AIAA Conference on Guidance, Navigation and Control
San Francisco, California, August 15-18, 2005
AIAA Paper 2005-6044

Polynomial Time Algorithms for Scheduling of Arrival Aircraft

Kaushik Roy* Alexandre M. Bayen† Claire J. Tomlin‡

A Mixed Integer Linear Program (MILP) formulation of an arrival traffic scheduling problem is used as the starting point in order to evaluate the respective capabilities of two real-time scheduling algorithms. Three main issues are addressed: computational time, suboptimality and statistical performance. The first algorithm is an approximation algorithm, which is polynomial time and has a guaranteed bound on the suboptimality ratio. The second algorithm is a heuristic algorithm, which is polynomial time. Both algorithms are compared to the exact solution (computed with CPLEX) under realistic air-traffic scenarios generated from American Airlines timetable data for arrival traffic into Saint Louis, Missouri. Results show that the heuristic algorithm has the fastest run-time and is nearly optimal in most scenarios. However, the approximation algorithm generates schedules well within theoretical bounds, providing a guarantee on performance that the heuristic cannot provide. The approximation algorithm is also more robust to adverse weather scenarios than the heuristic algorithm.

I. Introduction

The return of air traffic density to very high values in the United States has again motivated the need for efficiency in the Air Traffic Control (ATC) system. Optimization of arrival traffic is important in reducing delays in an airspace stretched to current capacity limits. For example, weather events such as thunderstorms can disrupt arrival airflow at a handful of airports, leading to large delays throughout the national airspace, reducing efficiency dramatically. A great deal of previous work has concentrated on routing and scheduling of arrival flows under weather restrictions (see for example, Krozel and Penny [7] and Prete and Mitchell [9]).

Currently, a semi-automated ATC system exists to help controllers manage air traffic flow near airports. Information tools such as CTAS,[8] and position estimation technologies such as GPS and WAAS have made it possible for the air traffic controller to obtain better information about aircraft position and intent. However, the management and coordination of multiple aircraft in arrival airspace is still generally performed manually. Specifically, conflict avoidance and some scheduling maneuvers are generated through experience-based rules (called playbooks) by air traffic controllers. Heuristics based on priority and some ideas of fairness are used to sequence aircraft arriving in the direct vicinity of an airport.[3]

The problem currently solved by air traffic controllers is based on the idea that each aircraft has an earliest possible arrival time and a delayed arrival time which can be achieved by deviations from the nominal speed or path. The air traffic controller can have aircraft land anytime during this interval by prescribing the proper control maneuvers. An assignment of landing times to aircraft requires that each landing time is within this feasible interval for the given aircraft and that the landing times are spaced apart. That is, only one aircraft can land in any Δ time units, where Δ is sometimes called the metering time. When such an assignment is not possible, aircraft may be put on holding patterns, according to controller discretion, to generate a landing schedule which is feasible in terms of the airport spacing requirements and the individual aircraft landing intervals. The assignment of landing times for aircraft depends both on a continuous landing interval and discrete decisions on whether to hold a specific aircraft.

*Ph.D. Candidate, Electrical Engineering, Stanford University.
†Assistant Professor, Civil and Environmental Engineering, University of California, Berkeley.
‡Associate Professor, Aeronautics and Astronautics, Courtesy Associate Professor, Electrical Engineering; Director, Hybrid Systems Laboratory, Stanford University.

Algorithms which generate real-time schedules for arrival aircraft can augment the intuition and experience of air traffic controllers. To obtain such algorithms, the problem of routing and scheduling of arrival air traffic flow may be abstracted into a problem of optimization of a *Mixed Integer Linear Program* (MILP). For example, previous work[2] proposes a polynomial time reduction of a hybrid systems control synthesis problem modeling maneuver assignment in arrival airspace into a MILP. Various criteria are available for determining an optimal schedule: this paper focuses on minimizing the sum of arrival times. Because the sequencing, routing, and holding of arrival aircraft is inherently combinatorial, an exact solution of the MILP is necessarily of exponential complexity.

The reason for using suboptimal algorithms is that they run in polynomial time. The reason for using suboptimal algorithms is that they run in guaranteed time and can be used in a real-time setting (i.e. deployed to aid air traffic controllers). This paper first summarizes an approximation algorithm developed earlier,[5] and then presents a newly developed heuristic algorithm. The approximation algorithm combines dynamic programming and linear program optimization techniques to generate feasible schedules for arrival air flows with guarantees on optimality and runtime. The heuristic is a greedy scheme, through an iterative scheme it attempts to land aircraft as soon as possible. The algorithms presented here generalize very well for other objective functions such as weighted sum of arrival times or arbitrary functions linear in the arrival times. Another possibility is to minimize the *makespan*, which is the maximum landing time of the aircraft to be scheduled. These algorithms are then simulated under realistic air traffic scenarios and their performances are evaluated. Air traffic scenarios are generated by using timetable data for American Airlines flights landing at Saint Louis, Missouri, as a model for expected arrival air flows.

The organization of the paper is as follows. In the next section, the air traffic problem and the related MILP are formally defined. Sections III and IV discuss the previously proposed approximation algorithm and the new heuristic algorithm, respectively. In Section V, a realistic air traffic scenario is discussed and results of Monte Carlo simulation are provided. The two algorithms are compared to the optimal solution in both optimality and run time. Finally, conclusions and future work directions are presented in Section VI.

II. Formulation of arrival scheduling problem and Mixed-Integer Linear Program (MILP)

As discussed in previous work,[3] each of N aircraft landing at an airport are presumed to have an earliest possible landing time and a delayed landing time. For the i^{th} aircraft, let these times be denoted a_i and b_i, respectively. Holding patterns take T time units for all aircraft. The feasible set of arrival times for aircraft i is thus $\cup_{k=0}^{\infty}[a_i + kT, b_i + kT]$, which will be denoted by $[a_i, b_i] + T\mathbb{N}$, where $\mathbb{N} = \{0, 1, 2, ...\}$ are the natural numbers. The holding pattern time T and the required spacing between landings Δ are parameters set by the environment of a given scenario, while the sets a_i and b_i are decided by the configuration of the aircraft arriving at an airport. The problem of minimizing the sum of arrival times can then be posed as an optimization problem as follows:

$$\begin{array}{lll} \textbf{min:} & \sum_{i\in\{1,...,N\}} \tau_i & \\ \textbf{s.t.:} & \tau_i \in [a_i, b_i] + T\mathbb{N} & \forall i \in \{1, ..., N\} \\ & |\tau_i - \tau_j| \geq \Delta & \forall i, j \in \{1, ..., N\}, i \neq j. \end{array} \tag{1}$$

Note that τ_i refers to the landing time for aircraft i in the optimal schedule.

This formulation can be encoded as a MILP by choosing variables in a different way. The formulation that results is

$$\begin{array}{lll} \textbf{min.:} & \sum_{i\in\{1,\cdots,N\}} \tau_i & \\ \textbf{s.t.:} & t_i \geq a_i + kT & \forall i \in \{1, \cdots, N\} \\ & t_i \leq b_i + kT & \forall i \in \{1, \cdots, N\} \\ & k \in \mathbb{N} & \\ & t_i - t_j \geq \Delta - Cc_{ij} & \forall (i, j) \in \{1, \cdots, N\}^2, \text{ s.t. } i > j \\ & t_i - t_j \leq -\Delta + C(1 - c_{ij}) & \forall (i, j) \in \{1, \cdots, N\}^2, \text{ s.t. } i > j \\ & c_{ij} \in \{0, 1\} & \forall (i, j) \in \{1, \cdots, N\}^2, \text{ s.t. } i > j \end{array} \tag{2}$$

where C is a "large" constant, whose size depends on the actual numerical range of k (since in practice, it will be finite for the implementation). k is the number of holding patterns and c_{ij} is a decision variable

