An Auction-Based Method for Decentralized Train Scheduling

David C. Parkes and Lyle H. Ungar
Computer and Information Science Department
University of Pennsylvania
200 South 33rd Street
Philadelphia, PA 1910
dparkes@unagi.cis.upenn.edu, ungar@cis.upenn.edu

ABSTRACT

We present a computational study of an auction-based method for decentralized train scheduling. The method is well suited to the natural information and control structure of modern railroads. We assume separate network territories, with an autonomous dispatch agent responsible for the flow of trains over each territory. Each train is represented by a self-interested agent that bids for the right to travel across the network from its source to destination, submitting bids to multiple dispatch agents along its route as necessary. The bidding language allows trains to bid for the right to enter and exit territories at particular times, and also to represent indifference over a range of times. Computational results on a simple network with straight-forward best-response bidding strategies demonstrate that the auction computes near-optimal system-wide schedules. In addition, the method appears to have useful scaling properties, both with the number of trains and with the number of dispatchers, and generates less extremal solutions than those obtained using traditional centralized optimization techniques.

1. INTRODUCTION

Auction-based scheduling methods are well-suited to the decentralized information and control structure of modern railroads. The flow of trains over a railroad network is not controlled by a single centralized scheduler, but rather by the joint decisions of a number of largely autonomous dispatcher agents, each responsible for a local track territory. In addition, trains are operated by competing companies, each of which would prefer for their trains to run on-schedule even if the trains of other companies must wait. Real train drivers receive bonuses for on-line arrivals, and have private information about repair schedules, etc.

Auction-based methods fill two important needs. First, they respect the natural autonomy and private information within such a distributed system. Secondly, they can provide incentives for trains to reveal truthful information (indirectly, via bids) about their values for different schedules. In a naive central implementation, a self-interested train with private information about its time constraints, value, and costs, cannot be expected to act truthfully, but rather to misrepresent this information if it will improve its own schedule in the system-wide solution.

The train scheduling problem that we address in this paper falls within a hierarchy of interrelated train scheduling problems; see [6] for a recent survey. We assume that all *strategic planning*, i.e. deciding on train routes and assigning values, times, and costs, is already completed. Our input is a set of trains, each with a defined routes over a track network, a value for completing its journey, and an optimal departure and arrival time and cost function for off-schedule performance. The system-wide problem is to compute a robust and safe meet/pass schedule for the movement of trains over the network to maximize the total cost-adjusted value over all trains.

In constructing an optimal meet/pass schedule we depart from earlier models for automatic train scheduling that used fixed-priority scheduling rules. These early models have been criticized for a "hurry up and wait" approach [7], with high priority trains moved down lines as fast as possible, possibly causing problems and inefficiencies at yards further down the line. We build instead on the *pacing models* of Kraay *et al.* [7], which control the speed of trains in finding optimal schedules.

Our auction design has each dispatcher agent running a separate auction, for the right to enter and exit its territory at particular times. There are necessarily *multiple* auctions, to respect the autonomy of individual dispatchers to make local decisions. All auctions terminate simultaneously, when there is quiescence across the system. A train agent must bid for pairs of entry and exit times across multiple dispatchers to complete its journey, which presents a coordination problem. The exit time from one dispatcher must be early enough to allow the train to enter the next dispatcher on its route at the required entry time. In order to help in this coordination process, we implement *iterative* auctions to allow trains to adjust towards a good solution, and we allow trains to submit bids for *sets* of times, i.e. "I want to enter your territory at any time after 10am, but leave no later than 1.30pm, and my maximum travel speed is 100km/hr."

This constraint-based bidding language is a concise way to handle the continuous time attribute of a bid without im-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AGENTS'01, May 28-June 1, 2001, Montréal, Quebec, Canada.
Copyright 2001 ACM 1-58113-326-X/01/0005 ...$5.00.

posing an explicit discretization on time, satisfying Nisan's [9] requirements that a language be both *simple* and *expressive*.

In each round the auctioneer computes the set of bids that maximize revenue, subject to the constraint that there must be a feasible meet/pass schedule for trains given the entry and exit times in accepted bids. Prices are maintained over a discrete price lattice on *pairs* of entry and exit times, with prices increased across rounds with an *i*Bundle style price-update [10], i.e. based on the bid prices from unsuccessful agents.

Experimentally, we compare the quality of schedules computed in the auction-based method with schedules computed under a traditional centralized optimization approach. In order to make a fair computational comparison across the methods we formulate both the global scheduling problem and the winner-determination problems as (closely related) mixed integer programs, and solve with CPLEX. We generate a set of stochastic problem instances, and compare the centralized solution (with complete information about agents' problems) with the auction solution. In the auction-based method we make a reasonable assumption about agent bidding strategies, we assume that agents follow a *myopic best-response bidding strategy* and submit bids to maximize value given the current ask prices.

Our initial computational results demonstrate that the auction-based method can generate *better* schedules than the centralized method, and in less time. Moreover, the auction appears to have good scaling properties with the number of agents and dispatchers, at least for the auction parameters selected in our tests (e.g. price update speed, time in each round to solve winner-determination, etc.)

As a word of caution, it also seems likely that the bid-coordination problem will become quite hard in the auction as the number of dispatchers increase, perhaps for routes across five or more dispatchers. The performance of the simple myopic bidding strategy might begin to fall-off in these cases, leaving agents "exposed" to times that they cannot fit with times from other dispatchers. Further study is required to consider alternative, more sophisticated bidding strategies in these cases.

