An Auction-based Scheduling Approach to the Dynamic Dial-a-Ride Problem

Rebecca Martin[1] , **Isaac Kumar**[1] , **Stephen F. Smith**[1]

[1]Robotics Institute, Carnegie Mellon University
Pittsburgh, PA

{rebecca2@andrew.cmu.edu, isaack@cs.cmu.edu, sfs@cs.cmu.edu}

Abstract

The dial-a-ride problem (DARP) involves transporting a set of customers from respective origins to destinations within requested pick-up and drop-off time windows, using a fixed fleet of transport vehicles. In over-subscribed problem settings, the main objective is generally to maximize the number of requests that can be accommodated. In settings where there is sufficient vehicle capacity to service all requests, the objective shifts to some combination of minimizing overall travel time and minimizing number of vehicles used. One common application of DARP is found in door-to-door transportation services offered to elderly or disabled travelers. Most prior research into solution approaches to DARP has focused on the *static* problem (where all requests are known in advance), and emphasized the use of offline optimization techniques and extended computation. These techniques are not viable however in *dynamic* DARP domains, where new requests continue to emerge through the day and unexpected events continually force changes to preplanned trips.

In this paper, we address this gap and focus on solving the dynamic DARP formulation. We propose an online, auction-based scheduling strategy that constructs a solution over time and hence is capable of efficiently incorporating new requests as they arise (while also accounting for resource usage constraints and minimizing vehicle travel times). The performance of the proposed algorithm is evaluated on both a set of DARP feasibility benchmark problems from the literature and a large-scale real-world paratransit dataset more recently introduced by another incremental scheduling approach designed to maximize the number of requests that can be serviced in an oversubscribed context. In addition to obtaining favorable comparative results, the real-world paratransit data set is also used to analyze the impact of temporal uncertainty on solution quality, and to evaluate the potential for utilizing asymptotically different cost functions as a basis for introducing customer priority.

1 Introduction

Dial-a-ride paratransit systems play an instrumental role in providing equitable transportation services to special groups of the population, such as the elderly or handicapped. With a fare scheme comparable to that of a fixed-route transit, paratransit provides shared-ride, door-to-door service with flexible routes and schedules. The dial-a-ride problem (DARP) aims to optimally schedule and dispatch transit vehicles to satisfy requests for travel between pick-up and drop-off locations at specified times. A typical request in this context provides details on pick-up and drop-off locations, number of passengers, type of request, and preferred time windows within which it needs to be fulfilled. The scheduling component can be static, dynamic, or a combination of both. In the context of static DARP, all the requests and available vehicle fleet are known well in advance. In a dynamic DARP, the requests are serviced on an ongoing basis with an ability to increase the fleet size as needed. In the hybrid approach, reservations made in advance permit the construction of day-ahead (off-line) schedules, while the same day requests and other events such as trip cancellations, vehicle breakdowns etc. will need to be integrated into the current schedule as such events occur.

Since first proposed by Wilson in 1971 [Wilson *et al.*, 1971], the Dial-a-Ride Problem (DARP) has been studied extensively in the literature. Recent research within the context of static DARPs focuses principally on exact local neighborhood search and insertion heuristic techniques, together with a more accurate estimation model of fuel consumption [Knörr *et al.*, 2011] and service quality [Paquette *et al.*, 2012]. For example, exact methods such as branch-and-cut [Cordeau, 2006] have been developed based on mixed-integer programming for minimum cost. While considering complex constraints such as capacity, duration, time windows, pairing, and precedence, this method has a long execution time for large-scale problems. To overcome this weakness, heuristic approaches, as in GPU accelerated tabu search [Pandi *et al.*, 2018] and multi-atomic annealing heuristics [Ho *et al.*, 2018] for instance, present significantly improved efficiency and near-optimal solutions to various degrees.

Compared to the static problem, far less work has been done for dynamic DARPs. [Psaraftis, 1980] carried out one of the first studies on the Dynamic DARP focusing on the single vehicle case. In order to minimize the average ser-

vice period and reduce the dissatisfaction portion, an exact O($n^2 3^n$) algorithm was proposed to partially modify prior-generated routes. Later [Horn, 2002] came up with software level strategical heuristics to help solve large systems. It periodically operates local search and accounts for practical considerations like time window restriction, book cancellation, and future request anticipation. Moreover, aiming at more complex DARPs, [Beaudry *et al.*, 2010] have developed a two-phase algorithm, which considers the urgency level of various requests and various modes of transportation as substantial constraints.