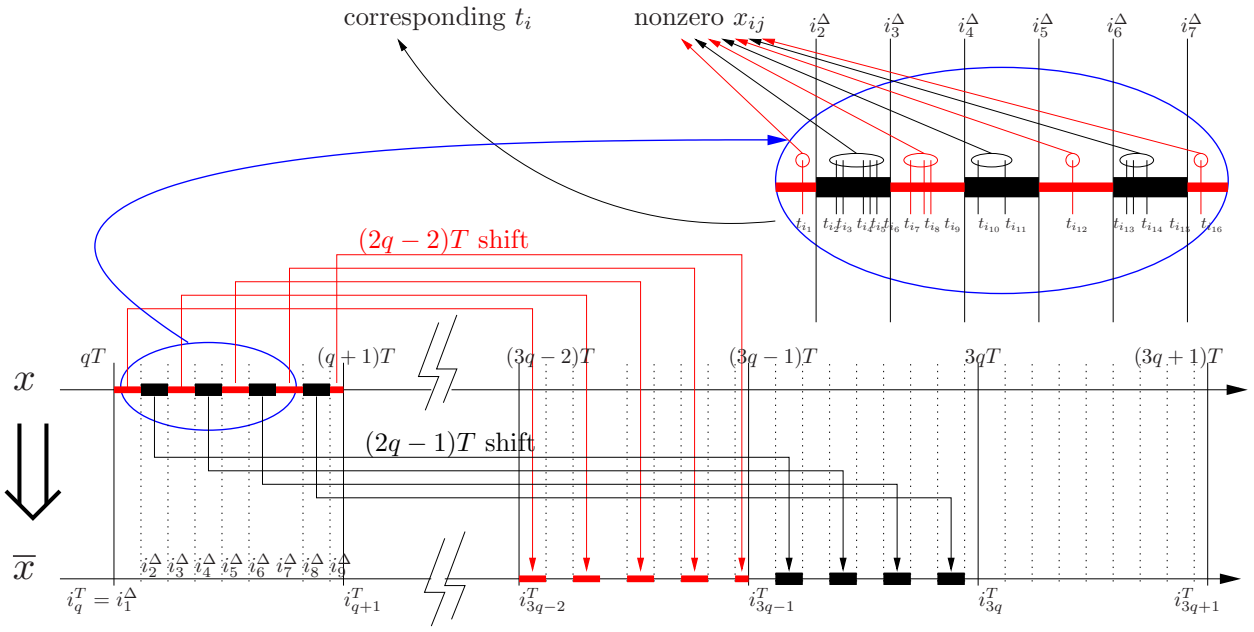


Figure 1. Illustration of STEP 4. This procedure transforms the feasible fractional solution $x = x_{ij}$ of (3) into another feasible fractional solution $\overline{x} = \overline{x}_{ij}$ with idle time. Half of the fractional x_{ij} in the interval $[qT, (q+1)T)$ is shifted to $[(3q-2)T, (3q-1)T)$ (shift by $(2q-2)T$), and the other half to $[(3q-1)T, 3qT)$ (shift by $(2q-1)T$). The interval $[3qT, (3q+1)T)$ is empty for now; this is a waste of space, but this enables us to satisfy the constraints of (3) for the part of the fractional solution between i_Q^Δ (i_9 on the figure) and i_{q+1}^T.

which determines the order of arrival of the aircraft.

The optimal schedule (that is, the τ_i) can be obtained by CPLEX, a mathematical optimization software package developed by ILOG[10] and coded in the AMPL modeling language.[6] CPLEX almost always finds the optimal scheduling of aircraft; thus, the optimal solution will also be denoted the CPLEX solution.

III. Approximation algorithm

The algorithm used by CPLEX to obtain the exact solution is highly efficient but still has exponential complexity. Because a real-time application requires guarantees on algorithmic runtime, we present two suboptimal algorithms which run in polynomial time. In Section V, we evaluate the tradeoff between optimality and run-time.

The approximation algorithm has been proposed previously[5] and is summarized in this paper. Dynamic programming is used to optimally schedule aircraft during the first T time units. The remaining aircraft are then input to the MILP in Equation (1). Relaxing the integer constraints yields a solution where fractions of aircraft can land at different times. However, the spacing constraint remains; that is, only one whole aircraft can land in any interval of length Δ. Next, the relaxed solution is expanded in time to ensure that later steps do not violate the Δ spacing requirement. This step of the algorithm is where optimality may be lost. After spacing the fractional aircraft schedules, whole aircraft are matched to their possible fractional landing times. This step is a matching problem which can be solved optimally in polynomial time. The algorithm is guaranteed to run in $O(N^9)$ time and the resultant solution is guaranteed to be at most 5 times optimal (referred to as a 5-approximation algorithm). More precisely:

STEP 1: Data preprocessing. The first step of the algorithm consists in preprocessing the data. The constraint set in equation (1) can in fact be reduced to a fully discrete set of polynomial size, as follows:

(i) Sort the aircraft by earliest possible time of arrival: without loss of generality, we thus assume $a_1 \leq a_2 \leq \cdots \leq a_{N-1} \leq a_N$.

(ii) Divide the N aircraft into $K+1$ subsets: $S_0 = \{N_0, \cdots, N_1 - 1\}$, $S_1 = \{N_1, \cdots, N_2 - 1\}$, $S_2 = \{N_2, \cdots, N_3 - 1\}$, $\cdots$ $S_K = \{N_K, \cdots, N\}$, where $N_1 = 1$, and N_k is given by $N_k = \min\{p | a_{N_{k-1}} + (p -$

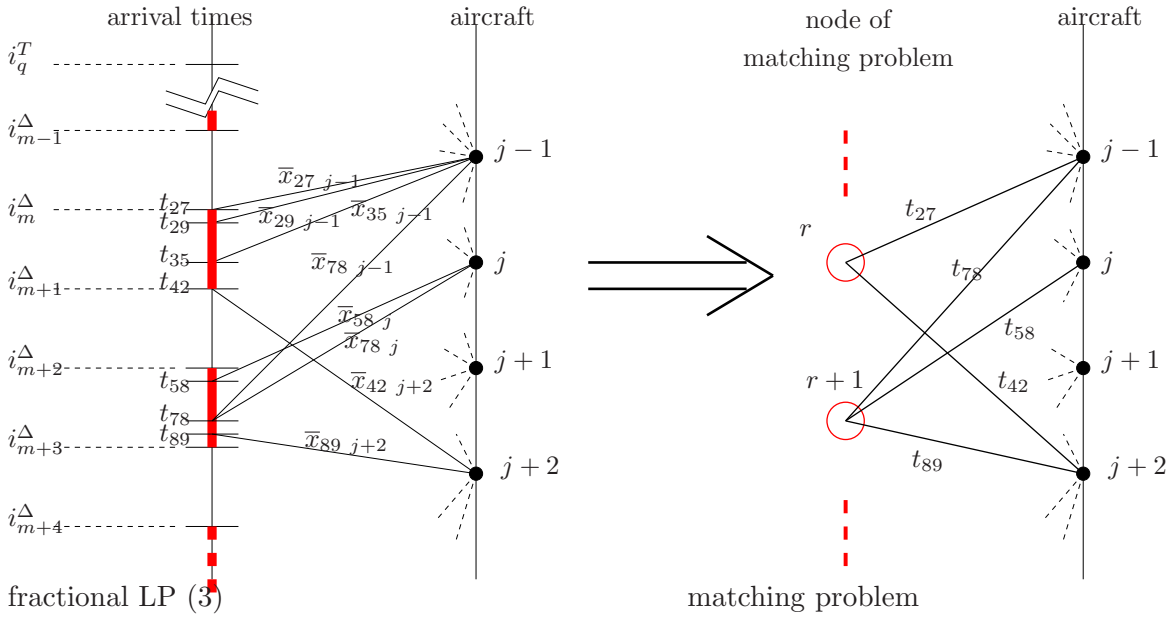


Figure 2. Illustration of STEP 5. This procedure transforms the feasible fractional solution $\overline{x}_{ij}$ into a weighted matching problem. In the fractional LP solution, only the nonzero $\overline{x}_{ij}$ are represented (solid if they are in the range of the figure, dashed if they connect aircraft with arrival times outside the figure). On the right plot, the weights are the arrival times from the fractional LP solution: for example, $\Theta_{rj-1} = t_{27}$.

$N_{k-1})(\Delta+T) < a_p\}$.

(iii) Let $\sigma_k = \left\{\bigcup_{l\in S_k}\{a_l + \Delta\mathbb{N} + T\mathbb{N}\}\right\} \cap [a_{N_k}, a_{N_k} + (N_k - N_{k-1})(\Delta+T)]$ for each k. Index the elements t_i of $\Sigma = \bigcup_{k=1}^{K} \sigma_k$ by $i \in \{1, \cdots, |\Sigma|\}$ in increasing order.

STEP 2: Extension of dynamic programming for the earliest aircraft. We apply an extension of dynamic programming previously published,[5] which is a modification of Baptiste's algorithm[1] to the aircraft capable of arriving before T.

If $a_1 > T$, skip this step. If $a_1 < T$, call F the set of i such that $[a_i, b_i] \cap \{t | t \leq T\}$ is not empty. Call $[a_i, b_i'] = [a_i, b_i] \cap \{t | t \leq T\}$. Schedule the maximum number of aircraft of F according to the extended dynamic programming algorithm from previous work.[5] If this number is equal to N, stop.

STEP 3: LP relaxation of a constrained matching problem. Solve the relaxed LP (3) for the remaining aircraft. If $a_1 \geq T$, solve the relaxed LP (3) directly.

$$\begin{array}{lll} \textbf{Minimize:} & \sum_j \sum_{i\in G(j)} t_i x_{ij} & \\ \textbf{Subject to:} & \sum_{i\in G(j)} x_{ij} = 1 & \forall j \in \{1, \cdots, N\} \\ & x_{ij} \geq 0 & \forall j \in \{1, \cdots, N\},\ \forall i \in G(j) \\ & \sum_{i'\in I(i)} \sum_j x_{i'j} \leq 1 & \forall i \in \{1, \cdots, |\Sigma|\} \end{array} \tag{3}$$

where the t_i denote the elements of Σ, and where $\forall i \in \{1, \cdots, |\Sigma|\}$, $I(i) = \{i' \leq |\Sigma| \mid t_{i'} \geq t_i \wedge t_{i'} - t_i < \Delta\}$, and $\forall j \in \{1, \cdots, N\}$, $G(j) = \Sigma \cap \{[a_j, b_j] + T\mathbb{N}\}$.

