Minimization of Cost for Transfer Escorts in an Airport Terminal

Elaine Angelino
Shaun Fitzgibbons
Alexander Glasser
Harvard University
Cambridge, MA

Advisor: Michael Brenner

Summary

We minimize the cost for Epsilon Airlines to provide a wheelchair escort service for transfers in an airport terminal. We develop probabilistic models for flow of flight traffic in and out of terminal gates, for the number of passengers on flight who require service, and for transfer destinations within the terminal.

We develop an economic model to quantify both the short- and long-term costs of operating such a service, including the salaries of escorts, the maintenance and storage of wheelchairs, and the costs incurred when late escorted transfers delay a departing flight.

We develop a simulated annealing (SA) algorithm that uses our economic models to minimize cost by optimizing the number and allocation of escorts to passengers. Having indexed the space of all possible escort allocations to be accessible to our SA, we selectively search the space of allocations for a global optimum. Although the space is too large to find a global optimum, our simulations suggest that the SA is effective at approximating this optimum.

Using current airport and airline data, we break our analysis down into short- and long-term costs, simulating escort service operation under dynamic airport conditions, varying air traffic, airport size, and the fraction of traveling population that requests wheelchair-aided transfer (simulating a greater future abundance of elderly travelers).

The UMAP Journal 27 (3) (2006) 349–365.

Introduction

We describe, quantify, and optimize a cost-effective approach to wheelchair-aided escorted transfers at Epsilon Airlines.

We investigate the airport layout, particulars of flight travel and passengers, and the economics of the escort service. We use a simulated annealing (SA) algorithm to find a cost-optimized solution, so that Epsilon's many satisfied customers will continue to say: "Epsilon will always be in my neighborhood!"

Definitions and Key Terms

- An **airport terminal** is an ordered pair $(\tau, d_{\text{terminal}})$, where τ is an ordered collection of concourses each of which is an ordered collection of gates, and d_{terminal} is a metric defining the distance between any two gates of the terminal (**Figure 1**).
- A **concourse** is an ordered collection of gates.
- A **gate** is an element of a concourse at which *arrivals* and *departures* take place. A gate's position in a terminal is determined by an ordered pair (concourse#, gate#)
- A **wheelchair passenger (WP)** is a passenger who arrives at a gate, has a connecting flight at another gate, and who can move between gates only by means of a wheelchair pushed by an airline escort. A WP normally notifies the airline in advance of the need for an escort.
- An **escort** is an airline employee who wheels WPs to connecting flights, with salary per shift of P_E.
- A **shift** is an 8-hour period during peak operating hours of the terminal.
- A **transfer job** is an ordered 4-tuple

$$(\text{concourse} \times \text{gate}_{\text{arr}}, \text{time}_{\text{arr}}, \text{concourse} \times \text{gate}_{\text{dep}}, \text{time}_{\text{dep}})$$

describing the time and place of arrival and of departure of a WP.

- The **delay time** of a departure is the time difference between the actual departure time and the scheduled departure time of a given flight, given that the actual departure time is later. The actual departure time is when the last WP booked on the flight arrives at the departure gate.
- A shift's **transfer schedule** is the set of escort-required transfers that are scheduled throughout the shift. New WP announcements and unexpected non-escort-related delays cause the transfer schedule to be updated throughout the day.

- An **escort job list** is the ordered set of transfer jobs performed by a particular escort over the length of the shift.
- An **escort allocation algorithm** is a general method of determining the number of escorts and specifying the escorts' behavior throughout a shift.

Model Assumptions and Key Concepts

Geometry of an Airport Terminal

We demonstrate our optimization over one airport terminal geometry (with a varying number of concourses.)

We represent a terminal by the union of a central hub with the cartesian product of n concourses and m gates per concourse: $\{0\} \cup \{1, ..., n\} \times \{1, ..., m\}$. (**Figure 1**). We denote the distance between the start of the concourse to the concourse's first gate, as well as the distance between any two adjacent gates of a concourse, by r_g, and the distance from the central hub to the start of each concourse by r_h. The distance between any two *non-hub* points (n_1, m_1) and (n_2, m_2) of a terminal is given by the metric

$$d_{\text{terminal}} = \delta(n_1, n_2)|r_g(m_1 - m_2)| + (1 - \delta(n_1, n_2))|r_g(m_1 + m_2) + 2r_h|.$$

