Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds

RUNWAY SCHEDULE DETERMINATION BY SIMULATION OPTIMIZATION

Thomas Curtis Holden
Frederick Wieland

The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102, U.S.A.

ABSTRACT

Many airport runway expansion projects are restricted by space limitations imposed by development in the vicinity of the airport. This often causes planners to choose configurations for new runways that limit the use of these runways in time and/or space. Studies that model airports with new runways that are not yet operational need to develop plausible operational models for these new runways since historical data is not available. We look at a runway schedule problem encountered during the configuration and validation step of an earlier study. We develop a method using simulation optimization to approach the runway schedule problem and compare it to a manual approach developed in the earlier study. We use the Total Airspace and Airport Modeler to model the airport and airspace operations and Fast Simulated Annealing for the optimization.

1 INTRODUCTION

Economic and aviation industry analysis suggests that passenger air travel demand in the United States will grow in the medium and long term despite the recent slowdown (FAA 2002a). The plans to deal with this growth involve the addition of new runways to airports in congested areas (FAA 2002b). Some airports are severely constrained by geography and local development causing planners to choose configurations for new runways that limit the use of the new runways in time and/or space. Integrating these constrained runways into the existing aviation infrastructure is a challenging task that has been the focus of a significant amount of research.

We look at a runway schedule problem encountered during the configuration and validation step of an earlier study. The previous study models an airport with a new runway that is restricted to either departures or arrivals, but not both, at any given time. Since no historical data existed for the new runway at the time of the previous study, the analysts needed to develop a plausible schedule for the new runway. We use simulation optimization to find such a schedule. The method developed here is offered as an alternative to a manual optimization approach used in the previous study.

The airport model used in this study is taken from the previous study. This model is built for the Total Airspace and Airport Modeler (TAAM). TAAM is a deterministic, fast-time, time-stepped simulation that models airports, airspace and flights. Airports can be modeled as simple point sources or in great detail as a network of taxiways connecting runways to gates. Flights are modeled as individual physical airplanes that have state (altitude, location, velocity, mass, etc.) and properties (average climb rate, fuel capacity, maximum speed, etc.) that depend on the type of the airplane. The state of each flight evolves in simulation time and is determined by the flight's schedule, its interaction with other flights and the airport and airspace rules. Simplified laws of kinetics that depend on the properties of each airplane govern the incremental movement of flights. Airport and airspace rules can be added to make simulated flights conform to standard departure, enroute and approach paths.

The rest of this paper is organized as follows. In section 2 the previous study and the runway schedule optimization problem are discussed in more detail. In section 3 the simulation optimization approach to the runway schedule optimization problem is developed. Section 4 discusses the simulation optimization results and contrasts them with the previous study approach. In section 5 we summarize with a conclusion.

2 THE RUNWAY SCHEDULE PROBLEM SETTING

2.1 Runway Operations

The runway considered in this study is restricted to operations at one end. We refer to each end of the runway by directional orientation, ϕ_0 and ϕ_1. The three possible

states of the runway are illustrated in Figure 1. These states are the following:

1. $\sigma^{(0)}$, the ϕ_0 direction is open for departure.
2. $\sigma^{(1)}$, the ϕ_1 direction is open for arrival.
3. $\sigma^{(c)}$, the runway is closed.

When the runway is open for departure in the ϕ_0 direction it obviously cannot accept arrivals in the ϕ_1 direction. To prevent operational conflict between departures and arrivals the runway is not allowed to be open for arrival or departure simultaneously.

Limiting the runway to arrivals or departures at any one time forces controllers to decide when ϕ_0 should be open for departures and when ϕ_1 should be open for arrivals. A "good" schedule for the runway that optimizes some metric like runway utilization or delay will involve tradeoffs between satisfying arrival and departure demands. The relationship between theses tradeoffs is quite complex since the arrival and departure demand as a function of time depends on many factors.

2.2 The Manual Optimization Approach

The manual approach involves running the simulation with two baseline schedules, one with the runway in state $\sigma^{(0)}$ all-day, and the other with the runway in state $\sigma^{(1)}$ all-day. The arrival delay associated with the $\sigma^{(0)}$ all day schedule and the departure delay associated with the $\sigma^{(1)}$ all day schedule is plotted vs. time of day on one graph. This graph for the model used in this study is shown in Figure 2. The estimate of the best schedule is developed by picking the time periods when delay for the $\sigma^{(0)}$ all day schedule is high and making ϕ_1 open for arrival during those times. During the rest of the day ϕ_0 is set open for departure.