STEP 4: Transformation of the LP using a "LP rounding like technique". The mathematical description of this technique is too technical to be part of this paper; details available in previous work.[5] In summary, this step takes every t_i for which there exists a nonzero fractional matching x_{ij} in the solution of (3), and shifts it by a prescribed amount (either $(2q-1)T$ or $(2q-2)T$ where q is integer). The result is a new fractional matching, satisfying the constraints of (3), but suboptimal. This transformation is illustrated in Figure 1.

STEP 5: Construction and solution of an unconstrained weighted matching problem. The fractional matching obtained in Step 4 is used to construct a weighted unconstrained matching problem. For this, arrival times

within Δ are assembled into a single node, such that consecutive nodes are separated by more than Δ. The result is a bipartite graph (see Figure 2). The corresponding matching problem can be solved in polynomial time with the following objective function:

$$\sum_{i,j} t_i x_{ij}.$$

In summary:

5 approximation algorithm

1. Preprocess the data to make the set of feasible arrival times fully discrete.
2. If the earliest arrival time of all aircraft is smaller than T, skip this step.
   Use extended dynamic programming to schedule as many aircraft as possible among those which can arrive before T. If all aircraft have been scheduled, stop.
3. Form a constrained matching problem, solve the LP corresponding to its relaxation.
4. Transform the relaxed solution using a "LP rounding like" method.
5. Construct and solve a new weighted unconstrained matching problem.

IV. Heuristic Algorithm

This paper proposes another polynomial time algorithm based on a heuristic for comparison with the CPLEX solution and the approximation algorithm. This heuristic is a "nearsighted", yet practical solution. Essentially, the algorithm is a priority queue where the next aircraft to land is the one whose latest arrival time b_i is the minimum out of all aircraft to land. At any point in time, this most urgent aircraft is landed at the earliest possible next time. The algorithm steps through time, scheduling aircraft one at a time. In this manner, the heuristic algorithm captures much of the basic behavior of an air traffic controller. Unlike a controller, the heuristic has no notion of fairness or extenuating circumstances for landing any specific flight. Such changes could be added to the algorithm by adjusting the priorities of the aircraft. A detailed description of the heuristic algorithm follows.

The current time *currtime* is a variable used to traverse time as the schedule is generated. The *most urgent* aircraft in a set is the one with the smallest b_i (i.e. - the aircraft whose landing interval ends first). The aircraft *next in line*, given a current time, is determined as follows. Find the most urgent aircraft amongst the set of aircraft which can land at the current time (i.e. - those with $a_i \leq$ current time $\leq b_i$). If no such aircraft exist, find the aircraft with minimum $a_i \geq$ current time. Ties are broken by urgency. The landing time of the next-in-line aircraft is either the current time, if aircraft exist which can land at the current time, or a_i for the aircraft with minimum a_i if none can land at the current time.

The algorithm first initializes the current time to $-\infty$. An iteration consists of first scheduling the aircraft that is next in line. The current time is then set to the landing time of the next-in-line aircraft plus Δ, to ensure spacing requirements are met. Any aircraft which are unable to land by this new current time are put on holding pattern until they can land at or after the current time. That is, $a_i := a_i + T$ and $b_i := b_i + T$ until $b_i \geq$ current time. Each iteration of the algorithm schedules one aircraft; N iterations creates a feasible schedule. The heuristic algorithm is most useful for its fast running time, $O(N^2)$. Its performance in comparison to other algorithms is evaluated in the next section. These steps are summarized in the table below.

An example of scheduling three aircraft is presented to show the procedure of the heuristic algorithm. The arrival intervals for the aircraft are $[a_1, b_1] = [1, 7]$, $[a_2, b_2] = [2, 3]$, and $[a_3, b_3] = [2, 4]$. The minimum required spacing $\Delta = 2$ and the time for a holding pattern $T = 10$. The optimal solution is to land aircraft 2 at time 2, aircraft 3 at time 4, and aircraft 4 at time 6, as shown in Figure 3(a). The heuristic lands the first aircraft possible: aircraft 1 at time 1, updating the current time to $3 = 1 + \Delta$. See Figure 3(b) for a representation of the scheduling of the first aircraft. At time 3, both remaining aircraft can land, but aircraft 2 is landed because it is more urgent (i.e. has smaller b_i). The current time is set to 5, as shown in Figure 3(c). Because aircraft 3 cannot land at time 5, it is put on a holding pattern and landed at 12,

Heuristic algorithm solving (1)

1) Initialize $currtime = -\infty$.
2) Find next-in-line aircraft. If any aircraft can land at $currtime$, find the most urgent aircraft from this set, where urgency is defined as the aircraft with minimum latest arrival time b_i. Schedule this aircraft at $\tau_i = currtime$. If none can land at $currtime$, find the aircraft with minimum earliest arrival time a_i, and schedule this aircraft to land at time $\tau_i = a_i$.
3) Update $currtime$ to $\tau_i + \Delta$. If any aircraft are no longer able to land in the current interval (if $b_i < currtime$), then shift landing interval by T ($a_i := a_i + T$ and $b_i := b_i + T$).
4) Repeat steps (2) and (3) until all aircraft are scheduled.

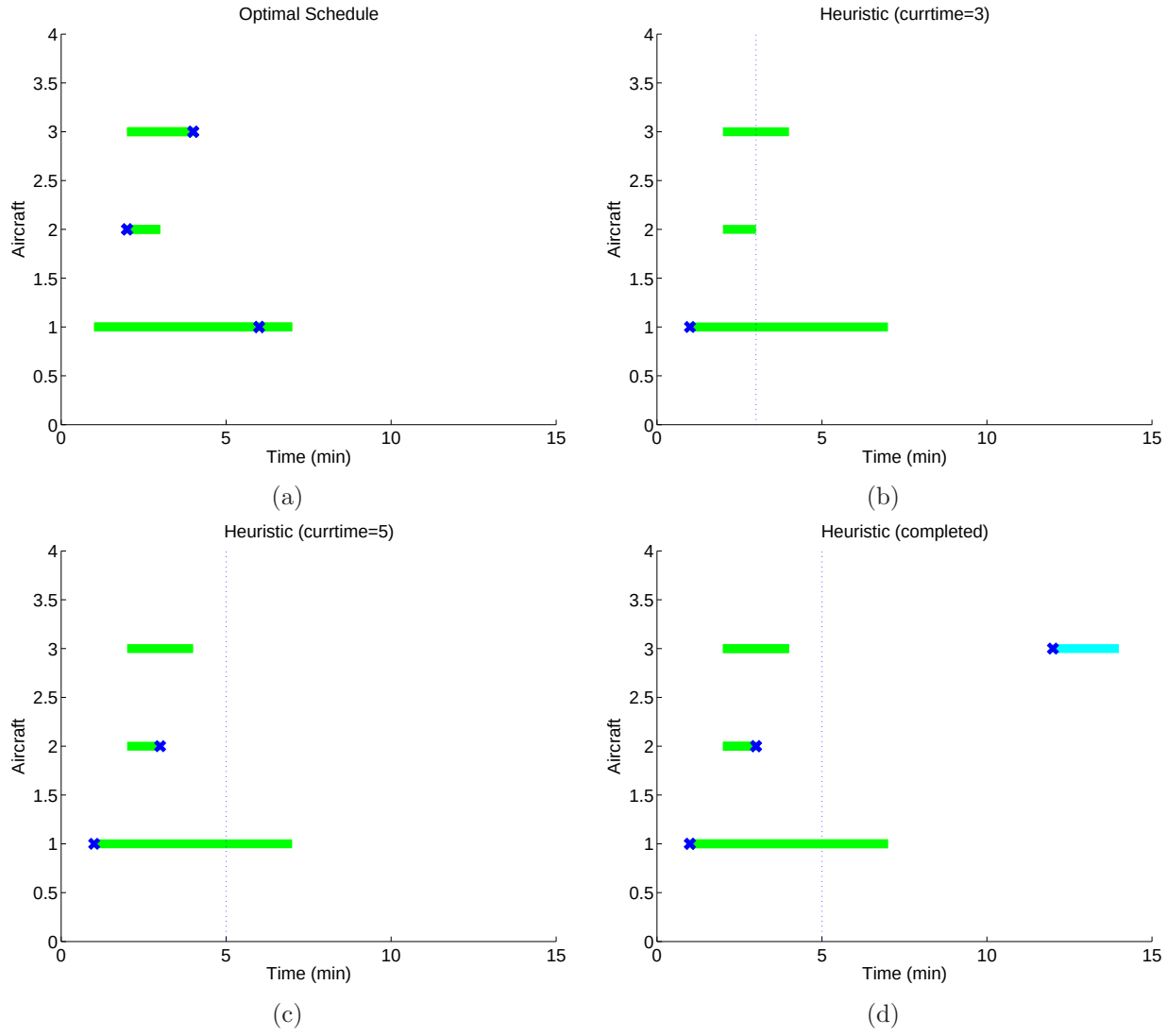


Figure 3. (a) Optimal scheduling of three aircraft with landing intervals marked as horizontal line segments and landing times marked by x's. (b) Heuristic algorithm after landing first aircraft. (c) Heuristic algorithm after landing second aircraft. (c) Heuristic algorithm at completion, having placed third aircraft on holding pattern.

the new minimum time of arrival. The final schedule is shown in Figure 3(d). Comparing Figure 3(a) and (d), we note that the heuristic schedule is suboptimal because it schedules aircraft greedily. In Section V, simulations are used to show that the heuristic generally performs more optimally.