2. THE TRAIN SCHEDULING PROBLEM

We assume that each train has a source and destination node, a value to complete its journey, and a cost for off-time departure and/or arrival. The global objective is to find a safe schedule that maximizes the total *net* value, the total value minus cost of delay across all trains that run. We introduce a novel mixed-integer programming (MIP) formulation that allows trains to be dropped when necessary, i.e. to allow other high-valued trains to run on-time. A very similar formulation is adopted for the winner-determination problem in the auction (see the next section).

2.1 Track Network: Topology and Constraints

In modeling the train scheduling problem we make a number of simplifying assumptions about the network structure and about the types of meets and passes that we allow.

The key assumption is that of a *single line* operation– a sequence of *single-track*, *double-track*, or *yard* sections, separated by *nodes*. This simplifies the specification of the global train-scheduling problem, and also of the winner-determination problem in the auction-based method. In addition, train agents must only consider tradeoffs across multiple temporally different routes, and can ignore alternate paths over the network.

An interaction between a pair of trains may be a *meet* or a *pass*, and is associated with a network location and a time. A **meet** is when two trains traveling in *opposite directions* are at the same location at the same time. A **pass** is when two trains traveling in the *same direction* are at the same location at the same time.

The *feasibility* of a schedule for trains across a network is determined by the *safety* of meets and passes. This depends on the type of section:
(**S1**) Any number of trains can meet and pass in yards.
(**S2**) Any number of trains can pass on double-track sections, but no trains can meet.
(**S3**) No trains can meet or pass on a single-track section.

In addition, a feasible schedule must maintain a *minimum separation distance*, Δ_{safety}, between trains on single and double track sections. We waive this minimum separation distance requirement for trains in yards. Finally, no train can exceed either its maximum speed, or the maximum safe speed on any section.

Allowing trains to pass but not meet on double-track sections reduces problem-solving complexity; intuitively there are many more ways for two trains to cross in the same direction than in opposite directions. Similarly, modeling infinite-capacity yards and double-track sections (sidings) is a simplifying assumption.

2.2 Schedules

A schedule specifies the network position across time for each train in the system. It is sufficient to to consider schedules in which trains travel at a constant speed across each section (the speed can vary from train to train and from section to section), by the following result:

Lemma 1. *Any feasible schedule can be reduced to a feasible schedule where each train travels at a constant speed within each track section.*

The transformation which maintains feasibility is to hold times at nodes between track sections constant, and *smooth* the speed of each train between these points. The proof is quite straightforward– just show that the number of meets are the same for any speed profile consistent with the entry and exit points, and that the number of passes is (weakly) less when trains travel at a constant speed. We ignore constraints on acceleration across sections.

This observation reduces the size of the search space in the scheduling problem, we can simply find optimal times for trains at the ends of each section.

2.3 A Mixed Integer Programming Formulation

Let $\mathcal{I}$ denote the set of trains and $\mathcal{N}$ denote the set of nodes between track sections. It is useful to view the network in an west–east orientation, with nodes ordered that $j > k$ for $j \in \mathcal{N}$ further east than $k \in \mathcal{N}$. We divide the trains into set $east \subseteq \mathcal{I}$ that travel west-to-east and $west$ that travel east-to-west. The nodes are labeled with the type of section to the east, e.g. the section between node j and $j+1$ is a yard if $j \in yard$, single-track if $j \in single$, and double-track otherwise. The minimum travel time for train i between node j and $j+1$, its *free-running time* $r(i,j)$, is

defined by the length of the section, the maximum speed of the train, and the maximum safe speed over the section.[1]

Each train $i \in \mathcal{I}$ has a *source node* and optimal departure time, $(s(i), t_s^*(i))$, a *destination node* and optimal arrival time, $(d(i), t_d^*(i))$, a *value* $V_i \geq 0$ for completing its journey, and a *cost penalty*, $cost_i(t_s, t_d)$, for off-schedule performance. Following [5] we assume a linear additive cost penalty for each train. Given actual source t_s and destination t_d times, the cost for off-schedule performance is computed as:

$$cost_i(t_s, t_d) = C_i \ |t_s - t_s^*(i)| + C_i \ |t_d - t_d^*(i)|$$

where $C_i > 0$ is train i's *marginal cost for off-schedule performance*.

We assume that performance is measured only on the basis of a train's time at its source and destination nodes. This is reasonable for a freight train with a single shipment to make, but less appropriate for a train that must make intermediate scheduled pick-ups and drop-offs.

We specify a schedule with the time, $t(i,j)$, of each train i at node j. This is sufficient to compute the optimal schedule, by Lemma 1. Let $y(i) \in \{0,1\}$ equal 1 if train i is not dropped from the schedule, and 0 otherwise. Let $\Delta_{\text{source}}(i)$ and $\Delta_{\text{dest}}(i)$ denote the *absolute* error in departure and arrival time for train i at source node $s(i)$ and destination node $d(i)$. The system-wide objective is to maximize total value minus cost:

$$\max \sum_i V_i y(i) - \sum_i C_i \Delta_{\text{source}}(i) - \sum_i C_i \Delta_{\text{dest}}(i)$$

The constraints make sure schedules are feasible, i.e. that a schedule is safe, trains are separated, and speed constraints are not violated. In the following ("the big M technique"), M is a large positive number, used to make sure that dropped trains do not restrict schedules for other trains and also to implement disjunctive logic constraints as a mixed-integer program.