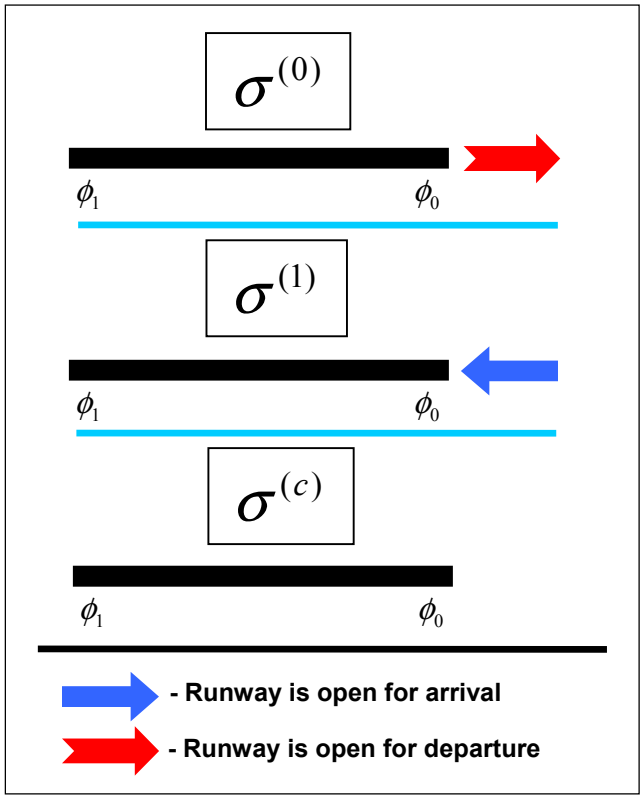

Figure 1: Runway States

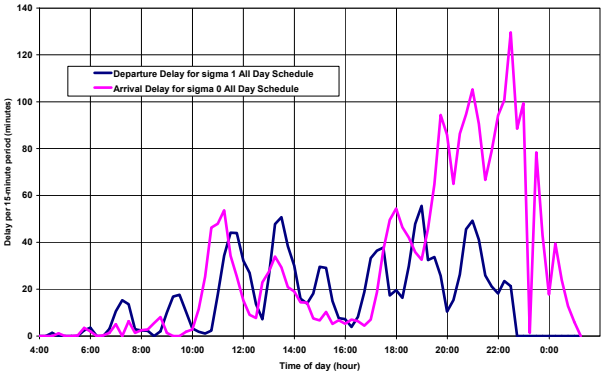

Figure 2: Arrival and Departure Delay

3 THE SIMULATION OPTIMIZATION APPROACH

The relationship between the runway schedule and delay is probably non-linear. Because of this, the manual approach to minimize delay is probably limited in terms of how much delay can be reduced. Here is a list of some additional factors that may limit the manual approach:

1. Certain flights in the model cannot use the runway, so delay contributions by these flights should not be considered in the same way as flights that can use the runway.
2. It is not clear which runway should be open when delay peaks for the $\sigma^{(0)}$ all-day schedule and the $\sigma^{(1)}$ all-day schedule occur during the same period and are of comparable magnitude.

A simulation optimization technique may be able to account for all of these factors.

3.1 Overview of Simulation Optimization

Simulation optimization is the use of search methods to find input parameter settings that improve selected output measures of a simulated system (Boesel 2001). The motivation for doing simulation optimization is to support analytical studies that use simulation to study real world systems. Applications of this technique include transportation systems, manufacturing systems, supply chains, call centers and finance (Fu 2001).

Most simulation optimization approaches include the following components: an optimization algorithm, an objective function, a set of constraints and a simulation engine. The optimization algorithm attempts to find a minimum or maximum value for the objective function. The objective function is a wrapper for the simulation that translates parameters from the optimization algorithm to a configuration object that the simulation uses. The objective function also gathers values from the simulation output to generate a single result. The constraints define valid solutions based on the objective function input parameters and/or results.

