Dynamic Management of Paratransit Vehicle Schedules

Zachary B. Rubinstein and **Stephen F. Smith**
The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
{zbr,sfs}@cs.cmu.edu

Abstract

In this paper we describe REVAMP, a mixed-initiative tool for real-time management of paratransit vehicle schedules. Like many applications, paratransit scheduling is a dynamic, execution-driven process, where unexpected events (e.g., traffic, breakdowns, new requests, cancelations) continually force changes to precomputed schedules. The design of REVAMP aims at support for this dynamic, real-time scheduling process. Real-time information on the status and location of vehicles and pending trips is used by REVAMP to maintain a "live" schedule and provide the coordinating Dispatcher with early visibility of potential delays. In response to detected problems or opportunities, REVAMP can be used to generate options for rearranging vehicle schedules to achieve better quality of service. REVAMP is being developed to support daily operations at ACCESS transportation systems, which provides an advance reservation, shared ride paratransit service for the greater Pittsburgh area in southwestern Pennsylvania. We are currently integrating REVAMP into the existing technology base used to support daily operations by one of ACCESS's service providers for an initial pilot test. We describe the principal components of REVAMP and the current state of our solution to the ACCESS scheduling problem.

1 Overview

Paratransit transportation is one of the primary means for people with disadvantages to get around in their daily lives. Typically, this door-to-door service is booked in advance and is publicly subsidized. As the demand for paratransit services increases and availability of public funding decreases, there is a constant balancing act between maintaining a high quality of service while keeping the operations financially feasible. One of the key activities in striking this balance is efficient allocation of vehicles to service trips, especially over the myriad of events that make it difficult to anticipate how a schedule is going to have to evolve throughout its execution during the day. The effectiveness and efficiency of paratransit operations depends heavily on the ability to dynamically manage vehicle schedules in response to execution dynamics.

To augment the ability of Dispatchers to manage their vehicles over execution events, we have developed REVAMP (REal-time Vehicle Allocation application for increased Mobility in Paratransit operations), a mixed-initiative, dynamic scheduling system. REVAMP receives real-time status and location information from the vehicles in the field and maintains a "live" schedule that is constantly updated to reflect what actually happens during execution. This active model provides Dispatchers with better situational awareness of the states of current and future trips and a basis for early detection of trips that are in jeopardy of having poor quality of service. For these problematic trips, REVAMP generates options to present to Dispatchers for rerouting the trips in order to improve their service.

The immediate target of REVAMP is to address the daily operations problem faced by service providers of ACCESS Transportation Systems, the largest paratransit organization in the Southwestern Pennsylvania Region. ACCESS oversees seven individual service operations that are apportioned geographically to the region. These service providers make about 6000 trips daily throughout Allegheny County. Of these about forty percent are subscription service (regularly scheduled) and sixty percent are day-ahead reservations. Over the past twelve months, ACCESS has provided service to around 26,000 unique customers. We are currently developing a system that integrates REVAMP into ACCESS's existing technology base, for the purpose of carrying out an initial pilot test with one of their larger service providers. Our overall goals are to improve customer quality of service, while simultaneously decreasing provider costs and providing the opportunity to offer expanded same-day request service.

2 Paratransit Management Problem

The Paratranist management problem is an instance of the Dial-a-Ride Problem (DARP) (Cordeau and Laporte 2003). The objective in a DARP is to design vehicle schedules to satisfy requests for travel between pick-up and drop-off locations at specified times. Typically, as input, DARPs specify a set of requests, a set of available vehicles, and a set of constraints ensuring the quality of the service, e.g., time windows within which pick-ups must be made or maximum allowable ride time for a passenger. DARPs may be oversubscribed, i.e., not all requests can be serviced within their constraints, in which case the constraints may be relaxed in order to help accommodate the extra requests. There often are optimizing objectives for DARPs, such as minimizing cost by minimizing the number of vehicles used while allowing some level of constraint relaxation, or maximizing service quality by determining the minimal number of vehicles to satisfy the requests without relaxing constraints. Formally, there are two classes of DARPs, static and dynamic. In a static DARP, all the requests and the available vehicle are known up front. In a dynamic DARP, the requests are serviced as they arrive and the available vehicles can be increased. Most real-world DARPs are hybrids, with a majority of the requests but not all being known up front and there mostly being a static number of available vehicles but extra ones are available at a cost.