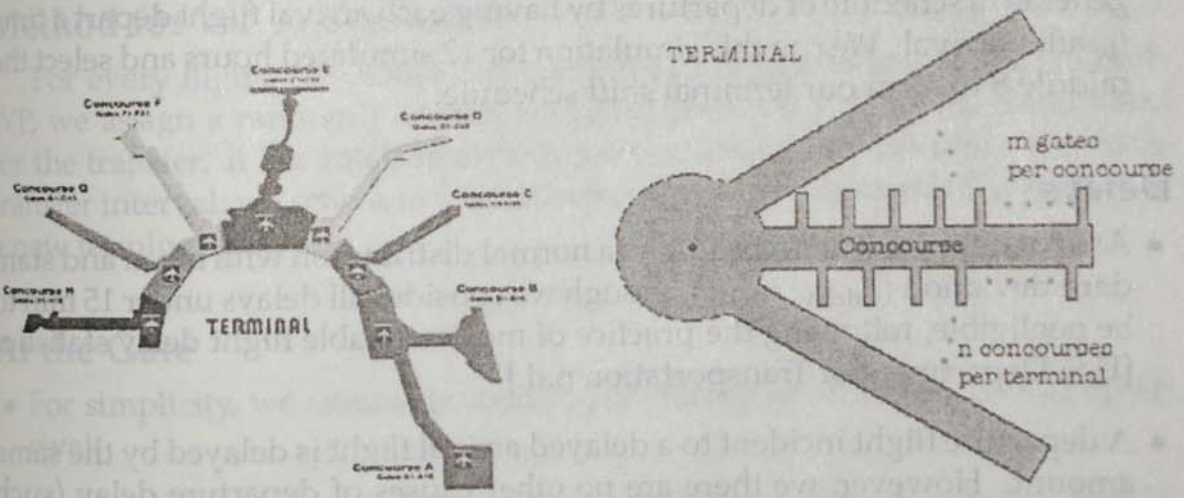

Figure 1. Terminal map of Miami International Airport [n.d.] and a schematic 2-D diagram of our essentially 1-D representation of an airport terminal.

Flight Specifications

Flow of Air Traffic at Terminal

- We take as *departure time* the latest time that a passenger can board a flight. We define a parameter t_{sit} to be the time that a plane sits at a gate after boarding and before takeoff.

- Every arrival flight becomes a departure flight at the same gate some fixed time later, and we call this fixed interval of time the *turnover time* t_{to}. We also call the departure flight *incident to* the arrival flight.
- We define a parameter O_G, a terminal's *gate occupancy*, as the fraction of terminal gates at which planes are docked at any given time.
- We describe the flow of flight traffic through a terminal via five parameters:
  - the number n of concourses per terminal,
  - the number m of gates per concourse,
  - the sit time t_{sit},
  - the turnover time t_{to}, and
  - the terminal gate occupancy O_G.

  The time between terminal arrivals is

$$t_{\text{gap}} = \left\lceil \frac{t_{\text{to}} + t_{\text{sit}}}{mnO_G} \right\rceil.$$

- We generate a terminal's daily flight schedule as follows: We begin with no airplanes docked at the terminal, and let them arrive one at a time, each at a time t_{gap} apart, randomly assigning them to free gates as they come in. We generate a schedule of departures by having each arrival flight depart a time t_{to} after arrival. We run this simulation for 12 simulated hours and select the middle 8 to form our terminal shift schedule.

Delays

- Arrival delays occur according to a normal distribution with mean and standard deviation $(\bar{x}_{\text{delay}}, \sigma_{\text{delay}})$, though we consider all delays under 15 min to be negligible, following the practice of most available flight delay statistics [U.S. Department of Transportation n.d.].
- A departure flight incident to a delayed arrival flight is delayed by the same amount. However, we there are no other causes of departure delay (such other delays are tangential to the problem of transfer escorts).

Transfer Specifications

WPs per Flight

- We fix a parameter P_F as the number of passengers per flight.
- We let a parameter S_{transfer} be the fraction of passengers who transfer from a flight upon arrival, and a parameter $S_{\text{wheelchair}}$ be the fraction who request wheelchair assistance. Therefore, the probability that a given passenger

requires wheelchair service (the probability of a WP) is $P(\text{WP}) = S_{\text{transfer}} \times S_{\text{wheelchair}}$. (We neglect passengers who require wheelchair service but do not transfer.) Given this probability, we determine the probability of r WPs on a flight, assuming a binomial distribution:

$$P(r) = \frac{P_F!}{r!(P_F - r)!} P(\text{WP})^r [1 - P(\text{WP})]^{P_F - r}. \quad \textbf{(1)}$$

We cap r at 3, since the probability of more is negligible.

Allowable Transfers

- Given a schedule of n transfers, and a general escort allocation algorithm, we define the ith component of a delay time n-vector $\vec{dt}$ to be the delay time of the ith transfer.
- We fix a buffer time t_{buffer} such that no airline will book a transfer time less than t_{buffer}.
- We fix a greatest allowable transfer time t_{max}, to capture the fact that the transfer time between flights is not usually very long.
- The transfer time is uniformly distributed between t_{buffer} and t_{max}.

Method for WP Assignment

For every flight, we choose the number of WPs according to **(1)**. To each WP, we assign a randomly chosen occupied terminal gate as the destination for the transfer. If the gate's flight's departure time falls within the allowable transfer interval, we schedule the transfer; otherwise, we search at random for a new terminal gate.

At the Gate

- For simplicity, we assume immediate deplaning of all passengers upon arrival.
- An escort may drop off a WP at a gate any time before departure. That is, WPs can board without the escort and escort's wheelchair.
- The time that it takes a WP to load in and out of a wheelchair is negligible.

Traversing the Terminal

- An escort who completes a transfer job moves directly toward the arrival gate of the next job, even if required to wait there for the flight's arrival. An escort always takes the shortest route and has complete informational contact with the manager assigning jobs.