3.2 Runway Schedule Objective Function

The runway schedule objective function creates a new TAAM runway preferences configuration file (a .prf file) each time it is called. The TAAM runway preferences file defines a set of consecutive time windows and corresponding states for each runway at the airport. Each time entry in the preferences file defines the end time of one window and the starting time of the next window.

The runway schedule objective function needs to translate the values passed to it by the optimization algorithm into a valid runway preferences file. These values are represented by a vector we will call $\boldsymbol{\theta}$. Each element in $\boldsymbol{\theta}$ is represented by θ_i, where i is the index of the value. The θ_i are indexed in the following manner: $\theta_0, \theta_1, \theta_2, \ldots, \theta_{n-1}$ where n is the number of elements in $\boldsymbol{\theta}$. It is clear that two classes of things need to be represented in $\boldsymbol{\theta}$: time windows and the corresponding runway states. To address these requirements, each θ_i value is taken to be the end time for the *ith* time window and the state for the *ith* time window is taken to be a state σ_i. The start time of each time window is the end time of the previous time window; therefore, the period spanned by time window $i+1$ is $\theta_{i+1} - \theta_i$. A fixed active time period during one day is defined: $\tau^{(s)}$ is the start time for the first time window while $\tau^{(e)}$ is the end time of the last time window. Each θ_i represents a number of minutes past $\tau^{(s)}$. The time resolution of the preferences file is to the minute, so the objective function essentially operates on a discrete set of parameters.

In the previous study it was determined that the runway needs to be closed for a short period of time when operations transition from arrivals to departures or departures to arrivals. This closure is required to flush out any flights from the old state so that interference between flights is avoided. It was determined that the runway needs to be closed for 5 minutes when transitioning from departures on ϕ_0 to arrivals on ϕ_1, and 10 minutes when transitioning from arrivals on ϕ_1 to departures on ϕ_0.

Since we are interested in a solution that takes the form of a set of time windows that each represent one of two states $\{\sigma^{(0)}, \sigma^{(1)}\}$, the simplest form of any solution will be a sequence of consecutive time windows that alternate state between $\sigma^{(0)}$ and $\sigma^{(1)}$ with the appropriate flush periods included in each time window. We mimic this form by alternating the σ_i states in the following way:

$$\sigma_i = \begin{cases} \sigma^{(1)}, & \text{when } i \text{ is odd} \\ \sigma^{(0)}, & \text{when } i \text{ is even.} \end{cases}$$

Each time window includes the appropriate flush period, which we will call δ_i, at the end of the time window. δ_i is determined by the states σ_i and σ_{i+1}. The state before $\tau^{(s)}$ is σ_{-1} and is set to the state of the first time window, σ_0, while the state after $\tau^{(e)}$ is σ_n and is set to the state of the last time window, σ_{n-1}.

Figure 3 is a diagram of the information represented by three hypothetical elements of $\boldsymbol{\theta}$. The values are: $\theta_i = 430, \theta_{i+1} = 460, \theta_{i+2} = 500$. The axis at the bottom of Figure 3 represents the time of day in minutes past midnight while the shading represents the state of the runway at a particular time. The time spanned by each of the three $\boldsymbol{\theta}$ elements and the flush periods are marked on the diagram.

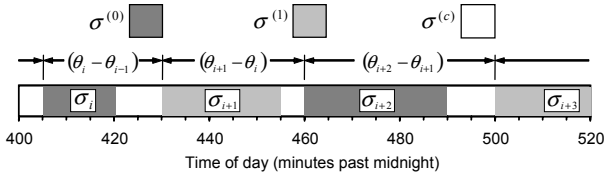

Figure 3: Example of $\boldsymbol{\theta}$ Information

It is possible that the difference between a set of consecutive θ_i values will be less than the intervening δ_i periods causing the δ_i to overlap. When overlap clusters like this form, the time windows represented by θ_i are replaced in the TAAM preferences file by a single time window with a new state σ' and flush period δ' that depend on the σ_i and δ_i values. When these overlap clusters form, the states represented by θ_i are squeezed out and the resulting preferences file will have less than n time windows. Since the absolute optimal schedule may have fewer than n time windows, allowing time windows to be

squeezed out will allow the optimal solution to exist in the solution space of $\boldsymbol{\theta}$.