Constraints (1a) and (1b) set the errors $\Delta_{\text{source}}(i)$ and $\Delta_{\text{dest}}(i)$ for train i:

$$\Delta_{\text{source}}(i) \geq |t(i, s(i)) - t_1^*(i)| - M(1 - y(i)) \quad \forall i \in \mathcal{I} \quad \textbf{(1a)}$$

$$\Delta_{\text{dest}}(i) \geq |t(i, d(i)) - t_2^*(i)| - M(1 - y(i)) \quad \forall i \in \mathcal{I} \quad \textbf{(1b)}$$

Notice that if $y(i) = 0$ for train i then $\Delta(i) = 0$ is a solution, and we count no penalty for dropped trains. This avoids requiring non-linear terms, such as $C_i \Delta_{\text{source}}(i) y(i)$, in the objective function. The absolute value constraint can be implemented by writing two greater than constraints, one for the positive term and one for the negative term.

Constraints (2a) and (2b) ensure consistency of travel times for trains, given free running times $r(i,j)$ for train i between node j and $j+1$. Again, neither constraint is binding for a dropped train by the "big M" formulation.

$$t(i, j+1) \geq t(i,j) + r(i,j) - M(1 - y(i)) \quad \forall i \in east, \forall j \in \mathcal{N} \quad \textbf{(2a)}$$

$$t(i, j+1) \leq t(i,j) - r(i,j) + M(1 - y(i)) \quad \forall i \in west, \forall j \in \mathcal{N} \quad \textbf{(2b)}$$

We introduce the zero-one variables $gap(i, i', j)$ to make sure that trains are a safe distance apart at all times; $gap(i, i', j) = 1$ iff train i trails train i' by at least time *safety* at node j. We capture with "big M" that either *a train must be more than safety ahead of another train*, **or** *safety behind another train*. Note that constraint (3b) is true whenever at least one of the trains is dropped, so that the times on dropped trains are not constrained.

$$t(i,j) - t(i', j) + M\, gap(i, i', j) \geq safety \quad , \forall j \in \mathcal{N}, \forall i, i' \in \mathcal{I} \quad \textbf{(3a)}$$

$$t(i', j) - t(i,j) + M(1 - gap(i, i', j)) + M(2 - y(i) - y(i')) \geq safety, \forall j \in \mathcal{N}, \forall i, i' \in \mathcal{I} \quad \textbf{(3b)}$$

We introduce the zero-one variables $after(i, i', j)$ to indicate whether train i arrives at node j after train i'; $after(i, i', j) = 1$ if train i is after train i' at node j. This indicator variable is set by constraints (4a) and (4b) to be consistent with the times defined by variables $t(i, i', j)$. A dropped train can assume the same ordering with respect to all trains, allowing them to trivially satisfy (4c) and (4d).

$$t(i,j) - t(i', j) \leq M\, after(i, i', j) \quad , \forall j \in \mathcal{N}, \forall i, i' \in \mathcal{I} \quad \textbf{(4a)}$$

$$t(i', j) - t(i,j) - M(2 - y(i) - y(i')) \leq M(1 - after(i, i', j)) \quad , \forall j \in \mathcal{N}, \forall i, i' \in \mathcal{I} \quad \textbf{(4b)}$$

Constraint (4c) captures the restriction that trains traveling in the same direction cannot pass on sidings or single-track sections. East-bound train i must remain after east-bound train i' at node j if it is behind train i at node $j+1$ and the section between j and $j+1$ is not a yard. Similarly for west-bound trains.

$$after(i, i', j) = after(i, i', j+1) \quad \forall j \notin yard, \forall i, i' \in east, \forall i, i' \in west \quad \textbf{(4c)}$$

Finally, constraint (4d) captures the restriction that trains traveling in opposite directions cannot meet on single-track sections. If east-bound train i is after west-bound train i' at node j it must also have followed west-bound train i' at node $j+1$ for single-track sections between j and $j+1$, otherwise the trains were on the same single-track section at the same time and traveling in opposite directions.

$$after(i, i', j) = after(i, i', j+1) \quad \forall j \in single, \forall i \in east, \forall i' \in west \quad \textbf{(4d)}$$

Taken together with the objective function, the constraints specify a mixed-integer program to solve the centralized train scheduling problem. The optimal solution specifies which trains are dropped (with $y(i) = 0$) and the times $t(i,j)$ for other trains at each node j in the network.

3. AN AUCTION-BASED SOLUTION

Let us assume that the track network is divided across dispatcher territories, with each dispatcher responsible for the local flow of trains. A separate dispatcher agent auctions the right to travel across each territory. Each train is associated with a train agent that places bids for the right to travel across a territory, and coordinates times across dispatchers on its route to achieve a good schedule.

We assume that dispatch territories are separated by neutral yards to allow the safety constraints on meet and constraints to be decoupled across dispatch territories, because yards have infinite capacity and allow arbitrary meets and

[1] Later, when we formulate the schedule problem for winner-determination we will leave this information implicit in the bidding language to simplify our presentation.

passes. The dispatcher on each side of connecting yards must simply ensure that trains remain a *safety* distance apart as they enter and exit their territory.

3.1 Auction Innovations

The nature of the train scheduling problem require a number of innovations in auction design:
(1) A constraint-based bidding language allows trains to submit bids with continuous time attributes, to represent a choice set over different pairs of times, i.e. without discretization.
(2) Prices are maintained over a discrete lattice with quotes computed on-the-fly for any pair of times, i.e. an approximate representation of a continuous and non-linear price space.
(3) The selection of revenue-maximizing bids is built on top of a feasibility check that looks for feasible meet/pass schedules given entry-exit times, i.e. the conflicts across bids are non-trivial and checked via solving for feasible schedules.

As noted earlier we implement multiple independent auctions, one for each dispatcher territory, to respect the decision autonomy of each dispatcher. Given that trains must receive compatible entry-exit times across multiple dispatchers, the auctions are necessarily iterative to allow train agents to coordinate their bids across multiple auctions.