- Escorts have two walking speeds: v_{walk} (with an empty wheelchair) and $v_{\text{wheel}} = \frac{3}{4} v_{\text{walk}}$ (with an occupied wheelchair).

Wheelchairs and Wheelchair Storage

A wheelchair in the hands of an escort presents no liability risk to other terminal traffic. Therefore, we stipulate that an escort walks around with a wheelchair at all times.

Economic Model of an Escort Service

Associated Costs

- Flight delays: If there are not enough escorts, our service will cause flight delays.
- Salary of escorts.
- Maintenance of wheelchairs.
- Storage of wheelchairs: Wheelchairs cannot be left lying haphazard about a terminal before or after a shift—they pose a liability risk. So we stipulate that outside of the escort shift, wheelchairs are stored in a storage facility.

Quantification of Escort and Wheelchair Costs

- Let K be the number of escorts and P_E be an escort's pay per shift.
- Let $W = K$ be the number of wheelchairs and let P_W be their maintenance cost per shift.
- Let A_W be the area that a wheelchair takes up in storage, and let R_A be the daily rent of airport storage space per square foot.

The Short Term Dollar Cost of Delay Time

- There is no effect of delay time on ticket sales.
- Escorts are paid on an annual basis, independent of extra hours logged due to flight delay. Therefore, delay times incur no overtime costs.
- The only short-term cost of flight delay, therefore, is the high cost of operating the plane during the delay.
- Therefore, given operating cost C_{pl} of a plane (\$/min) and the delay time dt of a flight (min), the short-term cost of a flight delay for duration dt is $C_{\text{pl}}\, dt$.

The Long Term Dollar Cost of Delay Time

- We attribute the long-term cost of delay time to two factors: the added operating cost of planes, and a long-term reduction in ticket sales.

- We quantify the reduction in ticket sales. Consider the standard Microeconomics 101 two-good problem: A consumer is faced with purchasing t_ϵ flight tickets on Epsilon Airlines or t_A flight tickets on Airline A. Assume that Epsilon Airlines and Airline A are indistinguishable except for ticket prices (P_ϵ, P_A) and total average delay times per shift (D_ϵ, D_A). The dollar-utility function of a consumer is

$$U(t_\epsilon, t_A) = E_{\text{pa}}(t_\epsilon + t_A) - P_\epsilon t_\epsilon - P_A t_A - C_{\text{pa}}(D_\epsilon t_\epsilon + D_A t_A).$$

where

  - t_i is the number of airline$_i$ tickets purchased per shift;
  - P_i is the price of an airline$_i$ ticket;
  - E_{pa} is the dollar utility that a consumer receives from a flight (assumed to be the same for all flights, for simplicity);
  - C_{pa} is the dollar utility of passenger time in dollars/time; and
  - D_i is the average WP-related delay time per shift of airline$_i$.

Optimizing this dollar utility with respect to the t_is, we find:

$$\frac{\partial U}{\partial t_\epsilon^*} = 0 \Longrightarrow P_\epsilon = E_{\text{pa}} - C_{\text{pa}} D_\epsilon, \qquad \frac{\partial U}{\partial t_A^*} = 0 \Longrightarrow P_A = E_{\text{pa}} - C_{\text{pa}} D_A.$$

Note that

$$D_\epsilon = \frac{\sum_i dt_i}{F_{\text{shift}}},$$

where the index i runs over all the delay times of all transfers in a shift, and F_{shift} is the number of flights per shift. Now, recalling that P_F is the passengers per flight, we find that

$$P_F F_{\text{shift}} P_\epsilon = E_{P\text{pa}} P_F F_{\text{shift}} - C_{\text{pa}} P_F \sum_i dt_i.$$

Since the product on the left side of the equation is total revenue per shift, we can associate the summands on the right with all contributing subrevenues and subcosts. (These numbers denote the contribution of single shifts to long-term revenue, after ticket prices have responded to changing market demand and market demand has responded to airline conditions—delay times, etc.). The effect of transfer delay times $\{dt_i\}_i$ per shift on long term revenue is given by $-C_{\text{pa}} P_F \sum_i dt_i$.

Summary: The Economic Model

By the above arguments, we may finally write the short- and long-term expense functions of our escort service per shift:

$$E_{\text{st}} = KP_E + KP_W + KA_W R_A + C_{Pl} \sum_i dt_i,$$

$$E_{\text{lt}} = KP_E + KP_W + KA_W R_A + C_{Pl} \sum_i dt_i + C_{Pa} P_F \sum_i dt_i.$$

Expectations of Our Model

We expect our model to exhibit the following behaviors:

- It should optimize allocation of escorts by minimizing costs.
- As passenger traffic increases, the number of escorts should increase.
- The number of escorts should increase with the number of concourses.
- Since long-term costs affect ticket costs but short-term costs do not, the number of escorts that optimizes long-term budget concerns should exceed the number that optimizes short-term budget concerns.
- The number of escorts should increase with life expectancy.

Annealing-Based Optimization

We consider the complete-information case where the transfer schedule is fully specified at the beginning of a shift.