The overlap clusters mentioned in the paragraph above are defined by any contiguous set $\{\theta_i \mid 0 \le m \le i \le m' < n\}$ where the following hold:

$$\theta_m - \theta_{m-1} > \delta_m \text{ when } m \neq 0,$$
$$\theta_{m'+1} - \theta_{m'} > \delta_{m'+1} \text{ when } m' \neq n-1,$$
$$\theta_{i+1} - \theta_i \le \delta_{i+1} \quad \forall \ i \text{ in } m \le i < m'.$$

The first two inequalities define conditions on the bounds m and m', which span all the θ_i in a overlap cluster, while the last inequality indicates that only the θ_i that qualify for the overlap cluster are permitted in the set. The window used in the runway preferences file in place of a overlap cluster is a window with state $\sigma' = \sigma_m$ and flush period δ' which is set to:

$$\delta' = \delta'_{m'}$$

where $\delta'_{m'}$ is defined by the recursive relation,

$$\delta'_{i+1} = (\theta_{i+1} - \theta_i)\left(\frac{\delta_{i+1} + \delta'_i}{\delta_{i+1}}\right) \forall\, i \text{ in } m \le i < m',$$

and

$$\delta'_m = \delta_m$$

when $\sigma' = \sigma_{m'+1}$ or

$$\delta' = \begin{cases} 10, \text{ when } \sigma' = \sigma^{(1)} \text{ and } \sigma_{m'+1} = \sigma^{(0)} \\ 5, \text{ when } \sigma' = \sigma^{(0)} \text{ and } \sigma_{m'+1} = \sigma^{(1)}, \end{cases}$$

when $\sigma' \neq \sigma_{m'+1}$. The total period of the window used in the preferences file is $\theta_{m'} - \theta_m$. This method of squeezing out time windows provides for smooth variation in the size of the overlapping flush periods, δ', with respect to the time difference between the squeezed time windows in the cluster.

It is possible that the optimal schedule may have more than n time windows. In this case the absolute optimal schedule will not be in the solution space of $\boldsymbol{\theta}$. The choice for the size of $\boldsymbol{\theta}$ will be limited by the number of times a runway can practically change state in the period of time defined by $\{\tau^{(s)}, \tau^{(e)}\}$.

The constraints on $\boldsymbol{\theta}$ are:

1. $\tau^{(s)} \le \theta_i \le \tau^{(e)} \ \forall\, i \text{ in } 0 \le i < n$
2. $\theta_i \le \theta_{i+1} \ \forall\, i \text{ in } 0 \le i < n$.

The first constraint is a simple projection that limits the range of the values in $\boldsymbol{\theta}$, while the second constraint requires the elements to be in ascending order which is required by the representation of the $\{\sigma^{(0)}, \sigma^{(1)}\}$ states.

The result of the runway schedule objective function is a single number, ρ, that represents average total delay. The average total delay is calculated by averaging the sequencing and departure delay values for all flights arriving and departing from the airport. The flight delay values are taken from the 'seq/dep delay' field of the TAAM report (.rep) file.

3.3 Optimization Strategy

The optimization strategy used in this study utilizes a variant of Simulated Annealing (SAN) called Fast Simulated Annealing (FSAN) (Szu and Hartley 1987). SAN is a technique for random search optimization based on an analogy to the condensed matter process of annealing. SAN was formalized for combinatorial optimization by Kirkpatrick et al. (1981) and was later extended to apply to general continuous and discrete optimization problems. Annealing is the process of heating a solid to a high temperature and then cooling it at a slow rate so that the final state is at or near the lowest energy state. The ground state, or lowest energy state, for many solids has a special form such as a crystalline structure. Physical annealing occurs naturally in magma intrusions in the crust of Earth and is also used in the laboratory and in industrial production to create solid materials with very specific properties such as specialized metals and silicon wafers.

The basic SAN algorithm has three main functional components: the acceptance function, the generation function and the cooling schedule. SAN is an iterative algorithm that successively calls the objective function with a point in the solution space returned by the generation function. The generation function picks a random point from a unimodal distribution which samples the entire solution space with non-zero probability. This distribution is centered on a reference point, $\hat{\boldsymbol{\theta}}$, which is taken as the current estimate of the optimum. The unimodal distribution biases the search to a neighborhood around $\hat{\boldsymbol{\theta}}$. At each iteration new points are accepted as $\hat{\boldsymbol{\theta}}$ with probability:

$$A(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}, t_A) = \min\{1, e^{-\left(\frac{L(\boldsymbol{\theta}) - L(\hat{\boldsymbol{\theta}})}{t_A}\right)}\},$$

where $L(\boldsymbol{\theta})$ is the value of the objective function at $\boldsymbol{\theta}$ and t_A is the acceptance temperature. $L(\boldsymbol{\theta})$ is often referred

to as the "energy" in the context of SAN. The value of t_{A_k} is controlled by the cooling schedule and is a decreasing function of the iteration k. This acceptance function allows SAN to accept new solutions with higher energy than the reference solution with non-zero probability.