All auctions close simultaneously when bid quiescence is detected across the system.

3.2 Dispatcher Auction

We allow train agents to bid for *entry* and *exit* times in a territory, but leave the dispatcher with the flexibility to decide exactly how a train will run, consistent with those times.

Bidding Language

The bidding language is quite expressive: a train can bid a price to enter a territory at time t_{entry} and depart at time t_{exit}, and state whether the times are *fixed* or *flexible*. With a fixed time the train must enter (or exit) the territory at that exact time. With a flexible entry time, any time *after* t_{entry} is acceptable; with a flexible exit time, any time *before* t_{exit} is acceptable (subject to constraints on a train's minimal travel time). Finally, a train agent can submit multiple bids coupled with an "exclusive-or" constraint, to state that the dispatcher can accept any *one* pair of times from any one bid.

Let $\mathcal{K}$ denote the set of all bids, and $\beta(i) \subseteq \mathcal{K}$ the bids received from agent i. A set of bids from agent i in a particular round are all associated with a single entry node, $n_{\text{entry}}(i)$, a single exit node $n_{\text{exit}}(i)$, and true/false values $\mathit{fixed}_{\text{entry}}$ and $\mathit{fixed}_{\text{exit}}$ to state whether the times are fixed or represent constraints. Each individual bid $k \in \beta(i)$ specifies an entry time, $t_{\text{entry}}(k)$, an exit time $t_{\text{exit}}(k)$, and a bid price $p(k) \geq 0$.

Example:

Bid $(5, 10, \$100)$ xor $(7, 12, \$150)$ for entry node A and exit node B, with $\mathit{fixed}_{\text{entry}}$ but $\neg\mathit{fixed}_{\text{exit}}$, states that the train agent is willing to pay up to \$100 to enter at A at time 5 and depart before time 10, or up to \$150 to enter at time 7 and depart before time 12.

To keep the winner-determination problem tractable we also find it useful to restrict the number of bids that an agent can place in any round.

Winner-determination

In each round of the auction the dispatcher solves the winner-determination problem, computing a provisional allocation to maximize revenue based on bids. The provisional allocation must be consistent with some feasible schedule.

The winner determination problem is formulated as a mixed-integer program, very similar in form to that for the centralized train scheduling problem. However, it tends to be much easier to solve because: the problem is restricted to the space of solutions compatible with the bids submitted by agents; and the problem is for only a single territory.

Mixed-Integer Program Formulation.

Borrowing as much from the earlier global MIP formulation as possible, we introduce new zero-one variables $x(i,k) \in \{0,1\}$ for agent $i \in \mathcal{I}$ and bid $k \in \beta(i)$, the set of bids from agent i, with $x(i,k) = 1$ iff agent i's bid k is in the provisional allocation. The linear objective function is:

$$\max \sum_{i,k \in \beta(i)} p(k)x(i,k)$$

i.e. maximize total revenue where $p(k)$ denotes the bid price of bid k from agent i.

Constraints (2a, 2b, 3a, 3b, 4a, 4b, 4c, 4d) are adopted from the MIP of the global train scheduling problem, with train times computed on the basis of bids from agents. Allowing for flexible bid times, we write:

$$t(i,s(i)) = \sum_{k \in \beta(i)} t_{\text{entry}}(k)x(i,k) \quad \text{, if } \mathit{fixed}_{\text{entry}}(i) \qquad \text{(1a')}$$

$$t(i,s(i)) \geq \sum_{k \in \beta(i)} t_{\text{entry}}(k)x(i,k) \quad \text{, otherwise}$$

$$t(i,d(i)) = \sum_{k \in \beta(i)} t_{\text{exit}}(k)x(i,k) \quad \text{, if } \mathit{fixed}_{\text{exit}}(i) \qquad \text{(1b')}$$

$$t(i,d(i)) \leq \sum_{k \in \beta(i)} t_{\text{exit}}(k)x(i,k) \quad \text{, otherwise}$$

$$\sum_{k \in \beta(i)} x(i,k) \leq y(i) \qquad \text{(1c')}$$

The source node $s(i)$ is the entry node $n_{\text{entry}}(i)$, and the destination node $d(i)$ is the exit node, $n_{\text{exit}}(i)$.

Constraints (1a') constrain the schedule for train i to an entry time consistent with its bid, similarly for (1b') for its exit time. Constraint (1c') ensures that at most one bid is accepted per agent, and that no bids are accepted from dropped trains.

Winner-Determination Cache.

One useful technique to speed-up winner-determination in iterative auctions is to maintain solutions from previous rounds in a cache, indexed against the bids that were submitted. The cache can be checked for a solution before solving the mixed integer program.

In this problem, a *hit* in the cache depends on the *constraints* submitted by agents. Given a set of bids $\mathcal{K}$ from agents, a match is found if a permutation (in terms of the order of agents) of the new bids are *consistent* with a set of bids in the cache. To be consistent:

(1) if bids from agent i are successful in the cached solution, then the new bids from i must *support* the time corresponding to the successful bid, and be (weakly) *less flexible* than the other times in the cached bid.[2]

[2]The other times can be *more* flexible if every agent that bids is in the

(2) if bids from agent i are unsuccessful in the cached solution, then the new bids from i must all be (weakly) *less flexible* than the old bids.

A bid is *less flexible* than another bid if it represents a smaller set of times and at the same price or less, vice-versa for a *more flexible* bid. An exclusive-or set of bids *support* an accepted bid if one or more of the bids in the set is (weakly) more flexible than the accepted bid.