Reduction to an Ordering Problem

We use the assumptions about escorts to reduce the space of escort deployments to a simple permutation group. In this formulation, with just a single escort, the problem of finding an optimal allocation is equivalent to finding an optimal hamiltonian path (a path in an undirected graph which visits each node exactly once) through a fully connected graph (one in which each pair of nodes is connected by an edge).

We define a *job node* as a transfer job indexed in a way convenient for our algorithm. It is characterized by arrival time (t_1), arrival gate (x_1), departure time (t_2), and departure gate (x_2). We index all this information by two numbers: node type (1 for a *job* node—we will see later that other node types are required to solve the multi-escort problem) and its position in the transfer schedule. We

Table 1.

Variables and parameters.

Variable	Meaning	Units	Estimate
n	Number of concourses per terminal		2
m	Number of gates per concourse	s	20
O_G	Average terminal gate occupancy		50%
K	Number of escorts hired		2
$W = K$	Number of wheelchairs		2
$S_{\text{wheelchair}}$	Fraction of passengers who require a wheelchair		0.4%

Parameter	Definition	Value	Units	(source)
P_E	Escort salary per shift	80	\$/shift	(1)
r_g	Distance between adjacent gates	30	m	(2)
r_h	Distance between hub and concourse	30	m	(3)
t_{sit}	Time a plane remains at gate after boarding	10	min	(4)
t_{to}	Time between arrival and incident departure	60	min	(5)
$\bar{x}_{\text{delay}}$	Mean flight delay time	35.6	min	(6)
σ_{delay}	Standard deviation of flight delay time	24.4	min	(7)
P_F	Passengers per flight	90		(8)
S_{transfer}	Fraction of passengers who transfer	0.5		(9)
t_{buffer}	Minimum allowable transfer	30	min	(10)
t_{max}	Maximum allowable transfer	120	min	(11)
v_{walk}	Average human walking speed	1.3	m/s	(12)
v_{wheel}	Speed at which escort wheels full wheelchair	1	m/s	(13)
P_W	Maintenance cost of wheelchair per shift	1	\$	(14)
A_W	Floor area that a wheelchair takes up	9	sq. ft	(15)
R_A	Daily rent of airport commercial real estate	2	\$/sq. ft	(16)
C_{pl}	Operating cost of a plane	1495	\$/h	(17)
C_{pa}	Value of passenger time	44	\$/h	(18)

Sources for Parameter Values

(1) `http://www.avjobs.com/table/airsalry.asp`
(2) Half the Boeing–747 wingspan (since gates alternate sides of concourse), `http://airportbusiness.cygnus.proteus.com/article/article.jsp?id=1291\&siteSection=1`
(3) Estimated to be same as (2)
(4) Approximate time for flight-attendant preparation, from personal experience
(5) Approximate time to fuel a Boeing-747, `http://www.uk-trucking.net/?page=news\&mode=view\&id=113`
(6) `http://news.bbc.co.uk/1/hi/business/1833213.stm`
(7) `http://news.bbc.co.uk/1/hi/business/1833213.stm`
(8) `www.faa.gov/library/reports/delay_analysis/media/DCOS1995.doc`
(9) Gatersleben, Michel R., and Simon W. van der Weij, Analysis and simulation of passenger flows in an airport terminal, in *Simulation—A Bridge to the Future, Proceedings of the 31st Conference on Winter Simulation*, vol. 2, 1226–1231; 1999
(10) Common airport practice
(11) Common airport practice
(12) `http://phyun5.ucr.edu/~wudka/Physics7/Notes_www/node18.html`
(13) 3/4 walking speed of (12)
(14) `http://www.1800wheelchair.com/asp/view-category-products.asp?category_id=498`, roughly annually replaced
(15) Measured at home
(16) `http://rebuildnewyork.nreimag.com/ar/real_estate_cw_report_new/`, assuming that concourse rent is comparable to highest NYC commercial rents
(17) `www.faa.gov/library/reports/delay_analysis/media/DCOS1995.doc`
(18) `www.faa.gov/library/reports/delay_analysis/media/DCOS1995.doc`

refer to such a node as job node i, and refer to the information that it contains with double indices (i.e., x_{i1}).

In general, we have multiple escorts. For k escorts, we try to find an optimal set of $\leq k$ distinct paths through the graph defined by the set of job nodes such that every job node is included in exactly one path. In this formulation, the problem is formidable. However, a reformulation transforms the problem back into a straightforward hamiltonian-path optimizing problem [Bellmore and Hong 1974]. An *escort node* is characterized (and indexed) by two numbers: its node type (0 for an escort node) and its placement within the ordered list of escorts. Adding $k-1$ escort nodes to our graph of job nodes partitions the transfer schedule into individual escort job lists for every hamiltonian path.

We now define a tool for labelling hamiltonian paths through this extended graph. A *configuration* is an ordered sequence containing exactly one each of the job and escort nodes of the extended graph.