V. Simulations and Results

The solution of the approximation algorithm is guaranteed to be within 5 times the optimal objective, though in practice this ratio can be much smaller. The solution of the heuristic algorithm, on the other hand, is not guaranteed to be within any ratio of the optimal objective; however, by construction, the heuristic solution is in general very close to optimal. Monte Carlo simulations are used to evaluate the practical factor of suboptimality of these polynomial time algorithms. Monte Carlo techniques are used to generate parameter sets which represent typical air traffic scenarios; these parameter sets are then simulated with both polynomial-time algorithms and CPLEX to obtain results. This section first presents the methodology used in generating parameter sets and in simulation. Next, three topics are discussed and relevant results are presented: (1) performance of approximation and heuristic algorithms as compared to CPLEX solution in terms of optimal objective, (2) differences in performance between nominal and weather-affected scenarios, and (3) run-time and scalability issues.

A. Simulation methodology

For all simulations, the time for a holding pattern is set at $T = 10$ (minutes). The required spacing Δ is either 2 or 4 minutes, representing nominal and weather-affected operations, respectively. The number of aircraft N is varied amongst powers of 2 between 2 and 16 to highlight the need for polynomial time algorithms whose run times scale better with more aircraft.

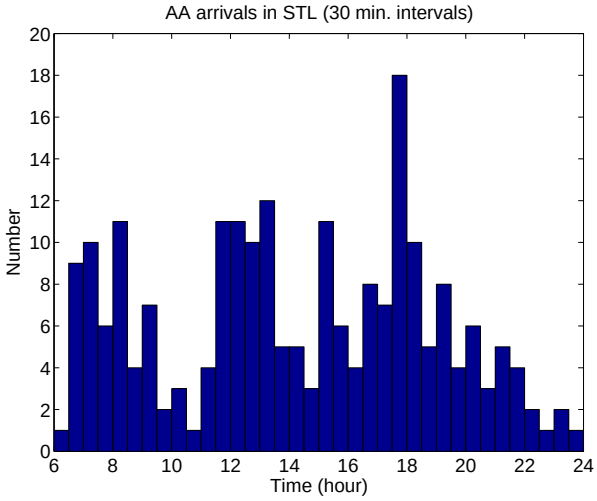


Figure 4. Time-of-day histogram of arrival aircraft for American Airlines into STL according to timetable data.[11]

The landing interval times a_i and b_i are chosen from probability distribution functions which represent a plausible air traffic scenario. Timetable data from American Airlines (AA) for flights arriving into Lambert-Saint Louis International Airport (STL) were analyzed to determine congested arrival times during the day.[11] These flights were chosen because AA has a hub at STL, and a large percentage of commercial flights into STL are therefore AA flights. In Figure 4, a histogram of the timetable arrival times of AA flights into STL is shown. Scheduling of arrival aircraft is hardest during periods of congestion; automated scheduling algorithms are therefore most useful during these periods. We define congestion as having at least 4 aircraft able to land in the next 5 minutes and at least 10 aircraft able to land within a half hour. Overlaying the congested periods throughout the day and normalizing yields the frequency plot in Figure 5. This frequency plot is the probability distribution function from which the earliest arrival times of aircraft a_i are chosen.

The latest arrival times, b_i, are calculated according to the formula, $b_i = a_i + l_i$, where l_i is an interval

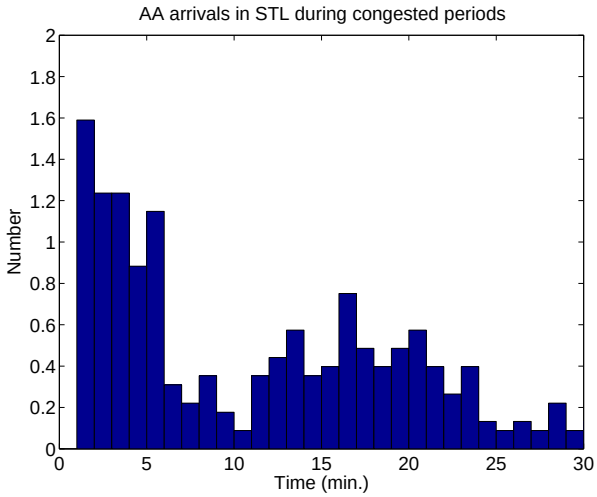


Figure 5. Frequency distribution for arrivals during congested periods. Horizontal axis plots time after start of congested period, while vertical axis represents average number of aircraft at that time. Initial peak of aircraft occurs within 5 minutes of start of congestion period; secondary peak follows at 15-20 minutes into congestion period.

length parameter. The l_i are drawn from the probability distribution function shown in Figure 6; this distribution was chosen by comparing *Standard Terminal Approach Routes* (STARs) for STL and other airports. Specifically, for a STAR on the order of 100 nautical miles, and an aircraft traveling at 300 knots, changes of plus or minus 10 percent in speed yield an interval of 4 minutes between earliest and latest arrival times. Accounting for other spacing maneuvers, we let l_i be uniform on $\{2, 3, 4, 5, 6\}$. The timetable and STARs data is static but yields a plausible distribution for the dynamic configuration of arrival aircraft, which would then be translated into a_i and b_i parameters. Thus, the a_i and b_i parameters chosen by Monte Carlo methods represent a plausible air traffic scenario. For each (N, Δ, T) triple, 500 sets of a_i and b_i are chosen from the given probability distributions; results shown are averaged over these 500 runs.

B. Comparison of optimality of algorithms

The optimal solution of (1) can almost always be found by CPLEX; thus optimal solution and CPLEX solution are used interchangeably. Arrival schedules generated by the approximation and heuristic algorithm are at best equal to and often suboptimal to the CPLEX solution. In Figure 7(a), the objective values for 500 runs with 2 aircraft are shown. The runs are sorted by ascending CPLEX objective value (hence the monotonically increasing lower envelope). Any given run (a specific x-coordinate) consists of three points plotted vertically: the CPLEX objective value, the heuristic objective value, and the approximation objective value. Figure 7(b)-(d) show results for 4, 8, and 16 aircraft. The plots show that the heuristic algorithm finds solutions which are optimal or nearly optimal. The approximation algorithm finds some schedules optimally, but it also exhibits larger suboptimality on a substantial portion of runs.

We observe that for scenarios with more aircraft, both absolute objective values and subobtimality relative to CPLEX increase for the polynomial-time algorithms. More aircraft trying to land during the same interval necessarily increases the objective, sum of arrival times. Because more aircraft have to be landed, the approximation algorithm is more likely to land fractions of aircraft. Step 4 in the approximation algorithm, discussed in Section III, separates these fractional solutions at a higher cost to optimality. For the heuristic algorithm, more aircraft to be landed implies the nearsighted landing of the most urgent aircraft is more likely to make the wrong decision. This again leads to less optimality for larger numbers of aircraft.

To better evaluate the performance of the polynomial-time algorithms, we look at the ratio between the objective values of either algorithms and the optimal solution. This ratio, which is always at least 1, represents the suboptimality of the algorithm for the given run. In Figure 8(a)-(d), each x-value represents one of 500 runs. The corresponding y-values are the ratio between the heuristic algorithm objective and the CPLEX optimal objective (marked by 'x') and the ratio between the approximation algorithm objective and

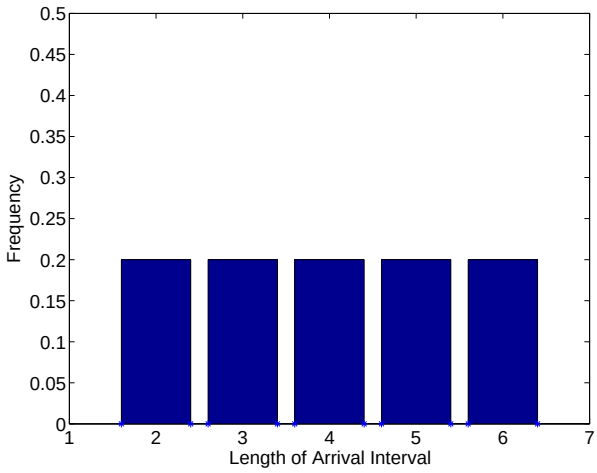


Figure 6. Probability distribution function for length of arrival intervals. That is, lengths l_i are chosen from this distribution, and $b_i = a_i + l_i$.

the CPLEX objective (marked by '+'). The ratios are plotted for all 500 runs for N = 2, 4, 8, and 16 in Figures 8(a)-(d), respectively. In Figure 7(a), we observe that the approximation algorithm is suboptimal only for larger optimal objective values (i.e. when more aircraft are on holding patterns). These suboptimal points are what are seen scattered in the right half of Figure 8(a). For more aircraft, the approximation algorithm performs increasingly poorly, and the corresponding plots (Figure 8(b)-(d)) show more points with ratios above 1.