The non-zero probability of accepting points with higher energy as the new reference point prevents SAN from searching only one neighborhood of the solution space. This can be beneficial when searching objective functions that have many undesirable local minima since SAN will tend to wander from the vicinity of one local minimum to another. This wandering behavior allows SAN to converge to a global minimum as $k \to \infty$ when certain conditions are met (Locatelli 2002).

FSAN uses a generation function that picks points from the Cauchy distribution while the classic form of SAN, sometimes known as Boltzmann annealing, picks points from the Normal distribution. The implementation used in this study scales the Cauchy distribution with a generation temperature, t_{G_k}, which decays as a function of the iteration k. This reduces the size of the neighborhood that is searched as the number of iterations increases, allowing the algorithm to eventually focus its search on one neighborhood (presumably a neighborhood that contains the global minimum). Szu and Hartley (1987) showed that when the Cauchy distribution is used to generate candidate points, the temperature schedules t_{A_k} and t_{G_k} can decay as fast as $1/k$ and still guarantee convergence. This is much faster than the $1/\log(k)$ decay schedule required for classic Boltzmann annealing.

We also ran the objective function against a simple blind random search optimization algorithm. The blind random search picks points from a uniform distribution that samples the entire solution space with equal probability. The blind random search does not focus its search on a particular neighborhood of the solution space. If the neighborhood search behavior of FSAN is any benefit when FSAN is applied to the runway schedule objective function, we will expect FSAN to perform better than the blind random search.

Since both FSAN and blind random search evaluate the objective function once for each iteration, the number of objective function evaluations is equivalent between FSAN and blind random search when the number of iterations is the same. This point is important because the computational time required to evaluate the objective function is orders of magnitude longer than the time taken by any other part of the optimization algorithm.

3.4 TAAM Simulation

The version of TAAM used in this study is TAAM Plus V1.2.1, Release 1 compiled on SunOS 5.8 for i86pc. The results from the runway schedule optimization used in the previous study cannot be compared directly to the results obtained using the simulation optimization technique developed in this study because this study used a different version of TAAM. Therefore, to facilitate a comparison with the simulation optimization method developed in this study, we use the method developed in the previous study to create a schedule.

3.5 Model Setup

The TAAM model was set up based on the model defined in the previous study. The model defines one full day of traffic which includes 784 departures and 783 arrivals. All TAAM runs were done with the airport ground model turned off. This is consistent with the previous study and eliminates any delay associated with taxiing and gate usage. The runtime of the model was about 200 seconds on a 700 MHz Xeon machine.

4 RESULTS

We obtain a solution using FSAN that result in lower average total delay than the blind random search and the manual method (see Figure 4). We ran 15 replications of FSAN, blind random search and the manual method using the runway schedule objective function. The parameters for the FSAN runs are listed in Table 1 while the blind random search parameters are listed in Table 2. The start time for the active period, $\tau^{(s)}$, is set to 360 because there is very little traffic before 360.

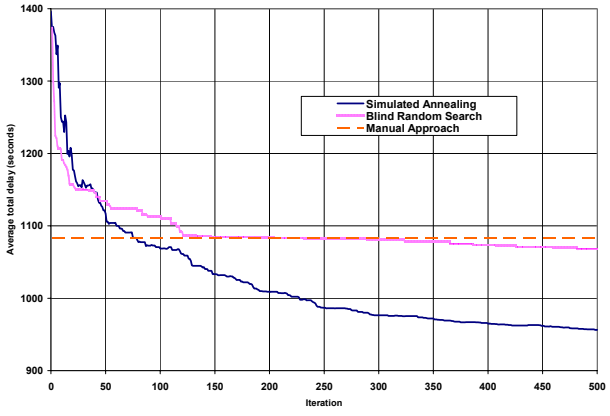

Figure 4: FSAN and Blind Random Search Results

Table 1: Parameters Used for FSAN Runs

FSAN			Objective function		
Iterations	t_{G_0}	t_{A_0}	Size of $\boldsymbol{\theta}$	$\tau^{(s)}$	$\tau^{(e)}$
500	1000	1000	13	360	1080