Traveling from a job node to an escort node ends the escort's job list and opens up the job list for the escort indexed by the escort node that is landed on. The first escort in the list is always assumed to be selected at the beginning of the node sequence. Traveling to a job node from any node type, by contrast, adds that job to the escort job list of the current escort. This implies that any time that an escort node occurs at the beginning/end of a configuration, or two escort nodes occur consecutively within a configuration, an escort's job list is terminated with no jobs—equivalent to that escort never being hired for the shift. This formulation allows us to optimize over the number of escorts hired (up to a certain maximum number of escorts) simultaneously with optimizing over escort job assignments.

Having thus formulated our problem as a choice of an optimal hamiltonian path through a graph of $n+k-1$ nodes, we move on to solution via simulated annealing.

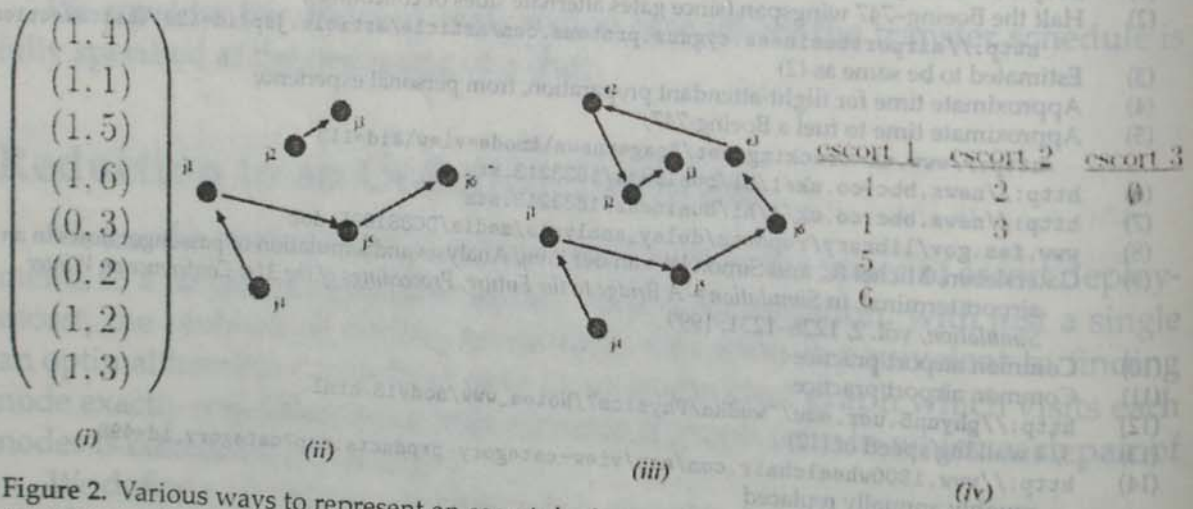

Figure 2. Various ways to represent an escort deployment configuration: (i) configuration vector, (ii) $k \leq 3$ paths through the graph of job nodes such that every job node is contained in exactly one path, (iii) hamiltonian path through the graph of job and escort nodes, and (iv) specification of each of the ($k = 3$) individual escort job lists.

Simulated Annealing Preliminaries

The inspiration for the simulated annealing (SA) approach is found in nature, in particular in the phenomenon of observable crystallization that occurs in certain systems in solid state thermodynamics when such a system is slowly cooled. The underlying idea is that if a system is cooled slowly enough from a disordered configuration at high temperature, it will tend strongly toward organizing itself into a highly optimized low-energy state. The subsequent discussion of annealing methods is adapted from Bentner et al. [2001].

One computes the number of microstates available to the total system. The fundamental assumption is that all available microstates occur with equal probability, given by the Boltzmann distribution. Normalizing this weighting distribution to obtain a probability distribution gives for microstate σ

$$\pi(\sigma) = \frac{1}{Z} \exp\left(-\frac{H(\sigma)}{k_B T}\right),$$

where

$$Z = \sum_{\tau} \exp\left(-\frac{H(\tau)}{k_B T}\right)$$

and $H(\sigma)$ is the energy of the system when in microstate σ. The function H is the *hamiltonian function* for the system. If we cool the system slowly enough to $T = 0$ so that equilibrium holds at all times (adiabatic cooling), then at $T = 0$ the system will be "frozen" into a lowest energy microstate.

For our problem, the microstates are configuration vectors, and our hamiltonian is the expense function (E_{st} (short-term) or E_{lt} (long-term). If we can simulate an adiabatic cooling, starting from an initial random microstate and moving through a trajectory of subsequent microstates, then we should obtain a deployment with lowest cost.

The Metropolis Criterion

Simulated annealing essentially runs on a trial-and-error (Monte-Carlo) scheme, aided by a clever choice of "allowed moves" from each microstate. At each iteration, an allowed move is randomly selected and a decision is made whether or not to make that move. We apply the *Metropolis criterion*:

$$p(\sigma \to \tau) = \begin{cases} \exp\left(-\dfrac{\Delta H}{k_B T}\right), & \text{if } \Delta H > 0; \\ 1 & \text{otherwise.} \end{cases}$$

Since our problem is nonphysical, we replace $k_B T$ in the formula with a control parameter T in units of money.