Price Updates

Each dispatcher agent maintains ask prices on a discrete price lattice. This discretization does not limit the times that a train agent can bid because the ask price for a particular bid is determined from the price of the nearest point on the lattice (or minimal over a set of prices in the case of a flexible bid). The lattice structure is used to approximate a continuous non-linear price space. A smaller unit of discretization leads to a higher computational cost and slower convergence but perhaps to a higher schedule quality. We choose this structure for simplicity, another structure might explicitly maintain unsuccessful bids and compute ask prices on-the-fly exactly.

Ask prices represent a *lower-bound* on the price that a train agent must bid to have any chance of success in the auction, but do not guarantee that a bid will be successful. An unsuccessful bid increases the price on its nearest lattice point, or multiple consistent lattice points in the case of an unsuccessful constraint-based bid.

For each bid in an unsuccessful exclusive-or set of bids:
(a) find the point on the lattice closest to the bid, or set of consistent points,
(b) and update the ask price at that lattice point to ϵ above the unsuccessful bid price,
where $\epsilon > 0$ is the *minimal bid-increment* in the auction.[3] The structure of this price-update is motivated by price updates in *i*Bundle [10], an allocatively-efficient ascending-price combinatorial auction that updates prices on bundles of items by ϵ in response to unsuccessful exclusive-or bids.

An "infinite" value is used to represent the case that the *safety* condition will be violated with any bid close to a particular grid point. This is used as items are sold to train agents at particular times (under the continuous clearing rules, see below), to move a train's bid focus away from a time that cannot be accepted at any price.

Price Quotes

The prices on the lattice are used to compute ask prices. The ask price for a fixed pair of times is read off the grid as the price at the closest point. For a bid with a flexible entry time and/or a flexible exit time, the price is computed as the *minimal* price over all compatible grid points.

The prices on flexible times have the following useful semantics:

$$p(k_1) \geq p(k_2), \quad \text{if } k_1 \subseteq k_2$$

for bids k_1 and k_2 if all times consistent with k_2 are also consistent with k_1. This follows immediately from the minimal operator used to compute an ask price under a flexible bid. The relationship allows a train agent to prune its local search when considering different times in its best-response strategy.

Bidding Rules

The bidding rules are quite simple: (1) an agent must bid at least the ask price for a good; and (2) an agent must repeat a bid that supports a pair of times it receives in the current provisional allocation. This ensures that progress is made across individual rounds of the auction.

Clearing and Termination Rules

The auctions terminate simultaneously when no new bids are placed by any train agent to any dispatcher agent. In addition, each auction has a continuous clearing rule, in which a dispatcher commits to a particular pair of times for an agent that receives those times in the provisional allocation for more than a fixed number of successive rounds, T_{clear}. Continuous clearing helps to reduce bidding complexity, committing trains to particular times (although they can continue to bid for alternate times at an additional cost), and focusing search. A countervailing force is that early commits can also lock-in a particular pair of times too quickly when continued search might find a better solution.

The MIP formulation for winner-determination is easily adapted to include committed times. These times can be represented with bids from a dummy agent, with acceptance of those bids forced within the MIP solution method.[4]

4. THE BIDDING PROBLEM

Recall that each train $i \in \mathcal{I}$ has value V_i to complete its journey, subject to a cost $cost_i(t_s, t_d)$ for off-schedule performance, given optimal source and destination times $t_s^*(i)$ and $t_d^*(i)$ and actual times t_s and t_d. The bidding problem is to purchase the right to travel across the network from source to destination at minimal total cost, where cost is the sum of the price it pays in each auction and the cost of off-schedule performance. In addition, if this cost is greater than the train's value then it would prefer to drop out completely.

We assume that each train agent follows a *myopic best-response bidding strategy*, bidding for the schedule that minimizes total cost given the current ask prices. Myopic best-response provides a good starting point to analyze the performance of the auction method. It would be interesting, but probably quite difficult, to also consider the effect of fully strategic agent behavior on the quality of solutions.

4.1 Myopic Best-response Bidding Strategy

The myopic best-response bidding problem can be formulated as a *shortest weighted path problem*. The edges in the graph correspond to pairs of entry-exit times at each dispatcher, fixed or flexible as appropriate. Edges are connected if the exit time on one edge is consistent with the entry time on the next edge. The cost associated with an edge represents the sum of the current ask price, and any cost for off-schedule arrival or departure if the dispatcher is at the source or destination of a train's route.

cache and the prices on the rejected bids from each agent are no greater than on the accepted bids.

[3]We also increase the price based on bids submitted by a train agent that is in the provisional allocation but receives the same pair of times from the last round of the auction and is trying to shift away from that allocation. We allow a train agent to indicate when it is merely repeating a bid because it must under the auction rules, rather than because it really wants that pair of times.

[4]The flexibility of mixed-integer program formulations of winner-determination problems was previously noted by Andersson *et al.* [1].

Formulation

A train's best-response, taking current prices as fixed, is to select a path from source to destination with minimal total cost (or no path in the case that the minimal cost is greater than its value for completing the journey).