Histograms are generated for these ratios and shown in Figure 9. In the top half of Figure 9(a), the first column (labelled by (1)) shows the percentage of solutions which are optimal. That is, the approximation algorithm generates the optimal schedule over 80 percent of the time for the 2 aircraft cases. The other columns show that the rest of the runs yield solutions with suboptimality ratios between 1 and 1.4. The bottom half of the plot shows that the heuristic always obtains the optimal schedule for the 2 aircraft cases. Figure 9(b)-(d) show suboptimality ratio histograms for 4, 8, and 16 aircraft. We observe that both polynomial-time algorithms have degraded performance with larger sets of aircraft.

Table 1. Table of ranges of average added delay per aircraft for various suboptimality ratios. Data for this table shown in Figure 10.

Suboptimality Ratio	Average added delay per aircraft (minutes)
1.1	0.5-2.5
1.2	1.8-4.5
1.5	5.0-10.5
2.0	12.5-21.0
2.5	20.8-32.8
3.0	30.4-36.9
3.5	54.0-61.0

Figure 9(a)-(d) are generated from the data in Figure 8(a)-(d), respectively. These plots quantify the observations of the previous paragraph: the approximation algorithm sometimes finds the optimal solution, often performs within 2 times optimal, and rarely performs about 3 times optimal. On the other hand, the heuristic finds the optimal solution from 50 to 100 percent of the time and performs within 1.2 times optimal the rest of the time. Although the idea of a ratio between algorithm objective and optimal objective is useful in terms of bounding suboptimality, a more practical measure is average added delay incurred per aircraft. This metric is obtained by subtracting the optimal objective from the particular algorithm objective and

dividing this added delay amongst the aircraft being scheduled. The result measures to some degree how much delay is incurred due to the suboptimality of the polynomial-time algorithm. For example, a ratio of 2.0 implies the added delay due to the suboptimal algorithm is equal to the optimal objective value. Depending on the number of aircraft and the optimal objective, the average delay incurred per aircraft may vary. In Figure 10, the ordered pairs (suboptimality ratio, average added delay per aircraft) are plotted for all runs. From this data, the information in Table 1 is generated. This table shows ranges of average added delay per aircraft for any given suboptimality ratio.

Overall, the heuristic outperforms the approximation algorithm for the data simulated; however, guarantees on optimality are only available for the approximation algorithm. Also, performance degrades for both algorithms as the number of aircraft is increased. In the next subsection, differences in performance between nominal and weather-affected scenarios are explored.

C. Nominal versus weather-affected scenarios

Simulations were performed to evaluate the performance of the algorithms under weather-affected scenarios, where required spacing Δ is 4 minutes, double the nominal 2 minutes. Figure 11(a) plots the objective values for 8 aircraft in nominal conditions while Figure 11(b) plots objective values for 8 aircraft in adverse weather conditions. These plots are similar to those in Figure 7 where each run shows the objective value for each algorithm, and the runs are sorted by increasing CPLEX objective. It is observed that objective values are much higher for adverse weather conditions; the doubled spacing requirements force more aircraft into holding patterns. From observation it is difficult to tell the relative performance of the polynomial time algorithms in the two different conditions; Figure 12(a)-(b) shows the histogram of ratios for the two algorithms under each weather condition. Again, this figure is similar to Figure 9. From this figure, we see that the heuristic outperforms the approximation algorithm in both nominal and adverse weather conditions. However, the performance of the heuristic is significantly degraded under adverse weather conditions, while that of the approximation algorithm is relatively unchanged.

Similar plots are provided for N=16 in Figures 13 and 14. Again, while the heuristic outperforms the approximation algorithm in both nominal and weather-affected scenarios, we see that the approximation algorithm is more robust to poor weather.

D. Run-time issues

The optimal schedule for landing aircraft can be obtained by software such as CPLEX, but the approximation and heuristic algorithms generate results in polynomial time. In this subsection, we present the run times for the three algorithms, all of which were run on a Sun Blade 2000 workstation with 2 UltraSPARC III+ 900 MHz CPUs and Solaris 8 operating system. Figure 15(a) shows the distribution of run times for the 500 runs for N=2, 4, 8, and 16 aircraft. The distributions are presented as box-and-whisker plots (also known as 5-number summary). The top and bottom of each box-and-whisker plot are the maximum and minimum values of run times for the given set of 500 runs. The top and bottom of the box are the first quartile and third quartile values of the set of 500 run times. The middle of the box is the median value. Figure (b) and (c) shows run times for the approximation and heuristic algorithms, respectively.

From this figure, it is clear that in terms of run time, the heuristic outperforms the other two algorithms in all situations. On average, the approximation algorithm takes less time than the CPLEX algorithm for all numbers of aircraft. Both CPLEX and the approximation algorithm scale poorly as the number of aircraft increase, but the approximation algorithm is guaranteed polynomial. That is, for other scenarios that have not been simulated, we can make guarantees on the run times for the approximation algorithm using data from Figure 15. However, for the CPLEX solution, because complexity is exponential, the data in the figure cannot be used to upper bound run times for any other scenarios. Therefore, in terms of run time, we rank the algorithms from best to worst: heuristic, approximation, and CPLEX. In this section, the simulation methodology, optimality results, weather-affected scenario results, and run-time results have been presented. The heuristic algorithm has performed best empirically, but the approximation algorithm has performed within proven theoretical bounds.

VI. Conclusions

The problem of scheduling arrival aircraft has been formulated as a mixed-integer linear program and then various solutions have been proposed. The solutions computed using a previously proposed approximation algorithm and a new heuristic algorithm are compared with the exact solution computed by CPLEX. These algorithms are evaluated using Monte Carlo simulation based on a plausible air traffic scenario generated from airline timetable data and airport approach maps. Simulations show the reasonable efficacy of both polynomial-time algorithms, though the heuristic algorithm outperforms the approximation algorithm in optimality and run-time. However, the approximation algorithm provides guarantees on suboptimality and performs more robustly under poor weather conditions than the heuristic algorithm. Because CPLEX is a suite of techniques among which the algorithm chooses in order to optimize running time as well as suboptimality, the exponential growth of the runtime of the CPLEX solution compares well with the absolute times of the approximation algorithm. The CPLEX algorithm still is exponential in complexity; even with branch-and-bound techniques, a problem may take exponential time to be solved. In fact, there has been evidence that this is the case on specific Air Traffic Control problems.[4] By construction, the heuristic algorithm generates schedules many times faster than either of the other two algorithms. The heuristic algorithm is suboptimal with ratio less than 1.5 (in practice), or an equivalent delay of less than 10 minutes per aircraft, while taking on the order of 100 times less time to complete than CPLEX. The approximation algorithm is more suboptimal and takes longer to generate schedules but provides guarantees on optimality which the heuristic cannot provide. Empirically, however, the heuristic outperforms the approximation algorithm and rivals CPLEX due to its very fast run time. A plausible improvement would be to combine the heuristic and the approximation algorithm, thus meeting the 5-times-suboptimal guarantee while substantially improving performance (empirically less than 1.5-times-suboptimal).

One direction for future work is to add heuristics to the approximation algorithm to improve solutions which have been generated. Large gaps in the schedule are created to help prove the 5 times optimal guarantee, but heuristics can be used to reduce this suboptimality for little additional computational cost. Other future work consists of identifying scenarios in which the approximation and heuristic algorithms can be shown analytically to perform more optimally. That is, we would like to identify scenarios in which the optimality bound can be made tighter.

Another topic toward which this topic can be extended is game theory and airline competition. These optimization algorithms are innovative because they solve the scheduling problem globally, but the algorithms fail to take into account other notions of fairness in competition. Experienced air traffic controllers often land an aircraft to help make connections or to balance out unfairness to a specific airline; using these techniques in a game theoretic setting would allow the exploration of these ideas of fairness in scheduling arrival aircraft.

References

[1]P. Baptiste. Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine when processing times are equal. *Journal of Scheduling*, 2:245–252, 1999.

[2]A. M. Bayen, P. Grieder, H. Sipma, G. Meyer, and C. J. Tomlin. Delay predictive models of the National Airspace System using hybrid control theory. In *Proceedings of the 2002 American Control Conference*, Anchorage, AK, May 2002.

[3]A. M. Bayen and C. J. Tomlin. Real-time discrete control law synthesis for hybrid systems using MILP: applications to congested airspaces. In *Proceedings of the 2003 American Control Conference*, Denver, CO, May 2003.

[4]A. M. Bayen, C. J. Tomlin, Y. Ye, and J. Zhang. Polynomial time algorithm for a MILP formulation of an aircraft scheduling problem. In *Proceedings of the* 42^{nd} *IEEE Conference on Decision and Control*, Maui, HI, Dec. 2003.

[5]A. M. Bayen, C. J. Tomlin, Y. Ye, and J. Zhang. An approximation algorithm for scheduling aircraft with holding time. In *Proceedings of the* 43^{th} *IEEE Conference on Decision and Control*, Nassau, Bahamas, Dec. 2004.

[6]R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: a modeling language for mathematical programming.* Boyd and Fraser, Danvers, MA, 1999.