To lower temperature, a logarithmic process is generally used. We update the temperature after each attempted configuration step with the rule $T_{\text{new}} = \alpha T_{\text{old}}$, where α is usually taken to be in the range $(0.8, 0.999)$ [Bentner 2001].

Choosing a Starting Temperature

Case 1

The idea is to start at a temperature sufficiently high that initially the trajectory is free to move across the configuration space. In this case, if α is close enough to 1 and the set of allowable transitions is chosen well, then the algorithm should slowly settle very near to the global minimum of the hamiltonian. The initial temperature can be chosen by performing a random walk across the configuration space with the defined set of configuration transitions (i.e., iterating the annealing algorithm from the initial condition with T fixed at ∞). Multiply the observed maximum of the ΔH by 10 to obtain an initial temperature T_s very likely to be sufficiently high to meet the above description.

Case 2

If a preprocessing step is added (i.e., use a naive algorithm to presort jobs among escorts), then it may no longer be desirable to start with so high a temperature. If there is a reason to be confident that the start is in the general vicinity of an optimum, then a good starting temperature may be the dollar value of the initial condition or even half that.

Our Hamiltonian

We define the hamiltonian in terms of an algorithm that takes a given configuration as an input. This algorithm can be viewed in the flowchart in **Figure 3.** We define some of the objects found in the chart.

We define two time functions that act on job nodes.

- The first one acts on just a single job node:

$$t_{\text{dep}}\big((1,i)\big) = \frac{d(x_{i1}, x_{i2})}{v_{\text{wheel}}}.$$

This simply defines the amount of time that it takes for an escort to perform a job from pickup to drop-off.

- The second one acts on a pair of job nodes:

$$t\big((1,i),(1,j)\big) = \frac{d(x_{i1}, x_{i2})}{v_{\text{wheel}}} + \frac{d(x_{i2}, x_{j1})}{v_{\text{walk}}}.$$

This measures the time to perform the job on the first node from pickup to drop-off and then further to walk to the arrival gate for the second job.

The algorithm starts at the $k = 1$ box. Where multiple arrows leave a box, all of those actions are performed, right to left (or up to down for horizontal arrows).