While most of these methodologies are capable of generating near optimal schedules on small problem sets, they don't scale well on large-scale real-world problems. Second, most planning algorithms require long execution times to generate a feasible schedule, thereby limiting their applicability in online scheduling settings. To address these limitations, in this paper, we propose an auction-based scheduler framework capable of providing a scalable approach to paratransit scheduling while preserving the ability to optimize under real-world constraints. Furthermore, the proposed framework partitions requests into different importance classes, allowing it to account for the "sense of urgency" of the trip, and search for opportunities to minimize the trip duration without violating the constraints of other trips. We evaluate the performance of the framework on a set of synthetic benchmark problems that have been studied in the literature and show the ability of the auction-based scheduler in solving these challenging feasibility problems. Next, we test its efficacy on a real-world paratransit scheduling problem and benchmark the performance against the Generalized Task Swap (GTS) algorithm. Lastly, we consider the impact of execution uncertainty on the effectiveness of the proposed approach, versus reliance on a precomputed static solution.

The remainder of this paper is organized as follows. We first formalize the dynamic dial-a-ride problem of interest in Section 2. Next, in Section 3, the details of our proposed, online solution procedure, which adopts an auction-based scheduling approach, are presented. Section 4 provides an initial benchmark comparison of the approach's performance on a set of previously studied reference problems in the literature. Then, in Section 5 the results of further performance analysis of the approach on a real-world paratransit scheduling problem are presented. In addition to showing a favorable performance comparison to the approach that originally introduced this real-world problem, problem data is also utilized to examine the ability to manage temporal uncertainty at execution time and to incorporate request priority. Finally, in Section 6, we summarize the main contributions of the work, and discuss future research directions.

2 Problem Formulation

As mentioned at the outset, the dial-a-ride problem (DARP) involves transporting a set of customers from respective origins to destinations within requested pick-up and drop-off time windows, using a fixed fleet of transport vehicles. More precisely, A DARP can be specified as a triple $< R, V, D >$ where R is the set of customer requests received over the course of the scheduling horizon, V is the set of vehicles available to service these requests, and $D_{|L|\times|L|\times tod}$ is a duration matrix defined over the set of locations L contained in the underlying road network. Each request r is a tuple that specifies a pickup location l_r^p, a drop off location l_r^d, a pickup window $< t_r^{est}, t_r^{pref}, t_r^{lst} >$, a travel demand (i.e., number of passengers) d_r, and a priority pr_r. For purposes of this paper, we assume just three priority classes: 1 (lowest), 2, and 3 (highest). Each vehicle v has a total capacity C_v, and an available capacity A_v^t at any time t over the scheduling horizon. All vehicles $v \in V$ start out at a particular location l_{depot}. The duration matrix $D_{|L|\times|L|\times tod}$ compiles road network distance and travel speed information into a functional interface that, for any ordered pair of locations l_1 and l_2 and travel start time tod, defines the expected travel duration from l_1 to l_2.

A feasible vehicle schedule S_v then is a sequence of locations $l_1, l_2, ...l_k$ that (1) respects the expected travel duration between any consecutive pair of locations in the sequence, (2) satisfies the capacity constraint C_v of vehicle v at every point in time over the scheduling horizon, and (3) ensures that each request also satisfies a maximum ride-time constraint. The overall objective is to produce a feasible fleet schedule S that minimizes overall vehicle travel time.

3 Auction-based Scheduler (ABS)

We propose an online, incremental approach to solving the DARP. At any point in time during generation of the fleet schedule, all requests that have been received but not yet assigned to a vehicle are ordered by their respective preferred pickup times (earliest first), and requests are assigned to vehicles in a time forward manner (i.e., as the search moves forward across the scheduling horizon). A vehicle assignment to a given request r is considered only when its preferred pickup time t_r^{pref} falls within a pre-specified lead time, lt - i.e., $t_r^{pref} - t_{now} \leq lt$. At this time, vehicle options for handling r are computed and the vehicle that minimizes disutility (defined as a function of added travel time) is chosen.