We will focus on the version of DARP as it is instantiated in ACCESS. Currently, ACCESS does not support day-of requests, so the requests are known in advance. The one exception to this advance knowledge is "will call" requests, where passengers do not know the exact time of a return trip, e.g., for a doctor's appointment. These requests are handled dynamically as soon as the call is received to pick up the passenger.

The number of available vehicles is also known in advance. There are different types of vehicles available, each providing capacity to carry a specified number of passengers. Some types of vehicles provide wheelchair capacity, while the others have strictly ambulatory capacity. In the former case, wheelchair capacity can be converted to ambulatory capacity, so the overall carrying capacity will vary depending on how the wheelchair capacity is used.

The service quality constraints for ACCESS are the following:

- *Pick-Up-Window Constraint* - The constraint on the window of time it is permissible to pick up a passenger is relative to time that ACCESS negotiates with the passenger for his or her pick-up. Specifically, a pick-up should not occur any earlier than ten minutes before the negotiated time or any later than twenty minutes after the negotiated time. For "will-calls", the pick-up should occur no later than forty-five minutes after receiving the call.
- *Ride-Time Constraint* - the ride time for a passenger should be no longer than the maximum of twenty minutes and the minimum of two hours and twice the estimated time to go directly from the pick-up to the drop-off (i.e., $\max(20min, \min(2hrs, 2 * \textit{direct-tranisit-time}))$).

The providers are allowed to relax these constraints, but, overall performance, is based on meeting them. If these constraints are violated too often or by large magnitudes, the provider risks having its service area reduced or loosing its contract. The goal for the provider is to meet these constraints while minimizing the resources it needs to use.

In preparation for the next day, providers generate a solution , i.e., a set of vehicle schedules, to the DARP for the following day's requests. The solution is generated by using an offline scheduler based on (Jaw 1984) to create the base schedules and then hand tweaking them. Complicating this process is that, on average, there is a 15% request cancellation rate that occurs during the day of service. The human schedulers actually over allocate the vehicle timelines in anticipation of these cancellations. That is, the start-of-day schedules would require frequent relaxations of the constraints if they were to execute as scheduled, so the schedules start with a number of trips that are already in jeopardy of missing their service constraints. It is left to the Dispatchers to resolve these problematic trips as cancellations arrive.

Although an advance reservation policy enables the advance development of a daily operations schedule, unexpected events that occur as execution proceeds (e.g., vehicle breakdowns, traffic accidents, "will call" requests, trip cancellations) quickly force changes and degrade the quality of originally planned vehicle itineraries. In current practice, the Dispatchers responsible for coordinating the movements of a service providers vehicles respond to these events in a fire fighting fashion. Since the start-of-day schedule becomes an increasingly distant reference point as the day progresses, they often become aware of problems (e.g., potential late pickups) late, restricting their ability to effectively respond. Although they have visibility of the real-time location of vehicles, Dispatchers must often make decisions to redirect vehicles without good understanding of the downstream consequences to subsequent trips scheduled for various vehicles. During peak request periods, the Dispatcher can often become overwhelmed due to time pressure and decision complexity. Ultimately, the quality of decisions depends heavily on the Dispatchers expertise, and it is difficult to find and retain experienced personnel in this position. As a result, the decisions that are made are often suboptimal with respect to maximizing customer quality of service, and this has a cascading effect through the day. This situation is typical of operations in other paratransit service contexts.

3 REVAMP

REVAMP is designed to address the problems identified above and provide a basis for improving the real-time decision-making of service provider Dispatchers. Most basically, it is designed to incrementally accept real-time updates of vehicle status, and provide the Dispatcher with a live schedule that continually reflects the current execution state. Since this schedule encodes all relevant constraints and requirements (e.g., expected travel durations, negotiated pick-up times, maximum transit time limits), one immediate benefit is the early detection and alerting of emerging problems (e.g., given the customer pick-up that was just reported, the next scheduled pick-up after this current trip is projected to be beyond its acceptable window). This capability alone affords the Dispatcher with more time to take corrective action, and in the worst case, enables the Dispatcher to inform customers in advance of unavoidable delays (a service that ACCESS has expressed an interest in providing). To support Dispatcher response to detected problems, REVAMP is designed to generate options for rearranging trips across vehicles to minimize the impact on customer quality of service. This same options generation mechanism can also be used as a basis for improving the efficiency of vehicle itineraries when opportunities arise (e.g., trip cancelations) and for accommodating new requests that arrive dynamically through the day (e.g., "will call" return trips).