Table 2: Parameters Used for Blind Random Search Runs

Blind random search	Objective function		
Iterations	Size of $\boldsymbol{\theta}$	$\tau^{(s)}$	$\tau^{(e)}$
500	13	360	1080

The manual data was generated by 15 runs of TAAM with 15 different variations of the TAAM preferences file. The preferences files were generated using the procedure described in section 2.2. The number of time windows used in the preferences file varied from 9 to 13.

The sample mean of the average total delay for the reference solution of the FSAN runs and the sample mean of the average total delay for the best solution of the blind random search runs are plotted as a function of iteration in Figure 4. The sample mean value of the manual runs also appears in Figure 4. The 500th iteration sample mean value is significantly lower for the FSAN runs than the blind random search runs. The sample mean delay for the FSAN reference solution matches the manual solution after about 75 iterations, while the blind random search matches it after about 150 iterations. The FSAN curve appear to be decreasing at iteration 500, which suggests that FSAN was still not close to convergence at iteration 500. This is supported by the fact that FSAN runs that were allowed to run out to 800 iterations achieved solutions with average total delay as low as 888.621 seconds.

Table 3 summarizes statistics on the runs. ρ_m is the value of the mean average total delay of all the manual runs and is the mean average total delay of the reference solution at iteration 500 for the FSAN and blind random search runs. The confidence interval was generated using 95% confidence interval t-statistics. Notice that the FSAN 95% confidence interval does not overlap with either the blind random search or the manual 95% confidence interval. This confidence interval data provides further statistical evidence that the improvement of the FSAN approach over the blind random search and manual approaches is significant. The confidence interval for the manual runs is much larger than the FSAN and blind random search runs which suggests that the optimization algorithms generate more consistent results than the manual method. Although this is true for these particular runs, it is important to note that the manual confidence interval and sample mean may vary depending on who picks the schedule since the results will depend significantly on the judgment of the person picking the candidate schedules. The mean total runtime is the mean total time required to do one run of an optimization method. The FSAN approach is somewhat faster than the blind search. This is because TAAM runs faster with schedules that produce lower average total delay. The manual solution required only one run, so it obviously ran much faster than the FSAN and blind random search runs. The mean total runtime data for the manual runs do not take into account the time required to do the two runs needed to make Figure 2.

Table 3: Summary of Results

Optimization Approach	ρ_m (s)	Confidence Interval	Mean Total Runtime (s)
FSAN	956.5275	$\rho_m \pm 12.77$	94775.9
Blind search	1067.802	$\rho_m \pm 11.93$	101102.6
Manual approach	1082.946	$\rho_m \pm 43.50$	198.93

The 500th iteration reference solutions for the 15 FSAN runs appear in Figure 5 and the initial (or 0th iteration) reference solution for all the FSAN and blind random search runs appear in Figure 6. The states for each time window follow the shading scheme defined in Figure 3 and the flush periods are not included. As we would expect, 500th iteration solutions have a bias towards the $\sigma^{(1)}$ state after around 16:00 when the arrival delay for the $\sigma^{(0)}$ all day schedule is high (see Figure 2). The manual schedules developed based on the delay graph shown in Figure 2 also have this bias, but the FSAN mean average total delay is lower. The differences between the schedules developed using the manual method and the FSAN runs may be due to the limitations discussed in Section 3.