At the heart of the approach, the assignment of requests to specific vehicles is adjudicated through a bidding process. Each vehicle is viewed conceptually as an agent that privately maintains its own schedule and interacts with an auctioneer agent to bid for and accept additional customer requests. As described in Algorithm 1, the auctioneer initiates the process of assigning a request r by first requesting each vehicle agent v to forecast its future geo-location at r's preferred pickup time (according to the commitments in v's current schedule S_v). This information provides a basis for determining the set of proximal vehicle agents to request bids from. Specifically, a pre-specified maximum driving distance d_{max}, which is incrementally relaxed if necessary until at least one vehicle candidate is within range, is used to determine the set of vehicle agents to request bids from (i.e., the set of vehicles V_r) .[1] If multiple bids are received, the auctioneer accepts the

[1]The need for d_{max} arises from computation reasons, due to the fact that a serial implementation of the auction process is used. This complication would not be strictly necessary if the implementation was instead decentralized.

Algorithm 1 Assign-Request (r,V)

Input:
$l_r^p \leftarrow$pick up location of r
for all v in V **do**
  $d_{(l_r^p,v)} \leftarrow$distance between v & l_r^p
end for
$d \leftarrow$the maximum driving distance to l_r^p (initially d_{max})
$c_{min} \leftarrow \infty$
while $c_{min} == \infty$ **do**
  for all v in V **do**
    if $d_{(l_r^p,v)} \leq d$ **then**
      $c \leftarrow$ Compute-Bid (v, r)
      if $c < c_{min}$ **then**
        $c_{min} \leftarrow c$
        $v_{opt} \leftarrow v$
      end if
    end if
  end for
  $d \leftarrow d + 1$
end while
Assign-Request (r, v_{opt})

Algorithm 2 Compute-Bid (v,r)

Input:
$C_v \leftarrow$capacity of vehicle v
$R_v \leftarrow$set of requests currently assigned to v
$D_v \leftarrow$cumulative demand of requests in R_v
$D_r \leftarrow$number of passengers in request r
$TW_v \leftarrow$ set of pickup windows $< loc, est, lst >$ in R_v
$S_v \leftarrow$current schedule for v
if $(D_v + D_r) > C_v$ **then** $c_v \leftarrow \infty$
else
  $S_v^* \leftarrow$ TSP-Solve $(R_v \cup r, TW_v \cup < l_r^p, est_r, lst_r >)$
  if S_v^* = **then** $c_v \leftarrow \infty$;
  else
    $c_v \leftarrow 0$;
    for all locations s in S_v **do**
      $t \leftarrow$ scheduled time of s in S_v;
      $t^* \leftarrow$ scheduled time of s in S_v^*;
      $delay \leftarrow t^* - t$
      $p \leftarrow$ corresponding request priority of s
      $c_v \leftarrow c_v + disutil(delay, p)$
    end for
  end if
end if
return c_v

lowest cost bid, and the winning vehicle is notified to update its schedule.

Algorithm 2 describes the bid computation procedure that is carried out by individual vehicle agents in response to the auctioneer query. The main objective of a vehicle agent is to generate a feasible schedule that minimizes disutility (i.e., the added cost of accommodating the new request). For a given vehicle v with a current set of assigned requests R_v and cumulative demand D_v, an optimal schedule S_v is generated by solving the corresponding traveling salesman problem (TSP) to produce the best ordering of requests in the schedule. For the experiments reported later in this paper, Google's OR-Tools [Perron and Furnon,] are used to formulate an exact TSP solution procedure and compute individual vehicle schedules.

Looking in more detail at the bid computation in Algorithm 2, it can be seen that only vehicles with sufficient available capacity to accommodate the new request r (i.e., $(D_v + D_r \leq C_v)$ will return a competitive bid. In the case that sufficient future capacity is confirmed, a hypothetical schedule (S_v^*) that incorporates r is computed and compared with the vehicle's current schedule (S_v) to determine the incremental cost or disutility associated with accommodating r. Specifically, for each stop s in S_v that services a request with priority pr_s, let t_s denote the scheduled time of s in S_v and t_{s^*} denote the scheduled time of s in S^*. Then

$$Cost = \sum_{s \in S_v} disutil((t_s - t_{s^*})), pr_s) \quad (1)$$

where $disutil$ is defined as:

$$disutil(\Delta t, pr) = \begin{cases} \log_2 \Delta t & \text{pr=1} \\ \Delta t & \text{pr=2} \\ (\Delta t)^2 & \text{pr=3} \end{cases} \quad (2)$$

$Cost$ is returned as the bid for request r.