REVAMP consists of a suite of components whose underlying object-model maps client requests and their corresponding pick-up and drop-off tasks to a Simple Temporal Network (STN) (Dechter, Meiri, and Pearl 1991). Included in that object model are vehicles and their timelines (schedules) populated with the request and travel tasks. REVAMP components include a *schedule loader* for installing the start-of-day schedule, a *schedule updater* for integrating real-time start and finish time information as well as cancellations into the schedule and for identifying problematic trips and opportunities for improving existing schedule quality, and a *gap finder* for generating high-quality options for inserting a request's constituent tasks on vehicle timelines.

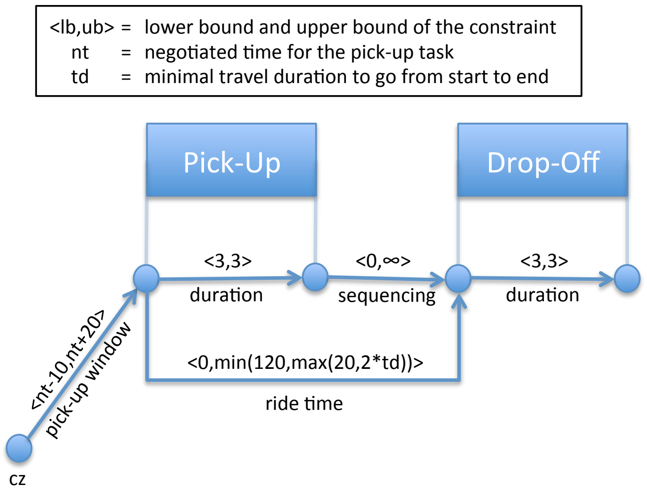
3.1 Object Model

The top-level objects in the model for REVAMP are vehicles, requests, and tasks. There are three types of vehicles that differ in capacity and whether or not they are wheelchair accessible. Each vehicle has a corresponding timeline (schedule), which consists of an ordered set of tasks. A request represents the information provided when a client books a trip and has attributes for the negotiated time (preferred pick-up time or the time when a "will call" is received), the latitudes and longitudes of the pick-up and drop-off locations, the number of wheelchair passengers, the number of ambulatory passengers, and whether or not it is a "will call" trip. A request also indicates the provider to which the request has been assigned by ACCESS and a unique identifying number for the request known as a trip id.

When a request is created, both pick-up and drop-off tasks are generated and mapped to the underlying STN. The STN is a graph of time points connected by binary constraints that determine the permitted intervals, i.e., lower and upper bounds, for any given time point. An absolute constraint emanates from the special, anchored calendar zero (cz) time point and sets the lower and upper bounds of the target time point to absolute times. A relative constraint sets the lower and upper bounds of the target time point as offsets from the source time point's bounds. A task is represented in the STN as a start time point and an end time point connected by a duration constraint whose lower and upper bound values are the time it takes to complete the task. As shown in Figure 1, the pick-up-window constraint in ACCESS is represented as an absolute constraint on the start time point of the pick-up task. The ride time constraint is represented as a relative constraint from the start time point of the pick-up task to the start time point of the drop-off task.

In addition to the pick-up and drop-off tasks, there are travel tasks, which are dynamically generated when inserting the request tasks onto a vehicle timeline. The computation for determining the duration of a travel task depends on the origin and destination of the travel and the time of day when the travel occurs. The travel duration is calculated by, first, using a great-circle computation to determine the distance between the origin and destination. Then, a speed is selected based on whether or not the travel will occur during rush hour. The final duration is the distance divided by the speed. The non-rush-hour speed is used if there is enough time within the bounds of the travel task to complete the travel without having to travel during rush hour. Otherwise, the rush-hour speed is used.