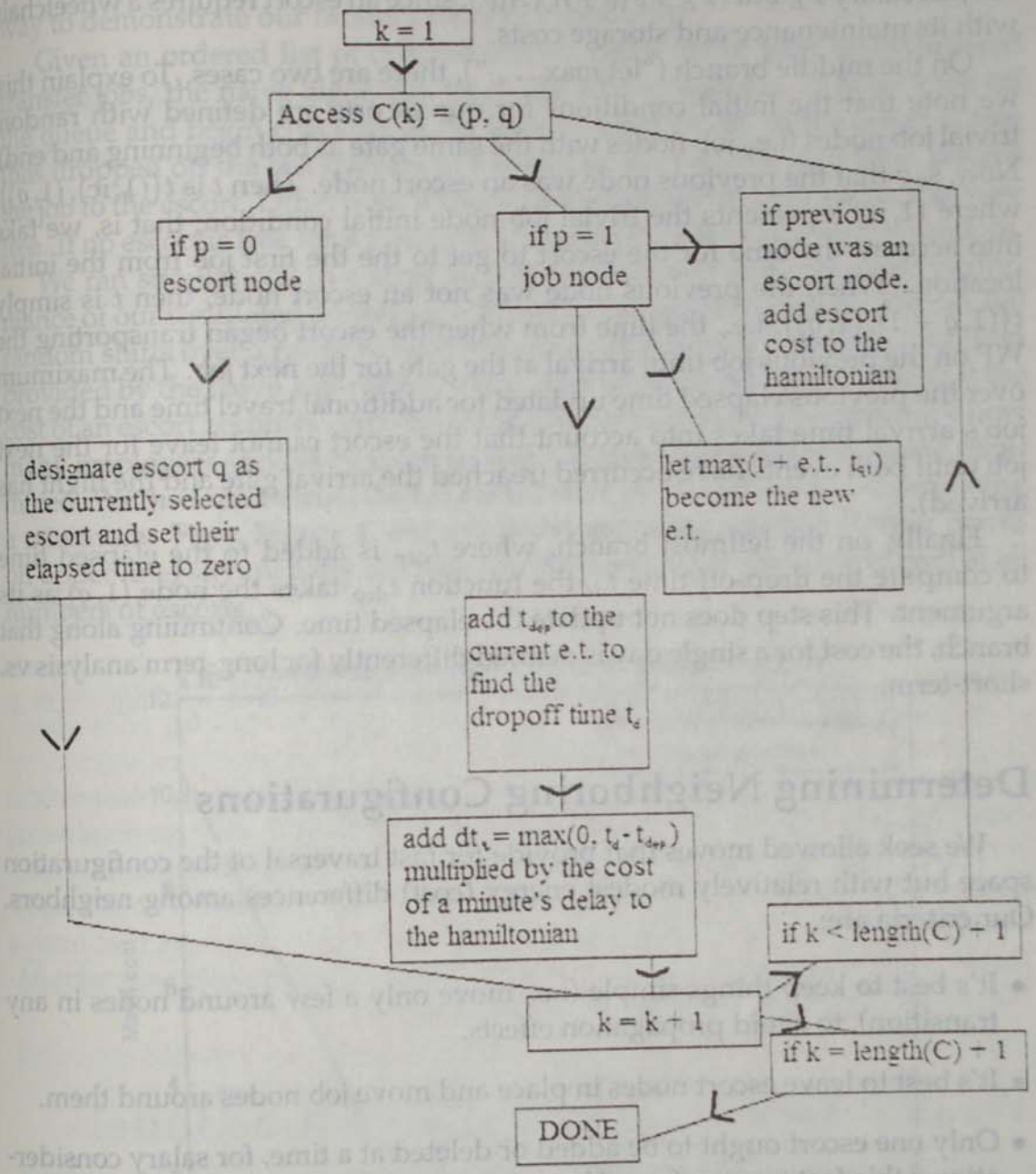

Figure 3. Flowchart outlining the algorithm for determining the hamiltonian value (cost) for a given input configuration.

The job node branches require a little more explanation. On the rightmost branch, when the job node was preceded by an escort node, we add the cost of an escort to the total cost incurred by the configuration. The cost of an escort is not just salary P_E but $(P_E + P_W + R_A A_W)$, since an escort requires a wheelchair, with its maintenance and storage costs.

On the middle branch ("let max ... "), there are two cases. To explain this, we note that the initial conditions for our escorts are defined with random trivial job nodes (i.e., job nodes with the same gate as both beginning and end). Now, say that the previous node was an escort node. Then t is $t\big((1, \mathrm{ic}), (1, q)\big)$, where $(1, \mathrm{ic})$ represents the trivial job node initial condition; that is, we take into account the time for the escort to get to the the first job from the initial location. When the previous node was not an escort node, then t is simply $t\big((1, q-1), (1, q)\big)$, i.e., the time from when the escort began transporting the WP on the previous job until arrival at the gate for the next job. The maximum over the previous elapsed time updated for additional travel time and the next job's arrival time takes into account that the escort cannot leave for the next job until both events have occurred (reached the arrival gate and the flight has arrived).

Finally, on the leftmost branch, where t_{dep} is added to the elapsed time to compute the drop-off time t_d, the function t_{dep} takes the node $(1, q)$ as its argument. This step does not update the elapsed time. Continuing along that branch, the cost for a single day is defined differently for long-term analysis vs. short-term.

Determining Neighboring Configurations

We seek allowed moves that provide for fast traversal of the configuration space but with relatively modest energy (cost) differences among neighbors. Our criteria are:

- It's best to keep things simple (i.e., move only a few around nodes in any transition), to avoid propagation effects.
- It's best to leave escort nodes in place and move job nodes around them.
- Only one escort ought to be added or deleted at a time, for salary considerations; this feature is enforced by our move types.

To avoid large changes in time ordering and propagation effects, only job nodes with departure times within a threshold of each other (we used 40 min) may be swapped. Likewise, a job node may be transferred only to a position where the job node ahead of it has a departure time within 40 min of its own.

Model Testing: Simulations and Discussion

We present a naive test algorithm that simplistically assigns escorts, as a way to demonstrate our model's relative superiority.

Given an ordered list of ordered escorts and a list of temporally-ordered transfer jobs, the naive algorithm treats the jobs as a queue. It takes a job off the queue and begin checking at the least-ordered escort. If the escort is free (has dropped off the last transfer or has had no job yet), the algorithm assigns the job to the escort. Otherwise, it checks if the next escort in order is otherwise free. If no escort is free for a given job, it assigns the job to an escort at random.

We ran simulations in a large airport environment and compared performance of our simulated annealing solution to this naive algorithm. We ran 100 random shifts over a range of numbers of escorts (using the starting condition provided by the naive algorithm), while modifying the hamiltonian so that the cost of an escort becomes zero. Essentially, we are simulating to determine how many employees should be hired so that (on an average basis) costs are minimized. We then reconstruct the full costs before plotting them and determining a minimum. From **Figure 4**, our simulated annealing solution (corner points on upper line) significantly outperforms the naive algorithm (lower line) at all numbers of escorts.

