\Delta > 0$, then need to adjust for drive time. If $\Delta > \hat{W}$, then infeasible; stop. Otherwise, set $D_j = D_j + \Delta, A_j = \max\{A_j, D_j - W\}$, and $\hat{W} = \hat{W} - \Delta$.
      iii. If $D_j > L_j$, then infeasible. Stop.
   (d) Set $\hat{W} = \hat{W} + (D_j - A_j)$.
   (e) If $j > 0$, set $j = j - 1$ and return to (b).
3. Third pass
   (a) Set $j = 1$.
   (b) Set $A_j = D_{j-1} + t_{j-1,j}$ and $D_j = \max\{D_j, A_j\}$.
   (c) If $D_j > \ell_j$, then infeasible. Stop.
   (d) If $j = i^-$:
      i. Set $\Delta = (D_{i^-} - D_{i^+}) - \alpha t_{i^+,i^-}$.
      ii. If $\Delta > 0$, then infeasible. Stop.
   (e) If $j < 2n$, set $j = j + 1$ and return to (b).
4. Route is feasible and times A_j, D_j give a feasible schedule.

Note that the use of the total waiting time $\hat{W}$ only improves the practical efficiency, but is not required for correctness of the algorithm.

4 Discussion

The presence of maximum wait time and maximum ride time restrictions makes it difficult to establish whether a feasible schedule exist for a given sequence of pickups and deliveries. In this section, we elaborate on the various forms that maximum wait and maximum ride time may take and how this impacts our proposed algorithm. In doing so, we distinguish between transportation of people and transportation of goods.

When transporting people, it seems natural to assume that delivery windows are of the form $[0, \ell_j]$, i.e., only a latest delivery time is specified. The earliest delivery time is likely to be taken into account when the pickup window was defined. Furthermore, it may only be acceptable to have waiting time at a pickup stop, i.e., passengers are willing to wait (a short time) for the arrival of another passenger, but they are not willing to wait when a passenger is dropped off. Our methodology can obviously handle delivery windows of the form $[0, \ell_j]$, but it is not hard to see that it can also be easily adapted to handle different wait times at different stops. (Note that even though with a delivery window of the form $[0, \ell_j]$ the departure can coincide with the arrival, our algorithm may still introduce waiting time to ensure feasibility later on the route.) Having delivery windows of the form $[0, \ell_j]$ and disallowing waiting at drop-offs also has the advantage that the ride time, which is measured in our algorithm as $D_{i^-} - D_{i^+}$, becomes $A_{i^-} - D_{i^+}$, i.e., the time between arrival and departure, which is more natural in the case of people transportation.

In addition to being able to handle different wait time restrictions at different stops, our algorithm can also handle different ride time limits for different passengers. This flexibility allows handling of different passenger classes, similar to frequent flyers in airline applications. Note that in that case the wait time restriction at stop depends on the people in the vehicle rather than on the passenger getting in or out.

Another issue that has to be dealt with is waiting time at consecutive "stops" at the same location, e.g., two consecutive pickups. Maybe passengers already in the vehicle are willing to wait longer at the same location if several passengers have to get on, but maybe they will not. Limiting the waiting per location rather than per stop, can be handled in our algorithm by appropriately changing the update of the latest possible departure time L_j. If the previous stop occurs at the same location, then the latest possible departure time is updated as $L_j = \min\{L_{j-1}, \ell_j\}$ as opposed to $L_j = \min\{L_{j-1} + t_{j-1,j} + W, \ell_j\}$.

When transporting goods, it seem unnecessary to have wait time limits. However, it is realistic to assume delivery windows of the form $[e_j, \ell_j]$, since there may be administrative tasks that need to be performed at delivery that require the presence of personnel of the receiving party. In such situations it may be more appropriate to have maximum ride time restrictions based on the difference between departure times, i.e., $D_{i^-} - D_{i^+} \leq \alpha t_{i^+,i^-}$. It is not hard to see that our algorithm can easily be adapted to this change as well.

References

[1] J. Desrosiers, Y. Dumas, and F. Soumis. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences*, 6:301–325, 1986.

[2] J. J. Jaw, A. R. Odoni, H.N. Psaraftis, and N. H. M. Wilson. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research B*, 20B:243–257, 1986.

[3] H. N. Psaraftis. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14:130–145, 1980.

[4] H. N. Psaraftis. Analysis of an $o(n^2)$ heuristic for the single vehicle many-to-many euclidean dial-a-ride problem. *Transportation Research*, 17B:133–145, 1983.

[5] H. N. Psaraftis. An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science*, 17:351–357, 1983.

[6] H. N. Psaraftis. k-interchange procedures for local search in a precedence-constrained routing problem. *European Journal of Operations Research*, 13:391–402, 1983.

[7] H. N. Psaraftis. Scheduling large-scale advance-request dial-a-ride systems. *American Journal of Mathematical and Management Sciences*, 6:327–367, 1986.

[8] T. R. Sexton and L. D. Bodin. Optimizing single vehicle many-to-many operations with desired delivery times: I. scheduling. *Transportation Science*, 19:378–410, 1985.

[9] T. R. Sexton and L. D. Bodin. Optimizing single vehicle many-to-many operations with desired delivery times: Ii. routing. *Transportation Science*, 19:411–435, 1985.

[10] D. M. Stein. Scheduling dial-a-ride transportation systems. *Transportation Science*, 12:232–249, 1978.