One of the complications of this duration model is that, as travel tasks move in time, i.e., become earlier or later due



<lb,ub> = lower bound and upper bound of the constraint
nt = negotiated time for the pick-up task
td = minimal travel duration to go from start to end
Pick-Up
Drop-Off
<3,3>
duration
<0,∞>
sequencing
<3,3>
duration
<nt-10,nt+20>
pick-up window
<0,min(120,max(20,2*td))>
ride time
cz

Figure 1: Request Tasks STN Mapping.

to changes in the schedule, their durations may change as a result of moving in or out of rush-hour periods. To ensure proper travel durations, REVAMP adjusts the travel durations of constituent tasks where necessary after any change is made to a vehicle timeline. This process is elaborated below in the descriptions of the components.

3.2 Schedule Loader

The schedule loader converts the provider's start-of-day schedule into the underlying object model. Complicating this procedure is that the start-of-day schedules, in anticipation of the 15% cancellation rate, are oversubscribed and can have many requests scheduled that will violate either or both pick-up-window and ride-time constraints. For example, in the start-of-day schedules of a particular service provider for a 1 week period, approximately 2/3 of the scheduled requests had constraint violations. A start-of-day schedule of request tasks (i.e., pick-ups and drop-offs) for a given vehicle is loaded into REVAMP by processing each task in order from earliest to latest. Constraint violations are handled by a constraint relaxation process (described below).

A given task is appended to a vehicle timeline by first checking to see if the task is at a different location than that of the previous task. In the case where the timeline is empty, the vehicle is assumed to be at the location of the garage of the provider. If the new task is at a different location, then a travel task is created with the appropriate duration for getting from the location of the previous task to the location of the new task sometime between the earliest finish time of the previous task and the latest start time of the new task. If the travel task is created, it is inserted on the timeline by adding a sequencing constraint between the end time point of the previous task and the start time point of the new travel task. A sequencing constraint dictates that the source time point must occur before (or simultaneously with) the target time point. Finally, the task being processed is added to the timeline by inserting a sequencing constraint between the last task on the timeline and it.

Since the start-of-day schedule can be overloaded, it may not be possible to assert the sequencing constraint to the new task without violating its upper bound constraint (either the pick-up-window-constraint on a pick-up task or the ride-time constraint on a drop-off task). If it cannot be added, then the upper-bound constraint is *relaxed* by remembering its original upper bound value and setting the upper bound on the constraint to ∞, i.e., unbounded. Then, once the sequencing constraint is asserted, the upper bound on the upper bound constraint is anchored to ensure that the current tardy time does not slip any later.

The end result of the schedule loader is a complete start-of-day schedule modeled with all necessary relaxations and in a form that can continue to be modified to reflect the impact of execution-time events on the schedule.

3.3 Schedule Updater

The schedule updater is responsible for keeping the internal model of the schedule in sync with how the schedule is actually executing. It incorporates into the model execution-time updates on the actual start and finish times of request tasks and on cancellations. In each case, constraints on downstream tasks are relaxed or tightened when necessary/possible. The schedule updater also detects impending requests that are in jeopardy of missing service quality constraints, and, in the case of cancellations, triggers the generation of options for the most imperiled requests in order to improve overall schedule quality.

Actual start and finish times for request tasks are reported by the vehicles as they make their stops. When the actual start time for a task is received, the start time point of the task is fixed at the reported time by adding an absolute constraint with the lower and upper bounds set to the reported time. In the straightforward case where schedule updates on the start and finish times of tasks are received in the order specified by the schedule, the updater relaxes any constraint that it must in order to achieve consistency with actual times. If a task finishes later than expected, then the duration constraint on that task is lengthened. If there is a violation when attempting to assert the new duration length, the STN returns the set of constraints that are inconsistent with the new constraint and the amount that the new constraint violates the existing constraints. The returned constraints are used to relax the downstream constraints in order to make it possible to assert the new duration constraint. First, the returned set of constraints are searched for the first upper bound constraint found. Then, that constraint is relaxed by the magnitude of the violation. Next, the new duration constraint is attempted once more. If it fails, then the relaxation process is invoked again with the new set of returned constraints. This iteration continues until the duration constraint can be asserted.