Given a set of dispatchers, $\mathcal{D}$, let $(d_1, \ldots, d_n)$ represent the dispatchers on the route of a particular train. Let $C^*_{1\to n}(t)$ denote the minimal total cost to enter dispatcher d_1 no earlier than time t, travel from d_1 to d_n, and exit from dispatcher d_n. This cost represents the cost of the best schedule, given current ask prices and the train's costs for off-schedule performance. The solution to C^* can be computed as a recursive relationship:

$$C^*_{j\to n}(t) = \begin{cases} \min_{\tau > t} \left(c_j(t,\tau) + C^*_{(j+1)\to n}(\tau) \right) & \text{if } j < n \\ \min_{\tau > t} c_j(t,\tau) & \text{if } j = n \end{cases}$$

where $c_j(t_1, t_2)$ is the cost to enter dispatcher j at time t_1 (or no earlier than t_1 in the case of a flexible bid time), and exit dispatcher j at time t_2 (or no later than t_2 in the case of a flexible bid time), computed as the sum of the price for times and any additional cost penalty for off-schedule performance if dispatcher j is at the end of the train's route. The price is the ask-price if the agent is not yet committed to the good (i.e. it has not cleared), or *zero* otherwise (in which case the price represents a sunk cost). Trains consider flexible bid times in the case of non-extremal nodes, but fixed times at source or destination because a cost is incurred for any deviation from optimal departure and arrival times. The intermediate time τ represents the time to cross from dispatcher d_j to d_{j+1}.[5]

Dynamic Programming Solution Method

We use dynamic programming to solve this problem, computing the best solution over a fixed lattice of time points (that can be different for each train agent), and working from dispatcher d_n to d_1, pruning any dominated solutions (for example higher cost edges with earlier entry times). In related work, Boutilier *et al.* [2] proposed a dynamic programming algorithm for agent bidding strategies in *sequential* auctions with complementarities.

We bias search towards solutions consistent with times the train receives in the current provisional allocations. This is quite reasonable because the ask prices on times represents a lower-bound on what might be a successful bid price but does not guarantee that a bid will succeed. That an agent currently receives a pair of times conveys useful information about the "fit" of those times with bids from other agents. This bias reduces problem complexity, limiting the size of the dynamic programming problems to be solved.

We implement the following algorithm:
(1) determine the maximal consistent set of current offers and sold items that also leave enough time for travel across those dispatch territories without suitable times.
(2) for each maximal set, use dynamic programming to determine minimal-cost routes in the gaps of the schedule, i.e. for those contiguous sequences of dispatchers for which the train is not currently holding a suitable pair of entry and exit times.
(3) fill the gaps and select the solution with the lowest total cost (including the cost for current offers/sold items used in the solution).

Whenever a gap occurs at the start or end dispatcher on a train's route the train considers tradeoffs between bid price and the cost of off-schedule performance for off-time departure and/or arrival times.

Finally, given a solution a train will submit as many bids that are consistent with the solution as possible, making use of XOR logic and constraints on times to submit multiple bids without compromising the solution. This increases its own likelihood of success, and also helps the dispatchers to coordinate joint search across multiple agents.

[5]We have finessed detail about the time to travel across yards between dispatch territories, which is simply incorporated into the recursion.

5. EXPERIMENTAL RESULTS

We ran experiments over networks consisting of linear chains of dispatcher territories. Our goal was to compare the quality of final schedules and the computational properties of the auction-based and centralized solutions. Both the global MIP formulation and the MIP for winner-determination in each round of the auction are solved with CPLEX. The rest of the code (myopic best-response, price-updates, etc.) was written in C++.

5.1 Dispatcher model

Each dispatcher territory has the same network structure, as depicted in Figure 1. The total distance is 157.5 km, consisting of a single-track section followed by a double-track section (siding) followed by a single-track section. When chaining dispatcher territories together we connect them with yards. Trains have a maximal speed of 100km/hr over single- and double-track sections, and 1km/hr within a yard. For simplicity, we model maximal speed as 100km/hr throughout the network and re-scale yards from size 0.5km to a model length of 50km.

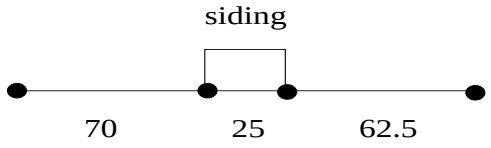


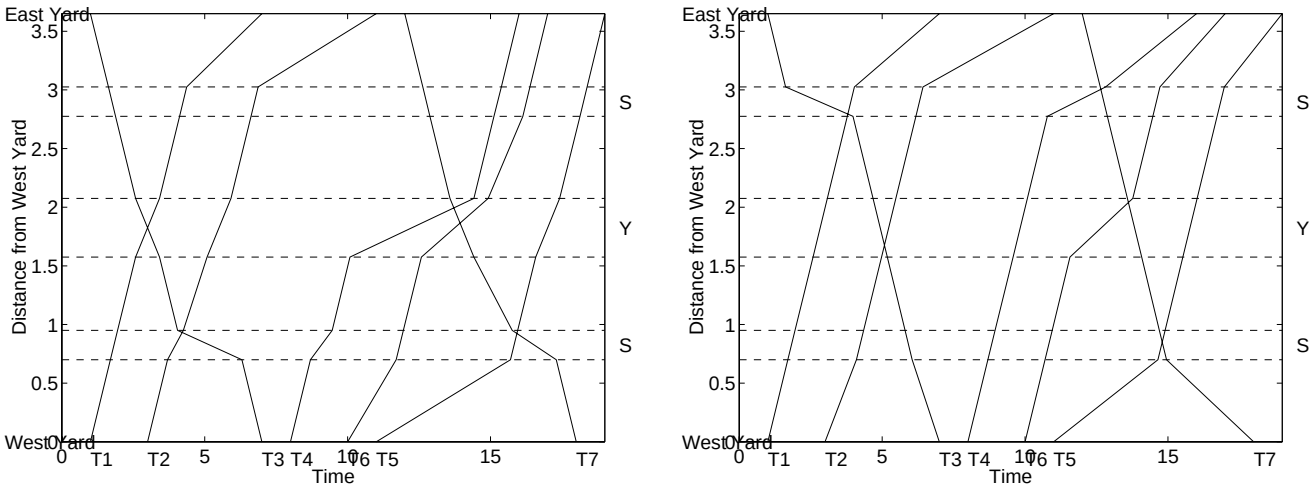
Figure 1: The Network Structure for a Single Dispatcher, with distances of each section (in km).