The intention of having different priority-based disutility functions is to give higher importance to the requests with higher priority, by making their cost functions asymptotically different. In this way, the delay of a previously accepted passenger with higher priority will result in a higher cost.

4 Benchmarking

To first benchmark our algorithm, we compare its performance on a benchmark of challenging feasibility DARP instances published by [Cordeau, 2006]. These problems have been fairly extensively studied in the literature and have been proven feasible/infeasible by sophisticated, extended search techniques [Berbeglia *et al.*, 2011] [Jain and Van Hentenryck, 2011].

The problems in the benchmark are randomly generated, set inside a [-10, 10]x[-10, 10] grid, with the coordinates of the pick-up and drop-off nodes chosen according to a uniform distribution. The depot is located at (0, 0), the time horizon is up to 12 hours, and the time windows are 15 minutes. There are two sets of problems in the benchmark: for the instances in set a, the vehicle capacity is 3 and for the instances in set b, the vehicle capacity is 6. The maximum ride time constraint is 30 minutes for set a and 45 minutes for set b.

Tables 1 and 2 give a comparison between our algorithm and the algorithm from [Cordeau, 2006] on the instances in problem sets a and b respectively that are known to admit feasible solutions. As can be seen, ABS finds a feasible schedule for every feasible instance in problem set a, but finds a feasible schedule for only 2 of the 12 feasible instances in problem set b. For those instances that were not feasibly solved, the schedules generated by ABS satisfied 93% of the input requests on average, with an average cumulative de-

Table 1: Comparison of feasible instances from problem set a

Instance	Total Delay (min)		CPU Time (Sec)		% Requests Satisfied	
	ABS	Cordeau	ABS	Cordeau	ABS	Cordeau
a2-16	0	0	0.093573	0.6	100	100
a2-20	0	0	0.115272	3	100	100
a2-24	0	0	0.127073	85.2	100	100
a3-18	0	0	0.16073	24.6	100	100
a3-24	0	0	0.196368	4595.4	100	100
a3-30	0	0	0.230494	14400	100	100
a3-36	0	0	0.253812	14400	100	100
a4-16	0	0	0.14405	1289.4	100	100
a4-24	0	0	0.207046	14400	100	100
a4-32	0	0	0.292053	14400	100	100
a4-40	0	0	0.4037	14400	100	100
a4-48	0	0	0.440826	14400	100	100

Table 2: Comparison of performance on feasible instances from problem set b

Instance	Total Delay (min)		CPU Time (Sec)		% Requests Satisfied	
	ABS	Cordeau	ABS	Cordeau	ABS	Cordeau
b2-16	31	0	0.152909	12.6	87.5	100
b2-20	31	0	0.164435	0.6	90	100
b2-24	36	0	0.148558	165.6	91.66	100
b3-18	16	0	0.158972	77.4	94.44	100
b3-24	14	0	0.198912	436.2	95.83	100
b3-30	50	0	0.296693	11384.4	90	100
b3-36	46	0	0.260533	14400	94.44	100
b4-16	0	0	0.143746	146.4	100	100
b4-24	21	0	0.420327	3558.6	91.66	100
b4-32	0	0	0.274105	14400	100	100
b4-40	11	0	0.470664	14400	97.5	100
b4-48	8	0	0.55389	14400	97.92	100

lay of 26.4 minutes to those requests that could not be satisfied within specified time window constraints. One factor that likely contributed to the performance results obtained by ABS is the fact that some of the requests in the dataset from [Cordeau, 2006] specify a desired dropoff time instead of a desired pickup time. Our algorithm takes in the desired pickup time as part of the input and ensures that the passenger is picked up within a pre-specified time window starting at the desired pickup time. However, for those requests that specified a desired dropoff time instead of a desired pickup time, it was necessary to back-calculate the desired pickup time by subtracting the ride duration from the desired dropoff time, and this pre-processing greatly decreased scheduling flexibility, making these requests harder to fulfill within the desired dropoff window.