[7]J. Krozel and S. Penny. Comparison of algorithms for synthesizing weather avoidance routes in transition airspace. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Providence, Rhode Island, Aug. 2004.

[8]F. Neuman and H. Erzberger. Analysis of sequencing and scheduling methods for arrival traffic. Technical report, NASA Ames Research Center, Moffett Field, CA, NASA-TM-102795 1990.

[9]J. Prete and J. Mitchell. Safe routing of multiple aircraft flows in the presense of time-varying weather data. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Providence, Rhode Island, Aug. 2004.

[10]`http://www.ilog.com/products/cplex/`.

[11]`http://www.aa.com/content/travelInformation/airportAmenities/electronicTimetable.jhtml`.

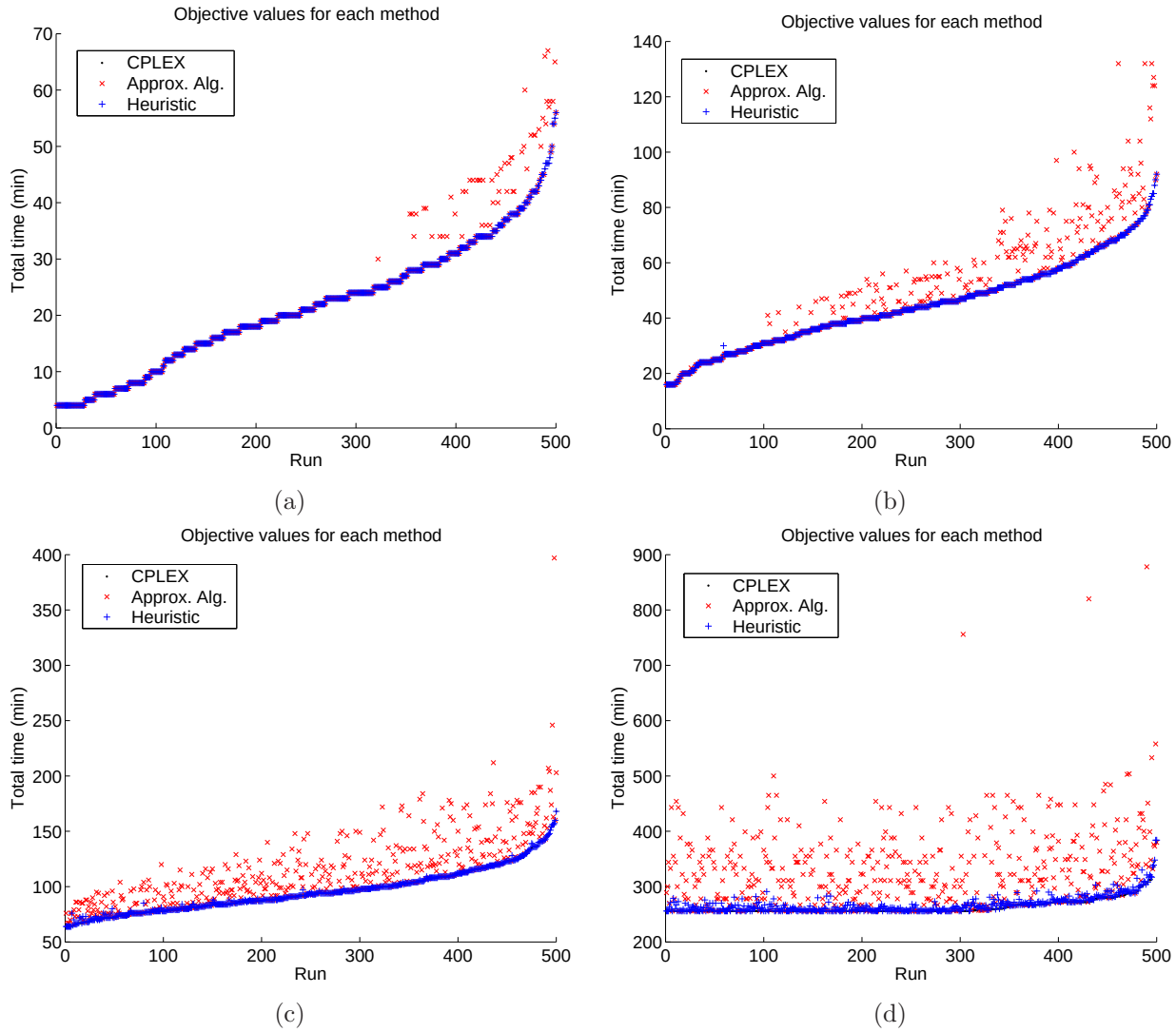


Figure 7. Objective values (i.e. sum of arrival times) for each of 500 runs for CPLEX, approximation, and heuristic algorithms for required spacing $\Delta = 2$, holding pattern time T=10, and different numbers of aircraft N. $N =$ (a) 2; (b) 4; (c) 8; (d) 16. The runs are sorted by increasing CPLEX objective value, but a given x-value shows results of each algorithm on the same parameter values.

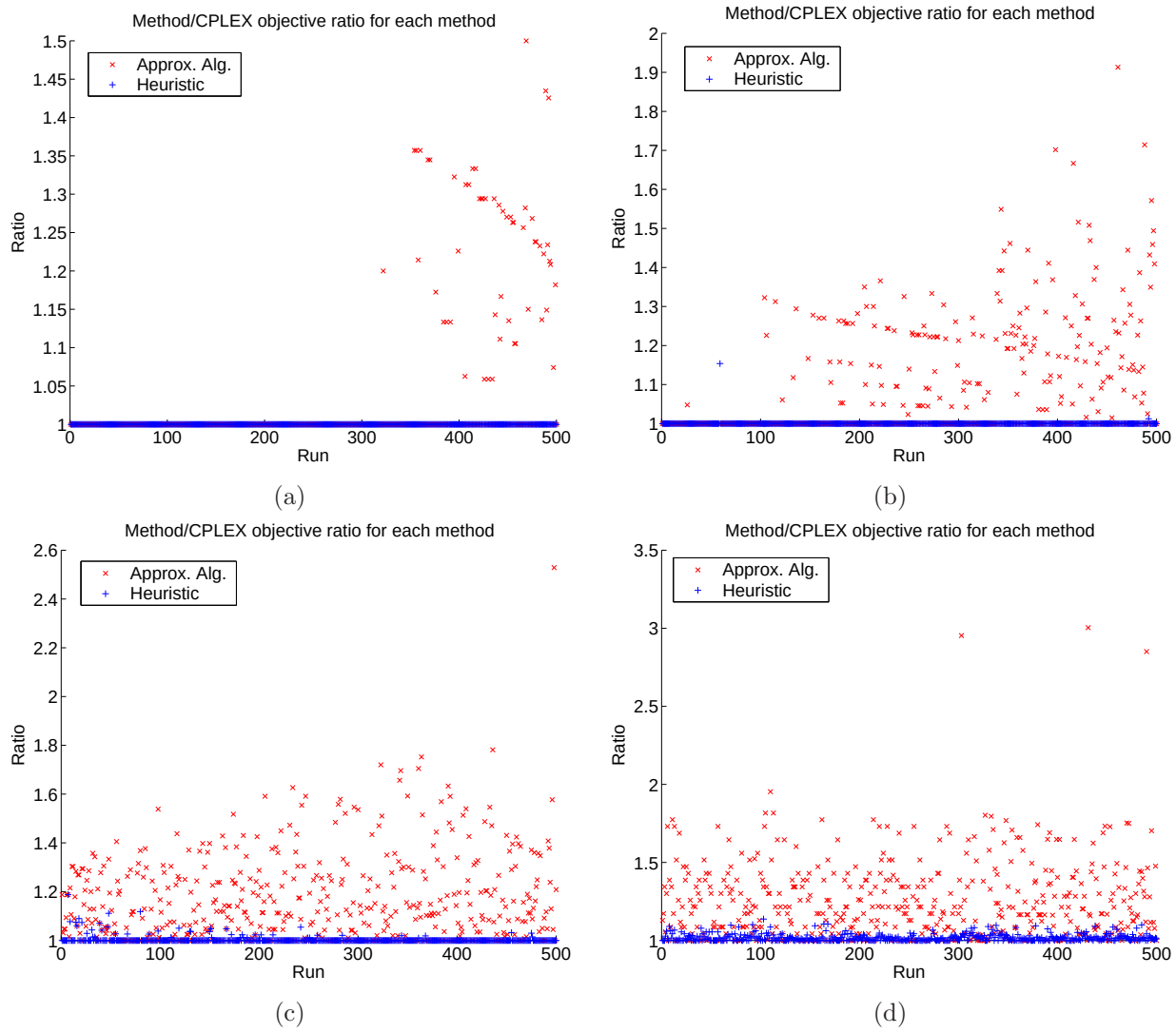


Figure 8. Ratios between polynomial-time algorithms and CPLEX solution for each of 500 runs for $\Delta = 2$, T=10, and varying N. $N =$ (a) 2; (b) 4; (c) 8; (d) 16.