In the case where a task finishes early, the duration constraint is shortened, all subsequent relaxed tasks on the timeline are tightened when possible. If a pick-up task starts earlier than its pick-up-window constraint allows, that constraint is relaxed to accommodate the actual start time. After any changes are made to the timeline, the scheduled tasks are mapped over to ensure that relaxed constraints are as tight as possible and travel tasks have their appropriate durations. A possible side-effect of these modifications is that the upper-bound constraints on downstream tasks may have to be tightened or relaxed, depending on whether or not the new travel durations are shorter or longer. Note that previously relaxed constraints are never tightened beyond their original values.

Drivers are not required to pick up passengers in the order of the schedule and, therefore, can report starting a task that is not the next task expected to execute on the schedule. In this case, the schedule updater moves the reported task up in the schedule to follow the last completed task, inserting travel tasks where needed to get from the last completed task to the reported task and from the reported task to the task that was scheduled to execute next. This insertion may require relaxation of some of the upper-bound constraints on pending tasks that were expected to execute earlier. After the insertion, the durations of the travel tasks among the subsequent tasks are corrected when necessary.

As the actual start and finish times are reported, the schedule updater keeps a running list of all tasks that have had one or more of their constraints relaxed. This list is used to determine if REVAMP should alert the human dispatcher of trips that are likely to violate the service constraints. The current policy alerts the dispatcher of trips in jeopardy that are to be serviced within the next two hours. However, if desired the dispatcher can also request to view all endangered trips, along with the expected magnitude of delay in each case.

The schedule updater also updates the schedules of vehicles when requests are cancelled. When a request is cancelled, its pick-up and drop-off tasks are removed from the timeline as well as their corresponding travel tasks. Then, when necessary, new travel tasks are inserted to bridge the distance between tasks on either side of the removed tasks. As with all timeline modifications, the travel tasks on the timeline are checked to ensure that they have the correct durations. There is a restricted cancellation that occurs when a driver arrives at a pick-up task and discovers that no one is there. In this "no-show" case, the drop-off task for the request is removed from the timeline. A cancellation potentially represents an opportunity for remedying an endangered trip by moving it to the timeline with the cancellation that now has more slack on it. The schedule updater will trigger the gap finder on cancellations to present rescheduling options to the dispatcher for rerouting problematic trips.

3.4 Gap Finder

The gap finder generates scheduling options for servicing a given request. It is triggered either automatically by the schedule updater in the case of cancellations or deliberately by the Dispatcher via the list of problematic or unscheduled requests. In the latter case, if the request is already scheduled, it is temporarily unscheduled before searching for options. The general strategy of the gap finder is to search all vehicle timelines for feasible slots between scheduled request tasks for the request's pick-up and drop-off tasks. A slot is feasible for a task if *a*) the time window of the slot can accommodate the constraints of the task plus any additional travel required, and *b*) the current available ambulatory and wheelchair capacity is sufficient to accommodate the passenger demands on the request

All pairs of feasible slots for a request on a given vehicle timeline are found by traversing the timeline and trying the eligible slots for the pick-up task on the request. For each slot within the pick-up window, the pick-up task is inserted and required travel tasks are inserted and adjusted as required. If no violation occurs, the slot is feasible. Then, with the pick-up task temporarily inserted, the downstream slots on the timeline are checked to see if they are feasible slots for the drop-off task. The valid slots for the drop-off task are found by searching the remaining slots on the timeline that have time bounds within the constraints of the drop-off task and hypothetically inserting the drop-off task and associated travel into each of these slots. Any time a drop-off task is successfully added, the travel tasks of other surrounding tasks are adjusted to ensure that they have the correct durations. If those durations cannot be enforced without a violation, the drop-off slot is not feasible. For each pair of feasible slots found, a candidate is generated and saved. Then, the drop-off task is unscheduled and the subsequent slots are checked to see if they are valid for the drop-off task. Once there are no more slots to search for the drop-off task, the pick-up task is unscheduled and the subsequent slots on the timeline are examined to see if they are valid slots for the pick-up task. This process continues until all feasible candidates for that request on that timeline have been generated. All possible candidates are generated by searching all the vehicles' timelines.