Table 3: Average Duration per Request

	Set a	Set b
Auction-Based	1.2367886179	1.2263719512
Cordeau	1.2085111789	1.3160442073

5 Real-World Application

As mentioned earlier, our Auction-based Scheduler (ABS) was designed principally to provide a scalable basis for generating efficient schedules under a myriad of real-world constraints. To explore this performance objective, we consider a reference data set introduced more recently by [Rubinstein *et al.*, 2012]. This data set characterizes the daily DARP faced by ACCESS Transportation Systems (the paratransit organization that services Allegheny County in Southwestern PA), where advance reservations for service must be combined with dynamic "same day" transport requests. In this version of the problem, the allowable time window for servicing a pickup request spans the interval from 10 minutes before the desired pickup time to 20 minutes after it. The maximum ride time constraint for a given request is the maximum of 20 minutes or twice the direct transit time between the request's pickup and drop-off locations, and all trip durations should be less than or equal to two hours. Trip durations are computed as a function of time-of-day, meaning that trips carried out during heavier traffic conditions (e.g., during rush hour) will experience longer trip durations. A typical day's operations by a given ACCESS service provider, involves up to 950 requests and 30-50 paratransit vehicles of 3 different types:

- *Sedans* - with capacity to carry four ambulatory passengers
- *Shoppers* - with capacity to carry 14 ambulatory passengers
- *Vans* - capable of carrying both ambulatory and wheelchair passengers, with a maximum of 10 ambulatory, or four ambulatory and three wheelchair passengers

We compare the results produced by ABS on this data set to those produced by the Generalized Task Swap (GTS) algorithm that was originally developed to provide ACCESS with a more effective, dynamic scheduling approach to their problem [Rubinstein *et al.*, 2012]. GTS is a controlled, iterative improvement algorithm that attempts to insert new requests into an existing schedule through some amount of reallocation. The basic intuition behind GTS is that, by reallocating scheduled requests whose tasks overlap in time with those in the new request, sufficient resource capacity can be freed up to accommodate the new request. Within the scope of algorithms that solve the paratransit scheduling problem, GTS is the most scalable and has been shown to outperform other leading algorithms, such as Tabu Search and CP [Berbeglia *et al.*, 2011], on the benchmark reference problems considered earlier. GTS is specifically designed to handle over-subscribed , where there is more demand than capacity. Over-subscription often leads to increasing wait times and can even result in failure to fulfill a subset of requests within the specified time windows. Thus, GTS's ability to re-plan based on feedback from a constantly changing environment gives it the flexibility to fulfill more requests than would otherwise be possible.

5.1 Baseline Comparison

The ACCESS dataset used for experimentation consists of a set of 902 requests for a single day, with a fleet of 36 vehicles (12 sedans, 9 shoppers, and 15 vans) available to service these requests. All requests are assumed to be known in advance and are auctioned in the order of their pickup times as described earlier in Section 3.

A new request r is put forth for auction 30 minutes prior to the specified pickup time. We set the initial value of d_{max} to 2 miles (this is the maximum distance between request pickup and the projected vehicle location at pickup time). The priority parameter (ρ) of each request is set to zero. Travel duration between locations is computed using ACCESS transportation systems' model. As per this model, a day is segmented into rush-hour and non-rush-hour epochs. The duration function

takes lat/lon coordinate pairs, and start time of travel for O-D pair of interest and in turn computes travel times based on great-circle distance and epoch specific parameters.

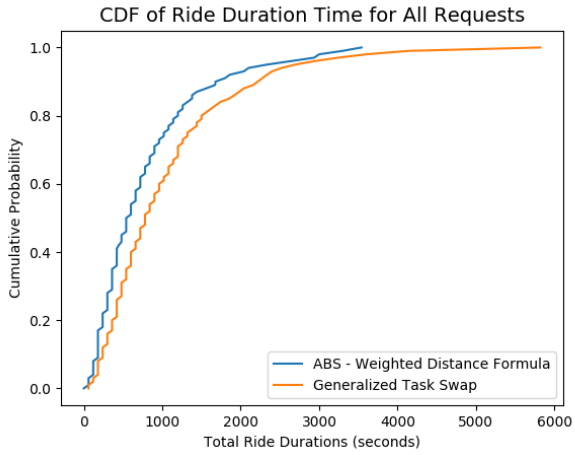


Figure 1: Ride duration for ABS and GTS schedules on ACCESS paratransit data

Trip Duration Distributions

Figure 1 presents the cumulative density functions (CDFs) of ride durations for ABS and GTS algorithms. The best performance is reflected by the curve farthest to the left. That curve always has a distribution of ride durations that are smaller than the others. Following inferences can be drawn from these plots: First, 90th percentile ride durations are significantly lower in the case of ABS as opposed to GTS. Second, when compared to GTS, ABS was able to *reduce the median ride duration* by **38.46%**, *mean duration* by **44.87%**, and *maximum duration* by **51.55%**. Lastly, with regard to minimizing the individual ride durations, ABS stochastically dominates GTS.