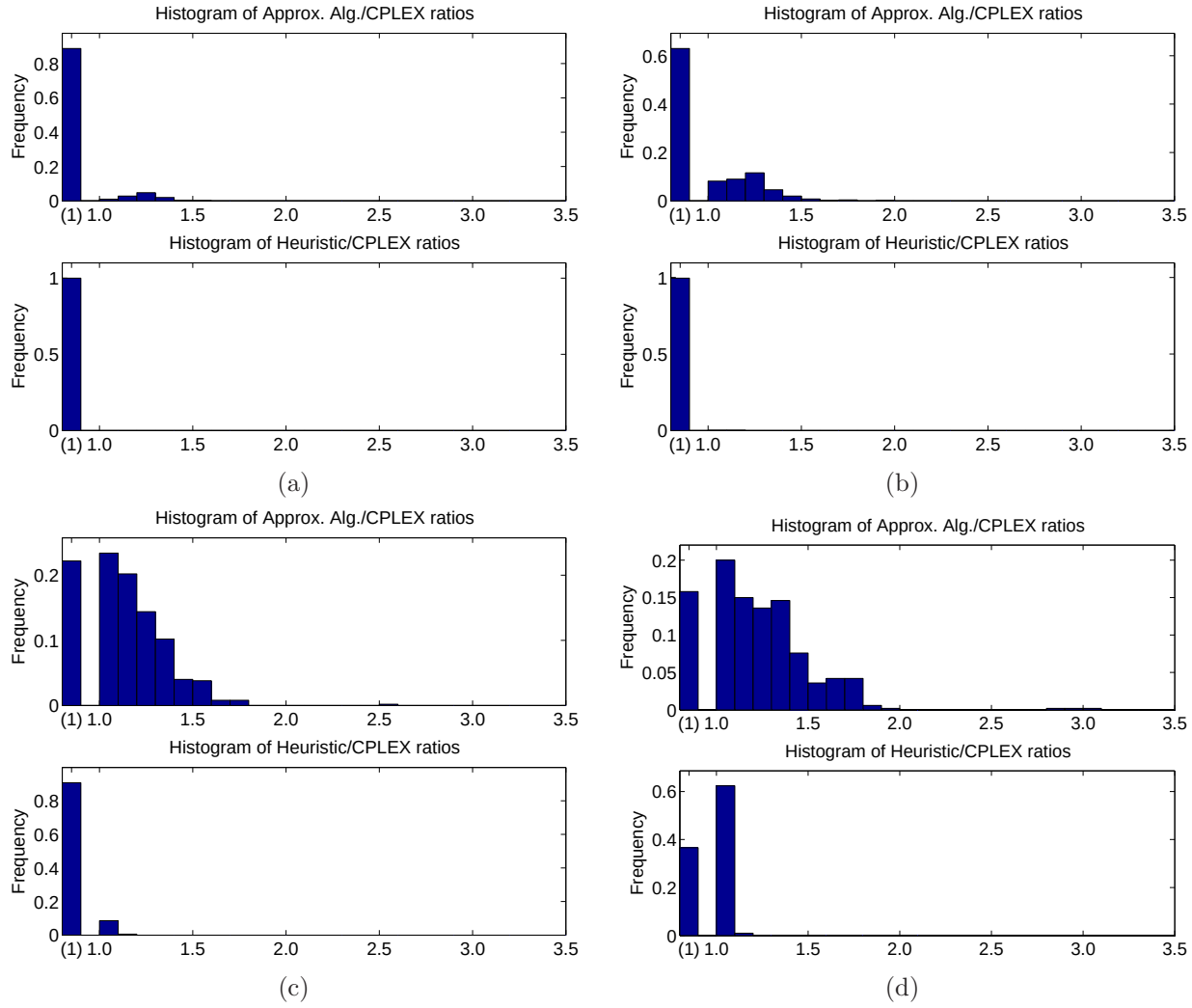


Figure 9. Histogram of distribution of ratios between polynomial-time algorithms and CPLEX solution for $\Delta = 2$, T=10, and varying N. $N =$ (a) 2; (b) 4; (c) 8; (d) 16.

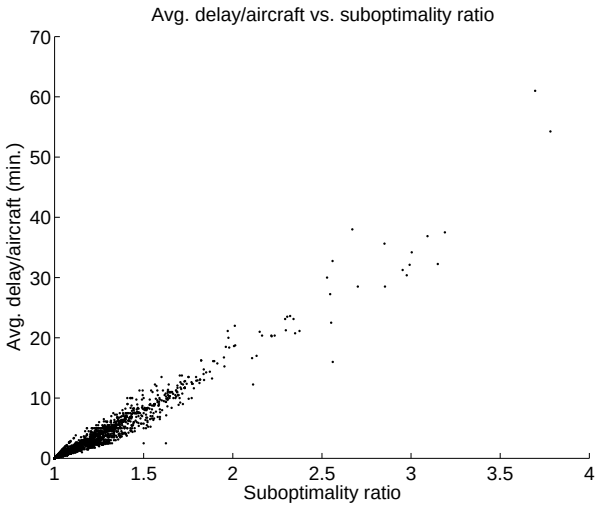
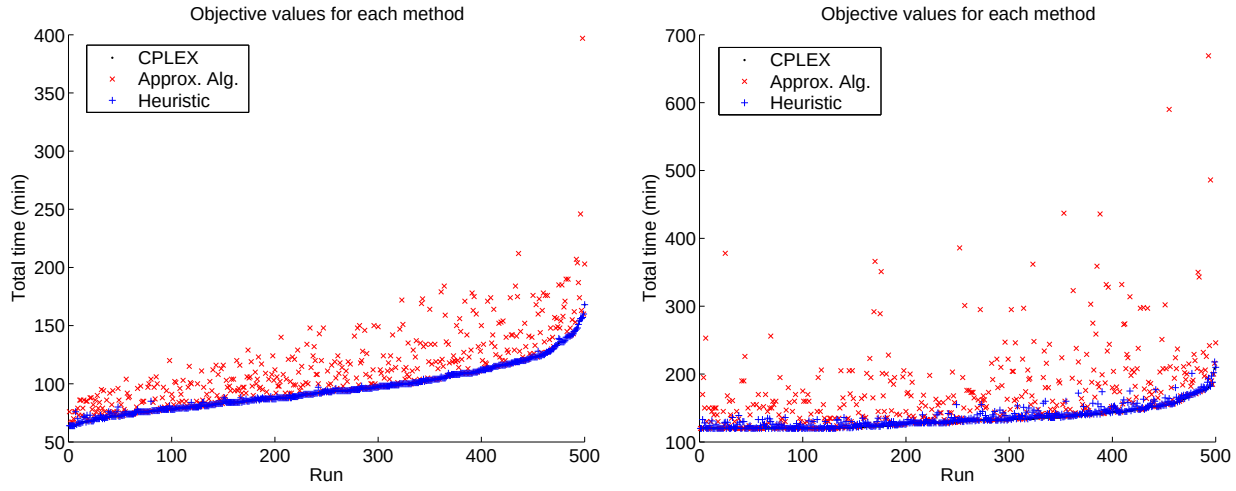


Figure 10. Average added delay per aircraft versus suboptimality ratio for all simulated runs.



(a) (b)

Figure 11. Objective values for CPLEX, approximation, and heuristic algorithms for $N = 8$, T=10, and $\Delta =$ (a) 2 or (b) 4.

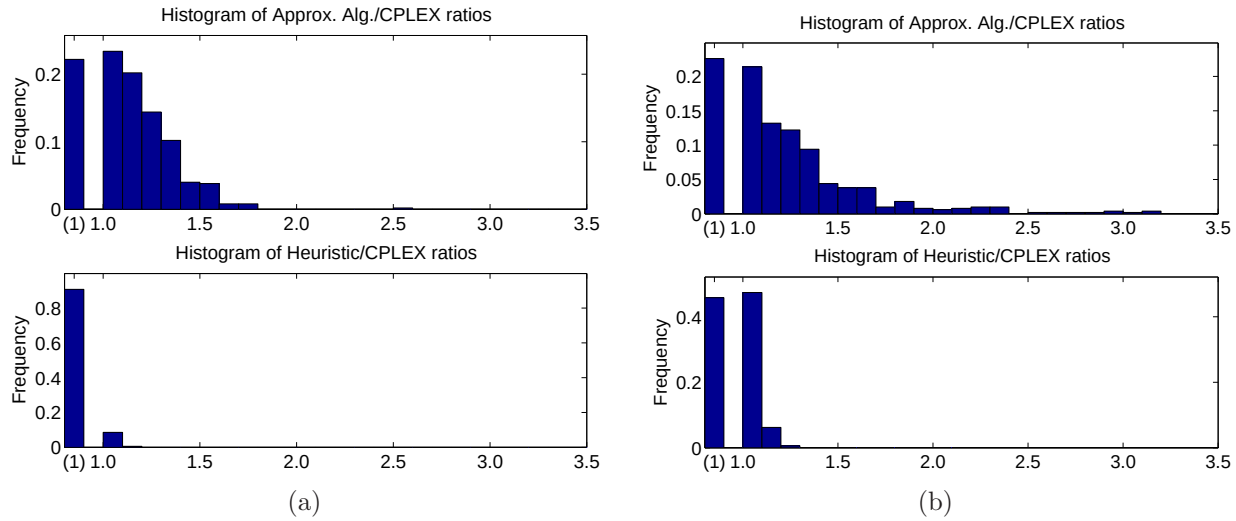


Figure 12. Histogram of distribution of ratios between polynomial-time algorithms and CPLEX solution for $N = 8$, T=10, and $\Delta =$ (a) 2 or (b) 4.