For each candidate option generated, the following performance measures are computed and associated as the candidate is generated:

- *Pick-Up Tardiness* - the maximum of zero or the amount of time by which the earliest start time of the pick-up task exceeds the negotiated time.
- *Drop-Off Tardiness* - the amount of time by which the earliest start time of the drop-off task exceeds the earliest possible start time of drop-off task (assuming direct travel from pick-up location to drop-off location).
- *Additional Travel Duration* - the amount of travel time that had to be added to the vehicle timeline to accommodate the candidate.
- *Additional Total Tardiness* - the sum of the pick-up tardiness and any tardiness introduced to other tasks on the timeline due to the shifting required to accommodate scheduling the candidate.

These metrics are used to prioritize the candidates. Which metrics are optimized depends on what is most valued in the organization. For ACCESS, the most important quality is to service all the requests without violating their service constraints. To match this priority, the gap finder prioritizes candidates to minimize Additional Travel Duration, which minimizes the amount of overall time that has to be allocated to service a request and, thus, leave more available capacity to meet future requests. A subset of the top candidates are presented to the Dispatcher as possible options, since the Dispatcher may have additional knowledge about the executing environment that might give preference to a lesser ranked candidate. In automated mode, REVAMP would simple select the top-ranked choice and add this commitment to the schedule.

When the timelines are tight, it is possible that no candidate can be found for a request. In this case, the upper bound constraints on the pick-up and drop-off tasks of the request are iteratively relaxed by set amounts until candidates can be found for the request. Before a candidate is generated, the upper bound constraints are tightened to their original (satisfied) values, the feasible relaxation values determined for these upper bound constraints by this search are stored with the candidate. As before, the candidates are prioritized. But, for relaxation, we chose to minimize the sum of the pick-up and drop-off tardiness, since in this case this alternative metric tends to better minimize worst case tardiness. When a relaxed candidate is scheduled, the upper bound constraints on the pick-up and the drop-off tasks are first relaxed to their stored values on the candidate.

4 Current State

The individual components of REVAMP are implemented and have been tested on the types of events present in historical data provided by ACCESS. These events include start-of-day requests and routes, actual start and finishes time, cancellations, no-shows, and vehicle downtimes. We are currently integrating REVAMP into the existing technology base provided by ACCESS to its providers, in preparation for an initial pilot test of developed capabilities with a particular ACCESS provider. This effort involves developing an API for the base system to convey to REVAMP the initial schedule, real-time tracking information, and the changes made to the schedule by Dispatchers, and for REVAMP to inform the base system of changes in the schedule during execution, of updates to the list of requests that are currently expected to violate service constraints, and of options for reassigning requests to vehicles to improve overall schedule quality. Where appropriate, we will use the base system's user interface to minimize user training, but we are also exploring use of other graphical display formats that better convey real-time schedule status and dynamic scheduling options.

We are currently running experiments to evaluate the potential improvement REVAMP can provide to Dispatchers. We have from ACCESS the start-of-day information, including all requests, initial schedules, and vehicles used, and the corresponding end-of-day information, including the actual times for the stops on the routes, the cancellations (including no-shows), and the information on the "will call" requests. We are using this data to simulate the use of REVAMP in fully autonomous mode to manage the schedule over the events of those days and compare its performance to the actual performance of the Dispatchers on those days. The metrics we use for comparing performances are the number of relaxed requests, the number of relaxed pick-ups, the mean/median/maximum relaxation for pick-ups, the average relaxation for pick-up over all pick-ups, the number of relaxed drop-offs, the mean/median/maximum relaxation for drop-offs, and the average relaxation for drop-offs over all drop-offs. This comparison is obviously an approximation, since the moment REVAMP diverges from the actual schedule, actual travel times are no longer available nor are the effects of drivers servicing stops in a different order than the schedule. To make the experiments more realistic, we have formed models based on the year's worth of data that we currently have. These models vary the duration appropriately on the new trips and report stops out of order in appropriate contexts with representative frequency.