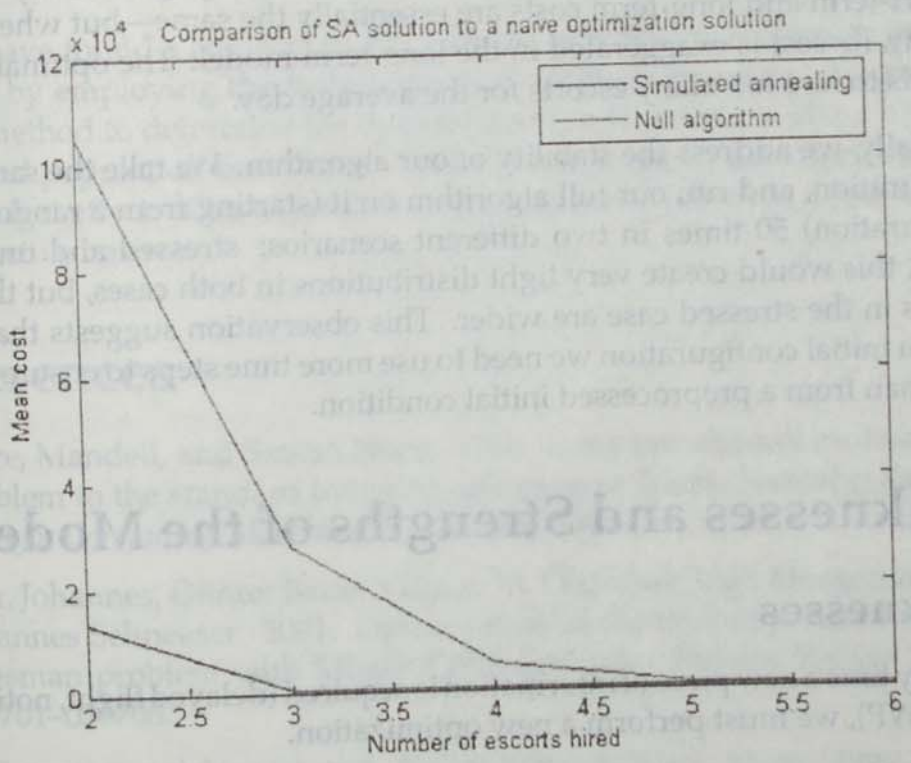

Figure 4. Trials for four concourses of 25 gates each, half the gates occupied by planes, 0.4% wheelchair passengers, and short-term cost analysis. For any number of escorts, the simulated annealing solution (corner points on lower line) yields a lower cost than the naive algorithm (upper line) because it is more efficient in allocating jobs. The apparent mean costs of 0 are an artifact of the large scale (1 vertical unit = $10,000); the optimal number of escorts in the simulated annealing solution is 4, with average cost about $475.

To check our model's response to the dynamic changes, we ran our algorithm for

- increasing concourses per terminal (airport size);
- increasing average gate occupancy (air traffic);
- increasing S_{WC}, simulating a future increase in life expectancy and thus an increase in abundance of elderly among air travelers;
- short-term vs. long-term.

We find that

- the optimal number of escorts increases from 1 in the small single-concourse terminal to 2 for the two-concourse terminal and to 4 for the large four-concourse terminal;
- as terminal traffic increases, the required number of escorts increases.
- increasing the percentage of passengers requiring wheelchair assistance increases the number of escorts needed;
- when there are enough escorts so that there is negligible delay time, the short-term and long-term costs are essentially the same—but when there is delay, its cost is exaggerated in the long-term model. The optimal solution tends to hire too many escorts for the average day.

Finally, we address the stability of our algorithm. We take the same initial configuration, and run our full algorithm on it (starting from a random initial configuration) 50 times in two different scenarios: stressed and unstressed. Ideally, this would create very tight distributions in both cases, but the distributions in the stressed case are wider. This observation suggests that from a random initial configuration we need to use more time steps to ensure repeatability than from a preprocessed initial condition.

Weaknesses and Strengths of the Model

Weaknesses

- Every time a new piece of information is acquired (delayed flight, notification of a WP), we must perform a new optimization.
- Another weakness involves the hamiltonian function. There is a range of configurations in which delay time is zero (and the number of escorts is the same). When the hamiltonian is constant across a large set, the annealing algorithm is designed to take any proposed move that keeps it in the set, essentially a random walk. In many cases, we have the ability to specify a

preference between two configurations with a delay time of zero; we prefer as many escorts as possible to be as early as possible on their deliveries.

- We do not have an optimal method to step through configuration space.

Strengths

- Our algorithm finds good allocations even for few escorts, and effectively reduces cost better than a reasonable alternative algorithm.
- Our model is robust. It easily accommodates any airport, and any number of wheelchair passengers per flight, and does so with speed. As a result, our model can easily be applied to any one of Epsilon Airline's fine terminals.
- Another feature of our model is that the coarseness of the optimization can be tuned with the logarithmic parameter α (i.e. by increasing the number of steps taken, α is made successively closer to 1, making our simulated annealing more and more adiabatic).

Conclusion

We have found a solution to the problem of wheelchair transfer escort allocation by employing the techniques of simulated annealing. We present a robust method to determine the optimal number of escorts, under a variety of traffic and population conditions. We have also mapped out short- and long-term budget effects. Our algorithm assigns escorts to jobs—on a real time and cost-effective basis.

References

Bellmore, Mandell, and Saman Hong. 1974. Transformation of multisalesmen problem to the standard traveling salesman problem. *Journal of the Association for Computing Machinery* 21 (3): 500–504.

Bentner, Johannes, Günter Bauer, Gustav M. Obermair, Ingo Morgenstern, and Johannes Schneider. 2001. Optimization of the time-dependent traveling salesman problem with Monte Carlo methods. *Physical Review E* 64 (3): 036701–036708.

Miami International Airport. n.d. Airline ticket counters. `http://www.miami-airport.com/html/airline_tickets_counters.html` . Accessed 3 February 2006.

U.S. Department of Transportation. n.d. Airline on-time statistics and delay causes. `http://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp`. Accessed 4 February 2006.