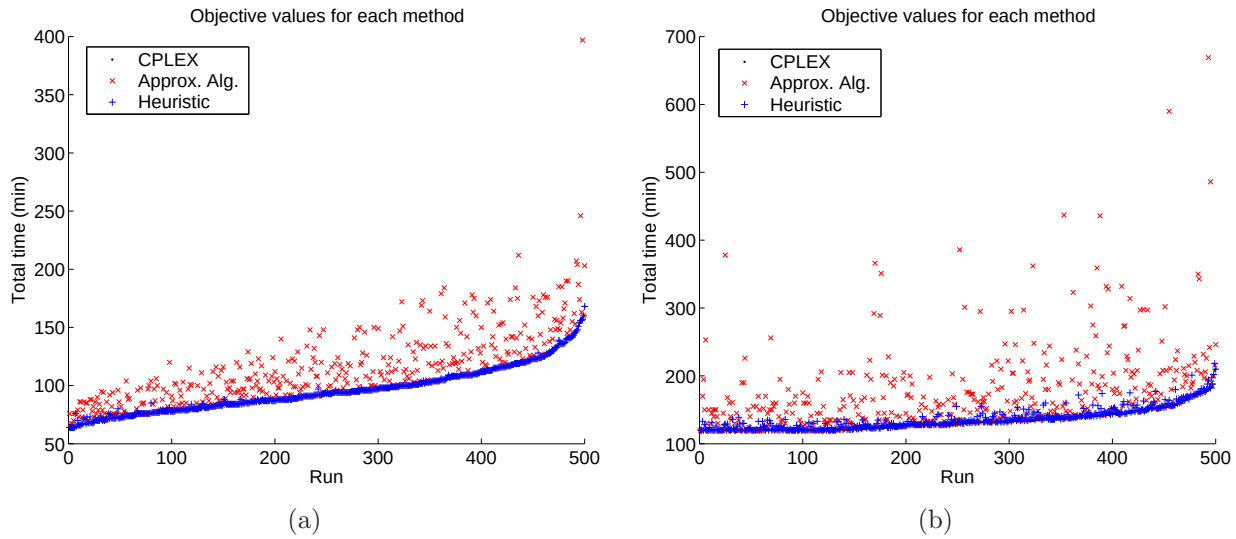


Figure 13. Objective values for CPLEX, approximation, and heuristic algorithms for $N = 16$, T=10, and $\Delta =$ (a) 2 or (b) 4.

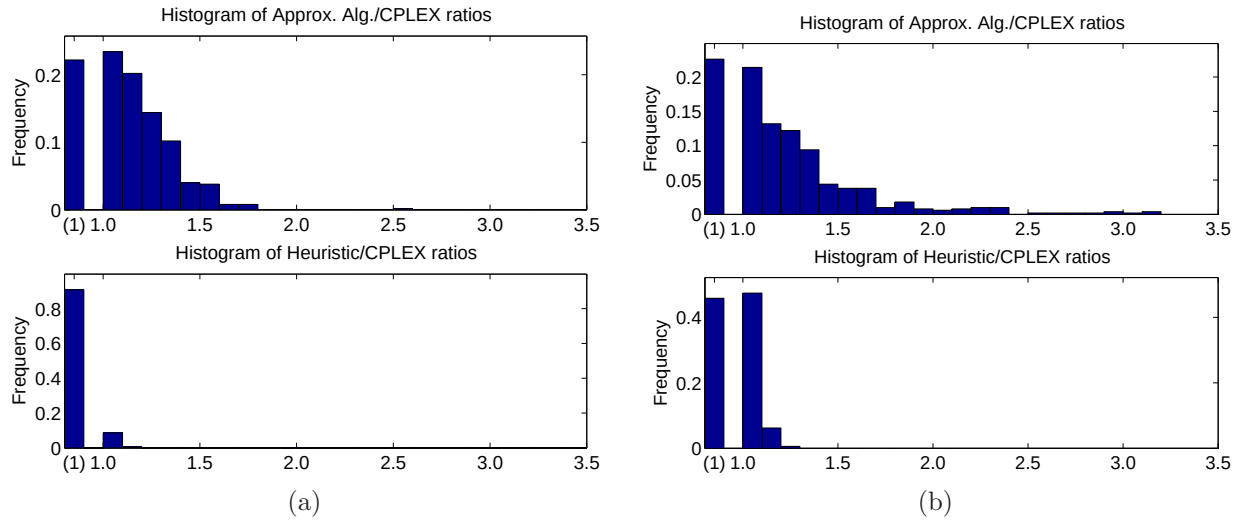


Figure 14. Histogram of distribution of ratios between polynomial-time algorithms and CPLEX solution for $N = 16$, T=10, and $\Delta =$ (a) 2 or (b) 4.

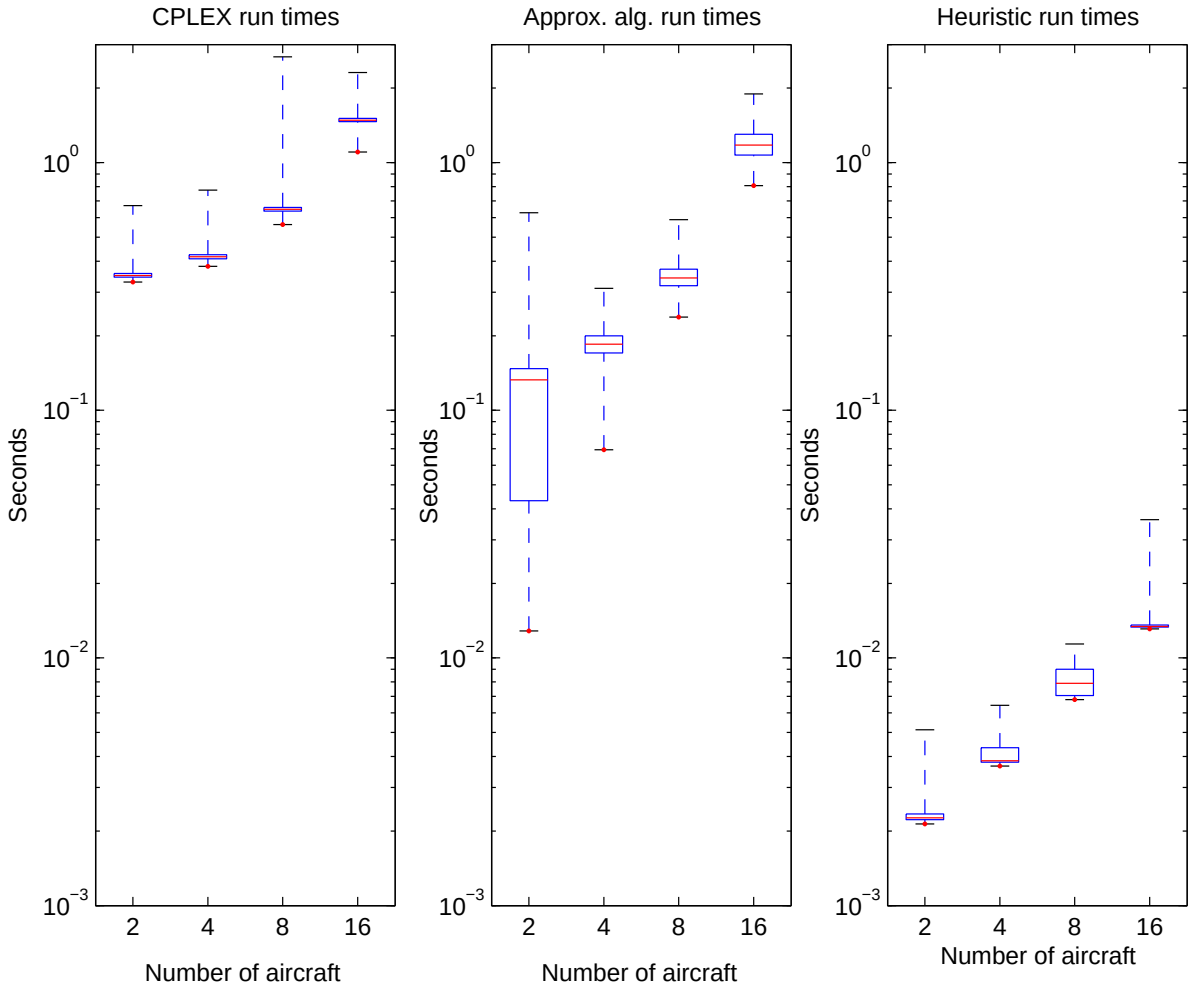


Figure 15. Run times for (a) CPLEX, (b) approximation, and (c) heuristic algorithms for $N = 2, 4, 8, 16$, T=10, and $\Delta = 2$. Data summarized using box-and-whisker plots.