5.2 Impact of Lead-time parameter

One parameter that we found to have a significant impact on the scheduler performance was the lead time, that is the time difference between the auction time and the desired pickup time. We know that having a lead time that is too small is not good, as it reduces the chance of the request being picked up on time. However, we found that making the lead time too long also caused the performance to deteriorate, as shown in Figure 2. An explanation for this would be that increasing the lead time would require each vehicle to extrapolate further into the future to predict their location and availability. There would be more time in which the vehicle's schedule could change, so the corresponding bid has an elevated uncertainty attached to it. Thus, the ABS algorithm is not agnostic to the lead time.

5.3 Request Priorities

In order to test the hypothesis of the ABS algorithm's ability to incorporate the notion of priority into schedule generation, we considered a use case with three priority classes namely

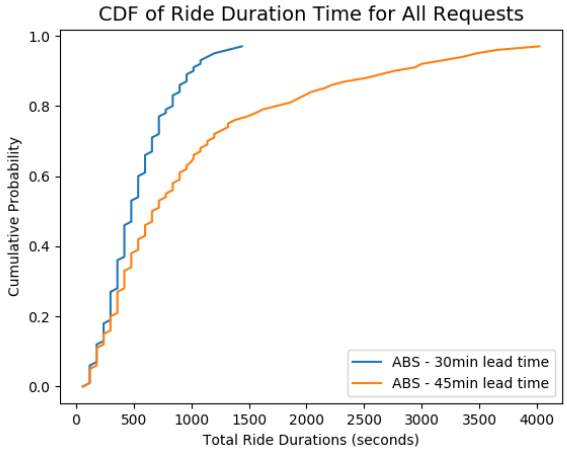


Figure 2: CDFs of the scheduled ride duration times in seconds comparing 30min vs 45min lead times

low (ρ_1), medium (ρ_2), and high (ρ_3) priorities. The associated disutility of these classes is given by:

$$disutil(\Delta t, \rho) = \begin{cases} \log_2 \Delta t & \rho_1 \\ \Delta t & \rho_2 \\ (\Delta t)^2 & \rho_3 \end{cases} \quad (3)$$

As presented in Equation 3, the disutilities of low, medium, and high priority classes are sublinear, linear, and exponential in nature. Each request in the dataset is randomly assigned a priority class. The unscheduled requests were then sorted in increasing order of their earliest start time for pickups. Ride duration results for each trip are assimilated and compared against the ABS algorithm with no priorities as the baseline. Table 4 summarizes these results. As it is evident from the table, the median ride duration was improved by 12.5% for high priority requests, remained the same for medium priority requests, and deteriorated by 12.5% for low priority requests. This trend suggests that the ABS algorithm is capable of generating schedules that reflect pre-specified user preferences. The less pronounced trends in average ride duration statistics indicates that ABS was able to accommodate the use preferences more effectively on requests with a shorter trip duration than those with longer durations. Figure 3 presents similar results but in the form of CDFs.

5.4 Execution Uncertainty

To demonstrate the robustness of our algorithm and test it in a more realistic scenario, we incorporated uncertainty into the execution of our schedules. A certain percentage of the requests were randomly chosen to be delayed; after these requests were picked up, the amount of time to travel to the next location was multiplied by a randomly determined factor. These factors were sampled from a Gaussian distribution with a mean of 1.2 and a standard deviation of 0.1. This setup more closely simulates the real world, where there are unexpected traffic delays or road closures.

Table 4: Ride Duration Statistics for Prioritization

Scenario	Ride Duration (Sec)					
	Average			Median		
	Base Case	Prioritized	% Impr	Base Case	Prioritized	% Impr
Overall Trips	572	565	1.23	480	480	0.0
Priority 1	598	601	-0.44	480	540	-12.5
Priority 2	581	555	4.41	480	480	0.0
Priority 3	539	540	-0.32	480	420	12.5