As a preliminary result to demonstrate that there is room for improvement, we took start-of-day schedules, loaded them, and collected the comparison metrics. Then, we used the gap-finder (in automated mode) to schedule all requests in the start-of-day schedules using the same configuration of vehicles. We ran this experiment on a week's worth of data from one of the providers. Table 1 shows the schedule comparison for one day, which is representative of the other days' results. As can be seen by the far fewer relaxations, reduced maximum relaxations, and mostly reduced averaged relaxations, there appears to be substantial room for improvement. A caveat to this result is that the human schedulers at the provider may be considering other constraints, such as stability of subscription routes, driver familiarity with particular areas, etc., that REVAMP does not currently model. And, as noted earlier, the start-of-day schedule is formed with the expectation of a 15% cancellation rate. Nevertheless, the difference is so dramatic that it appears that substantial improvement is likely. In particular, just having the start-of-day schedule that is better aligned within customer service constraints reduces the initial demand on the Dispatcher to pay attention and manage those trips that are already in jeopardy of being delayed.

	provider	REVAMP
total-requests	902	902
total-relaxed-requests	577	17
total-relaxed-pickups	386	17
avg-relaxed-pickup-overall	25m 25.84s	44.80s
avg-relaxed-pickup	59m 25.57s	39m 37.29s
median-relaxed-pickup	31m 9s	42m 29s
max-relaxed-pickup	5h 1m 17s	1h 15m 17s
total-relaxed-dropoffs	327	9
avg-relaxed-dropoff-overall	7m 59.65s	13.63s
avg-relaxed-dropoff	22m 3.07s	22m 46s
median-relaxed-dropoff	13m 14s	16m 5s
max-relaxed-dropoff	4h 10m 11s	50m

Table 1: Start-of-Day Schedule Comparison

5 Future Work

Our short-term focus is on refining and hardening REVAMP in its integration with ACCESS's existing technology base to ensure its performance and reliability in the pilot and in wider deployment in ACCESS. A large part of that effort will be focused on developing the user interface that will allow the Dispatchers to grasp the situational awareness and options provided by REVAMP, evaluate the alternatives, and take quick effective actions. If the pilot is successful and REVAMP is able to quickly offer good scheduling solutions to unscheduled requests, we would like to work with ACCESS to offer a limited same-day service.

We are also investigating incorporation of more sophisticated option generation capabilities within REVAMP. First, the current method for relaxing a request that cannot be feasibly scheduled has a limited degree of freedom in that the only constraints relaxed are those of the tasks constituting the unscheduled request. It may be possible to find better solutions by adjusting multiple trips on timelines to accommodate a given unscheduled request. One approach we are exploring is a variant of task swapping (Kramer and Smith 2004), where other temporally overlapping requests are temporarily unscheduled to create space for a target request and then reinserted in a feasible manner.

Second, we would like to expand REVAMP to include backward search from the drop-off time, i.e., given when a person needs to be at a destination, schedule backward to establish the pick-up time. In this case, the drop-off time would be treated as a hard, upper-bound constraint. In addition to better servicing appointment constraints of passengers, it also could be used to improve multi-modal traveling where paratransit transportation is used to connect with other fixed route services. In general, REVAMP should provide a basis for better coordination of multi-modal trips involving ACCESS transport to a public transit gateway (or other fixed route service) and vice versa. By incorporating external schedules and capitalizing on real time information of the locations and status of ACCESS vehicles (and, in the future, potentially the real-time location of public transit vehicles), we believe that better synchronization of multi-leg trips can be achieved, resulting in less customer wait time and shorter overall travel times. To facilitate this type of planning by passengers, we envision use of on-line browser-based and mobile-based applications for planning and booking both basic paratransit trips and multi-modal trips.

Acknowledgements The research described in this paper was funded in part by a grant from the 2010 Federal Transit Administration New Freedoms Initiative and the Robotics Institute at Carnegie Mellon University.

References

Cordeau, J.-F., and Laporte, G. 2003. The dial-a-ride problem (darp): Variants, modeling issues and algorithms. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies* 1:89–101.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49:61–95.

Jaw, J.-J. 1984. *Solving large-scale dial-a-ride vehicle routing and scheduling problems*. Ph.D. Dissertation, Massachusetts Institute of Technology. Dept. of Aeronautics and Astronautics., Cambridge, MA.

Kramer, L., and Smith, S. F. 2004. Task swapping for schedule improvement - a broader analysis. In *Proceedings 14th International Conference on Automated Planning and Scheduling (ICAPS 04)*.