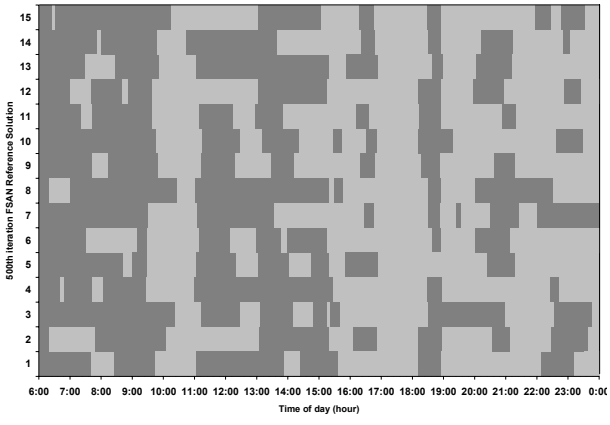

Figure 5: 500th Iteration FSAN Solutions

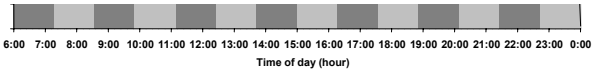

Figure 6: Initial FSAN Solution

5 CONCLUSIONS

In this study we developed a method to do runway schedule optimization using simulation optimization. This method used the TAAM simulation in conjunction with Fast Simulated Annealing (Szu and Hartley 1987). The method applies when there are two possible states for the runway. The method results compare favorably to a man-

ual method developed by a previous study and results from a blind random search algorithm.

The time and resources required to do simulation optimization using TAAM are not trivial. The sample mean runtime of 15 500-iteration runs of FSAN using the TAAM-based objective function is about 26 hours on a 700 MHz Xeon machine. If simulation optimization is to be used to calibrate and validate TAAM models, the time and effort must be carefully weighed against an alternative approach (such as a manual approach).

It would be interesting to explore the possibility of using a runway schedule simulation optimization technique in a real-world setting. This could be used to help ground controllers decide what runway schedule to use. This application would require a fast, high fidelity simulation and a simulation optimization procedure that can be updated as the real time state evolves. It would also require that actual airline schedules, including provisions for general aviation traffic that have filed flight plans, be used by the model during the optimization phase. As external changes to the schedule take place (cancellations, substitutions, additions), then the optimization procedure may need to be repeated with the changed traffic.

ACKNOWLEDGMENTS

We would like to thank Preston Aviation for providing the TAAM software for this project. We would also like to thank Mike Yablonski for providing the model used in the study. Mike also provided crucial tips and insight on the basic and not so basic aspects of using TAAM. John Kuchenbrod provided us with additional vital tips and tricks about using TAAM. The contents of this material reflect the views of the authors. Neither the Federal Aviation Administration nor the Department of Transportation makes any warranty or guarantee, or promised, expressed or implied, concerning the content or accuracy of the views expressed herein.

REFERENCES

Boesel, J., R.O. Bowden Jr., F. Glover, J.P. Kelly, and E. Westwig, 2001. Future of Simulation Optimization. In *Proceeding of the 2001 Winter Simulation Conference,* J.A. Joines, R.R. Barton, K. Kang, and P.A. Fishwick, eds., pp. 1466-1469. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Federal Aviation Administration (FAA), 2002a. FAA Aerospace Forecasts: Fiscal Year 2002-2013, Washington, DC: US Department of Transportation.

Federal Aviation Administration (FAA), 2002b. National Airspace System Operational Evolution Plan, Washington, DC: US Department of Transportation.

Fu, M.C., 2001. Simulation Optimization. In *Proceeding of the 2001 Winter Simulation Conference,* J.A. Joines, R.R. Barton, K. Kang, and P.A. Fishwick, eds. pp. 53-61. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Kirkpatrick, S., C.D. Gellatt, and M.P. Vwecchi, 1981. Optimization by simulated annealing. *Science* 220: 671-680.

Locatelli, M., 2002. Simulated annealing algorithms for continuous global optimization, *Handbook of Global Optimization* volume 2, P.M. Pardalos, H.E. Romeijn, eds., Dordrecht, The Netherlands: Kluwer Academic Publishers.

Szu, H. and R. Hartley, 1987. Fast Simulated Annealing, *Physics Letters A* 122: No. 3, 157-162.

AUTHOR BIOGRAPHYS

THOMAS CURTIS HOLDEN is a Senior Simulation and Modeling Engineer at The MITRE Corporation's Center for Advanced Aviation Systems Design (CAASD). His e-mail address is <tholden@mitre.org>.

FREDERICK WIELAND holds a PhD in information technology/applied probability theory from George Mason University. He is the developer of numerous simulations for the U.S. Department of Defense as well as the Federal Aviation Administration, including CTLS, DPAT, Matrix and others, and has done extensive research in the parallelization of large-scale simulations such as TAAM. He has been working in the simulation field for 20 years. His e-mail address is <fwieland@mitre.org>.