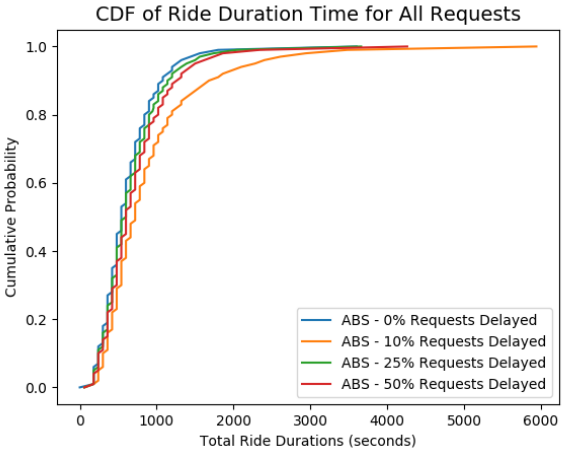


Figure 4: Ride duration for ABS schedules with executional uncertainty

In our experiments, we tested with delaying 10%, 25%, and 50% of the requests. From Figure 4, we can see that this added uncertainty only caused a small decrease in performance. However, an interesting thing to note is that the performance improved as we increased the percentage of delayed requests. This could be because each vehicle is reevaluating its schedule every time a request is delayed. More delayed requests lead to more reevaluations, and so allows each vehicle to reorder the stops on its route if needed.

5.5 Sensitivity Analysis

Our original experiment results suggest that both ABS and GTS algorithms are able to generate feasible schedules for all the requests. However, as mentioned earlier, paratransit vehicle schedules on a given day tend to be oversubscribed. In order to quantify the change in algorithmic performance for various levels of over-subscription, we conducted a sensitivity analysis by reducing the number of available vehicles by a certain percentage of the original number. We conducted the experiments with number of available vehicles set to 90%, 85%, and 75%. Table 5 summarizes the number of unscheduled requests and the additional delay needed to fit these requests in the schedule.

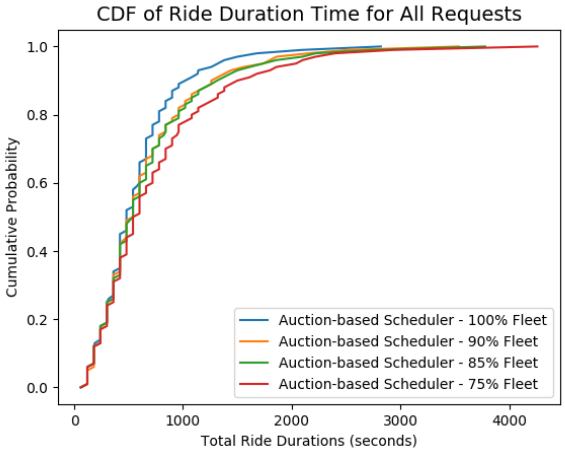
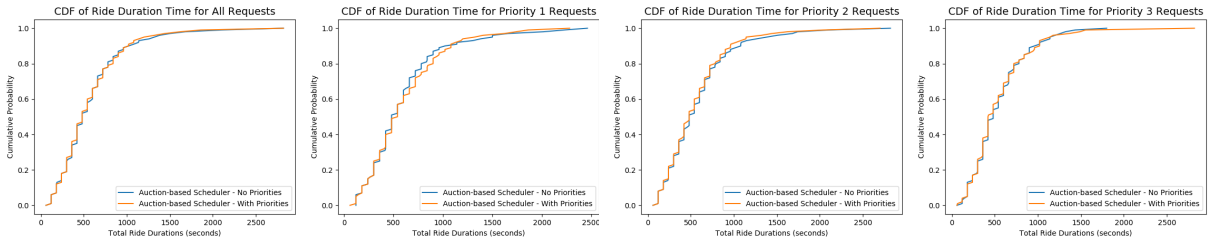


Figure 5: Ride duration for ABS sensitivity analysis

The results indicate that ABS was able to generate feasible schedules for all the requests, whereas GTS was unable to schedule some of the requests as the number of available vehi-



(a) All Requests (b) Priority 1 (c) Priority 2 (d) Priority 3

Figure 3: CDFs of the scheduled ride duration times in seconds, with priorities introduced

Table 5: Comparative Metrics for ABS & GTS

Metric	Orig		90%		85%		75%	
	ABS	GTS	ABS	GTS	ABS	GTS	ABS	GTS
# Unscheduled	0	0	0	1	0	3	0	17
Total Delay (Min)	0	0	0	54.6	0	15.9	0	1035.7

Table 6: Summary of Duration Statistics (ABS)

Metric (sec)	Resource Availability (ABS)				GTS
	100%	90%	85%	75%	(100%)
Median	480	540	540	540	780
Average	572	652	668	745	1037
Maximum	2820	3540	3780	4260	5820

cles reduced. The average computational time for scheduling a new request is 0.92 seconds for ABS, and 6.7 seconds for GTS suggesting that both ABS and GTS are well within the usability constraints for dynamic DARP problems.