5.2 Example Problem

Consider a problem with a chain of two dispatchers, and 7 trains, each with value \$200 and cost \$50 per hour of delay. Trains 1, 2, 4, 5, and 6 run east with optimal departure and arrival times (in hours) of { (1, 7), (3, 11), (8, 16), (11, 19), (10, 17) }, while trains 3 and 7 run west with optimal times { (1, 7), (12, 18) }. Given a maximal speed of 100km/hr the *free-running time* of a train across the network is 3.66 hr, i.e. this is how long it would take a train with no delays.

The auction-based and centralized solutions to this problem are illustrated in distance-time charts in Figure 2. Both solutions find optimal solutions, with value \$1400 (all trains run on-time). This problem is quite under-constrained, with a number of possible optimal schedules.

Notice that the auction-based solution is less extremal than the centralized solution. This is quite typical, a result of the fact that train agents tend to bid less extreme times than those selected with a global LP-based branch &



(a) Auction solution.

(b) Centralized solution.

Figure 2: Example: 7 trains, 1 dispatcher territory. Distance in 100's of kms, time in hrs.

bound method such as CPLEX, and achieve a more evenly paced schedule from source to destination. We would expect this property to make auction-based solutions more robust against unexpected minor delays during the execution of a schedule.

5.3 Results

The auction algorithm was parameterized as follows: at most 5 bids per-agent in each round, at most 240 seconds to solve winner-determination in each round (with the best feasible solution used if the optimal solution is not found), a minimal price increment of \$25. The price lattice was maintained over points with a granularity of 0.2 hrs, and we used a time interval size of 0.3 hrs in the trains' dynamic-programming algorithm.

Problem Generator

We propose a stochastic method to generate a set of problem instances. Our approach is loosely based around instances in Kraay et al. [7]. We refer the reader to Hallowell [5] for an account of other interesting train scheduling problem sets in the literature. In this paper we report results on problem sets with between 2 and 4 dispatcher agents and between 5 and 15 train agents. We consider linear networks, formed from dispatcher territories as shown in in Figure 1 and connected with yards. The problem sets are parameterized by constants: prob(E), μ_V, σ_V, μ_C, σ_C, $dep_{\max}$, μ_{slack} and σ_{slack}, as described below.

All trains travel from end-to-end over the network, and travel East with probability prob(E). A train's value is selected from a normal distribution, $V_i \sim N(\mu_V, \sigma_V)$, with mean μ_V and standard deviation σ_V, and its marginal cost C_i for off-schedule performance is normally distributed $N(\mu_C, \sigma_C)$. The optimal departure time for a train, $t_d^*(i)$, is uniformly distributed, $t_d^*(i) \sim U(0, dep_{\max})$. Finally, a train's optimal arrival time, $t_s^*(i)$, is computed so that the *relative slack*, i.e. (available time - free-running time) / free-running time, is normally distributed with mean μ_{slack} % and standard deviation σ_{slack} %.

The complexity of a train-scheduling problem depends on many factors, including the slack time available to each train, the network section types, and the number of "cross-overs". We count a cross-over whenever two trains traveling on-time must cross at some point in the network. As we scaled the problems, with more train agents and more dispatcher agents, we adjusted the $dep_{\max}$ parameter to maintain the same number of average cross-overs per-agent, in an effort to maintain a similar problem complexity. An appropriate $dep_{\max}$ value was computed separately for each problem size based on statistical analysis. Without this adjustment, adding more trains and more dispatchers increases the number of cross-overs and makes problems much more difficult to solve.

We selected prob(E) = 0.7, μ_V = \$200, σ_V = 50, μ_C = \$100, σ_C = 25, μ_{slack} = 100%, σ_{slack} = 25%, and set $dep_{\max}$ to give average cross-over complexity of 2 per-train.[6] We generated 10 problem instances for each problem size.

	Model Size								
	2 dis			3 dis			4 dis		
	5	10	15	5	10	15	5	10	15
Global-time (s)	97	1438	2022	1161	2442	2495	886	2378	3155
Val-Global (\$)	895	1699	2097	854	1561	1177	898	1106	1132
Auc-time (s)	15	792	2568	15	1192	2039	26.9	944	2448
Agent time (s)	0.4	0.6	2.7	0.9	2.2	3.3	1.9	4.4	8.8
Auc-value (\$)	850	1893	1737	842	1855	2632	768	1832	2162
Revenue (\$)	315	698	1045	470	1030	1690	700	1365	2142
# rounds	12	13	25	13	16	18	16	16	23
Cache hit (%)	60	50	30	61	50	47	50	52	37

Table 1: Comparative performance: Auction vs. Centralized methods.

Results

We tabulate results in Table 1, with the computation time of the centralized method bounded at 3600 secs (at which point we take the best available solution).[7] Notice that the

[6] The problem sets are available at http://www.cis.upenn.edu/~dparkes/train.html.

[7] CPLEX also ran out of memory (at 200 MB) in a few problem instances, stopping before 3600 secs but without an optimal solution.

quality of the schedule computed in the auction dominates that from the centralized solution in hard problems, as the number of dispatchers and/or the number of agents increase.

The winner-determination time in the auction (totaled over all rounds and all dispatchers) has reasonable scaling properties, with the number of train agents and in particular with the number of dispatchers. The auction appears able to decompose the problem effectively across dispatchers, such that most computation in terms of coordinating trains is performed by one "critical" dispatcher.

It is noteworthy that the train agent best-response bidding problem is quite easy, indicating that we might experiment with a smaller discrete time step. Finally, notice that the simple cache proves quite effective, finding the optimal solution around 50% of the time.

More experiments are required, both to better understand the average-case scaling properties of the auction, and also to look more deeply at agent strategies. Our current conjecture is that the average-case run time in the auction scales *quadratically* with the number of train agents, and perhaps *sub-linearly* with the number of dispatch territories. In terms of agent strategies, we suspect that some agents purchase times that they cannot use as the number of dispatchers increases, and as the bid coordination problem gets more difficult. This belief is based on a comparison of the revenue with value in the auction, see Table 1. If this is the case it will be necessary to consider more sophisticated bidding strategies to avoid this exposure problem.[8]

6. RELATED WORK

This is not the first study of auction-based methods for train scheduling problems. Brewer & Plott [3], proposed the BICAP ascending-price auction for distributed train scheduling. Our auction is more flexible: while we allow trains to construct arbitrary schedules across the network, BICAP restricts trains to bid from a small set of fixed paths. Market-based methods have also been advocated for other distributed scheduling problems, such as for airport take-off and landing slot allocation problems [11].

Returning to centralized approaches to train scheduling, Kreuger *et al.* [8] have proposed a constraint-based method which appear to have better scaling properties than straight applications of MIP methods, but is perhaps less suited to making tradeoffs across schedules with different qualities.

Wellman *et al.* [12] propose an auction-based method for a factory-scheduling problem, in which agents compete for periods of time on (one or more) shared machines. As in train scheduling, agents often require a combination of time periods, and perhaps across multiple machines. The train scheduling problem is different in nature because it is not possible to define up-front a static set of mutually-compatible times, any of which can be safely allocated to any agent. Instead, a feasible allocation of times to agents must be checked for a safe underlying meet/pass schedule. Our approach is also rather different to that adopted by Wellman *et al.*, since we avoid imposing a discretization of time into finite slots, but allow agents to use a simple and expressive constraint-based bidding language.

[8]A similar exposure problem is noted in the FCC spectrum auction problem, in which agents need sets of compatible licenses, and bid across simultaneous auctions [4].

7. CONCLUSIONS

We have introduced a novel auction mechanism for a distributed train scheduling problem, which is a better match to the natural information and control structure of modern railroads than traditional centralized scheduling solutions. The auctioneer's winner-determination problem is formulated as a mixed-integer program and solved with an off-the-shelf optimization problem, while agents' best-response bidding strategies are solved with a dynamic-programming algorithm. Simulations on a simple network show that the auction-based solution can indeed generate good global schedules, and preliminary empirical analysis suggests favorable computational scaling properties in comparison with a centralized solution.

8. ACKNOWLEDGMENTS

This work represents the culmination of a long term collaborative project with Mei Xue, Patrick Harker, Rinaldo Jose, Julian Kwan, Rebecca Yount, Shai Shen-Orr, Tina Cheung, and Sharon Teo. This research was funded in part by National Science Foundation Grant SBR 97-08965. In addition, the first author gratefully acknowledges financial support from an IBM Fellowship.

9. REFERENCES

[1] A. Andersson, M. Tenhunen, and F. Ygge. Integer programming for auctions with bids for combinations. In *Proc. 4th Int. Conf. on Multi-Agent Systems (ICMAS-00)*, 2000.

[2] C. Boutilier, M. Goldszmidt, and B. Sabata. Sequential auctions for the allocation of resources with complementarities. In *Proc. 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*, pages 527–534, 1999.

[3] P. J. Brewer and C. R. Plott. A binary conflict ascending price (BICAP) mechanism for the decentralized allocation of the right to use railroad tracks. *Int. Journal of Industrial Organization*, 14:857–886, 1996.

[4] M. M. Bykowsky, R. J. Cull, and J. O. Ledyard. Mutually destructive bidding: The FCC auction design problem. *J. of Regulatory Economics*, 2000.

[5] S. F. Hallowell. *Optimal Dispatching Under Uncertainty: With Application to Railroad Scheduling*. PhD thesis, The Wharton School, University of Pennsylvania, 1993. OPIM TR 93-12-02.

[6] D. R. Kraay and P. T. Harker. Real-time scheduling of freight railroads. *Transportation Research-B*, 29B(3):213–229, 1995.

[7] D. R. Kraay, P. T. Harker, and B. Chen. Optimal pacing of trains in freight railroads: Model formulation and solution. *Operations Research*, 39:82–99, 1991.

[8] P. Kreuger, M. Carlsson, and J. Olsson. The TUFF train scheduler– trip scheduling on single-track networks. In *CP97 Workshop on Industrial Constraint-Directed Scheduling*, Linz, Austria, 1997.

[9] N. Nisan. Bidding and allocation in combinatorial auctions. In *Proc. 2nd ACM Conf. on Electronic Commerce (EC-00)*, 2000.

[10] D. C. Parkes. *i*Bundle: An efficient ascending price bundle auction. In *Proc. 1st ACM Conf. on Electronic Commerce (EC-99)*, pages 148–157, 1999.

[11] S. J. Rassenti, V. L. Smith, and R. L. Bulfin. A combinatorial mechanism for airport time slot allocation. *Bell Journal of Economics*, 13:402–417, 1982.

[12] M. P. Wellman, W. E. Walsh, P. R. Wurman, and J. K. MacKie-Mason. Auction protocols for decentralized scheduling. *Games and Economic Behavior*, 2000. To appear.