We further explored the robustness of ABS algorithm by analyzing the statistical similarities/differences among various resource availability scenarios described above. Figure 5 presents the CDFs of ride durations for these scenarios. Larger variation in higher percentiles among the CDFs suggests that the reduction in resources had a higher impact on the ride durations of longer trips than it did on the shorter trips. Table 6 presents the summary statistics of ride durations for various ABS scenarios. These results suggest that even the most constrained ABS scenarios (75% resource availability) outperformed GTS with 100% resource availability.

6 Conclusion

In this paper, we have presented a scalable auction-based scheduler framework for addressing dynamic dial-a-ride problems. In addition to being able to accommodate real-world constraints, we have also shown that the ABS algorithm is capable of generating schedules that reflect pre-specified user preferences. Our scheduling algorithm was evaluated on a set of synthetic benchmark problems from the literature and on a real-world paratransit scheduling problem, compared against the Generalized Task Swap algorithm. The results show that ABS significantly improved on the results of the current state-of-the-art algorithm, GTS, in terms of both decreasing ride duration and utilizing resources more efficiently. We also incorporated request urgency and tested in scenarios with execution uncertainty. In the future, we will test different priority functions to see whether the cost functions need to be asymptotically different.

References

[Beaudry *et al.*, 2010] Alexandre Beaudry, Gilbert Laporte, Teresa Melo, and Stefan Nickel. Dynamic transportation of patients in hospitals. *OR spectrum*, 32(1):77–107, 2010.

[Berbeglia *et al.*, 2011] Gerardo Berbeglia, Gilles Pesant, and Louis-Martin Rousseau. Checking the feasibility of dial-a-ride instances using constraint programming. *Transportation Science*, 45(3):399–412, 2011.

[Cordeau, 2006] Jean-François Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54(3):573–586, 2006.

[Ho *et al.*, 2018] Song Guang Ho, Ramesh Ramasamy Pandi, Sarat Chandra Nagavarapu, and Justin Dauwels. Multi-atomic annealing heuristic for the dial-a-ride problem. In *2018 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, pages 268–273. IEEE, 2018.

[Horn, 2002] Mark ET Horn. Fleet scheduling and dispatching for demand-responsive passenger services. *Transportation Research Part C: Emerging Technologies*, 10(1):35–63, 2002.

[Jain and Van Hentenryck, 2011] Siddhartha Jain and Pascal Van Hentenryck. Large neighborhood search for dial-a-ride problems. In *International Conference on Principles and Practice of Constraint Programming*, pages 400–413. Springer, 2011.

[Knörr *et al.*, 2011] W Knörr, S Seum, M Schmied, F Kutzner, and R Anthes. Ecological transport information tool for worldwide transports. In *Methodology and data update*. IFEU Heidelberg, Oko-Institut, 2011.

[Pandi *et al.*, 2018] Ramesh Ramasamy Pandi, Song Guang Ho, Sarat Chandra Nagavarapu, Twinkle Tripathy, and Justin Dauwels. Gpu-accelerated tabu search algorithm for dial-a-ride problem. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2519–2524. IEEE, 2018.

[Paquette *et al.*, 2012] Julie Paquette, François Bellavance, Jean-François Cordeau, and Gilbert Laporte. Measuring quality of service in dial-a-ride operations: the case of a canadian city. *Transportation*, 39(3):539–564, 2012.

[Perron and Furnon,] Laurent Perron and Vincent Furnon. Or-tools.

[Psaraftis, 1980] Harilaos N Psaraftis. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14(2):130–154, 1980.

[Rubinstein *et al.*, 2012] Zachary B Rubinstein, Stephen F Smith, and Laura Barbulescu. Incremental management of oversubscribed vehicle schedules in dynamic dial-a-ride problems. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[Wilson *et al.*, 1971] Nigel HM Wilson, Joseph M Sussman, Ho-Kwan Wong, and Trevor Higonnet. *Scheduling algorithms for a dial-a-ride system*. Massachusetts Institute of Technology. Urban Systems Laboratory, 1971.