On the Problem of Pairing Aircraft for Closely Spaced Parallel Approaches

Amir H. Farrahi, Savita A. Verma, and Thomas E. Kozon

***Abstract*—The problem of scheduling pairs of aircraft for simultaneous landing onto very closely spaced parallel runways is studied. The pair scheduling problem and its generalization group scheduling problem for simultaneous landing onto parallel runways are formulated and shown to be $\mathcal{NP}$-hard, in general. A genetic pairing scheduler algorithm is developed, capable of handling a wide range of constraints, and used in a real-time human-in-the-loop simulation that was carried out to study the operational concept. Experimental data from these simulations as well as an extensive set of stress tests are presented and analyzed. Results indicate that while the problem is $\mathcal{NP}$-hard in general, practical instances of the algorithm are not necessarily very hard to solve. As such, the proposed algorithm succeeded in finding and suggesting aircraft pairs that met all the problem constraints and thus were accepted by the controllers in over 97% of the cases. High solution quality, scalable runtime, and flexibility of the proposed algorithm in handling different constraints, suggest it is a suitable candidate for use in a real-time application.**

***Index Terms*—Closely spaced parallel runways, parallel approaches, pairing aircraft, scheduling aircraft landing.**

I. Introduction

INCREASING the airport arrival rates is an important factor towards meeting the growing demand in air traffic [2]. In the implementation of the Next-Generation Air Transportation System (NextGen), the concept of Very Closely Spaced Parallel Runway (VCSPR) operations is considered a crucial step for realizing significant increase in arrival throughput under instrument meteorological conditions (IMC) [3]. It aims to maximize utilization for parallel runway systems that may be spaced as closely as 750 ft apart, and thus increase the landing capacity at hub airports during poor weather conditions, without significant increase to the airport footprint.

Several operational concepts have been developed for simultaneous approaches [4–6]. All of these concepts assume that the aircraft pairs are identified and paired by the air traffic controllers before the approach clearances are issued to the aircraft. In the current National Airspace System (NAS), there are no formal tools, processes or metrics for pairing aircraft for simultaneous landing. This paper attempts to fill this gap by formulating, studying, and providing a practical solution for the problem of identifying aircraft pairs automatically in the arrival-scheduling context.

While there has been much research on arrival scheduling, studies applicable to scheduling aircraft for simultaneous parallel approaches are relatively scarce. This paper presents a general formulation for the pair-scheduling problem and its extension to group scheduling for the multiple-runway condition, and shows that these problems are $\mathcal{NP}$-hard, in general, in the strong sense. Furthermore, an algorithm is developed, tested, and used successfully by air traffic controllers during Human-in-the-Loop (HITL) simulation runs, conducted to study the role of the air traffic control in pairing aircraft for simultaneous approach [7].

The rest of this paper is organized as follows. First, we describe the operational concept, motivation for this work, and provide an overview of the previous work. Next, the formulation of the pair-scheduling problem is presented, followed by a discussion of its computational complexity in the context of the broader scheduling problems. The considerations and rationale for the adopted solution methodology are provided next, followed by a detailed description of our pairing algorithm. Experimental results from HITL simulations as well as a large number of stress test scenarios are presented and discussed next. The paper ends with some concluding remarks and a discussion of some practical considerations suggested for future work.

II. Background

The Federal Aviation Administration (FAA) recognizes that significant capacity is lost when simultaneous operations performed under visual conditions are not operational under poor weather conditions [8]. As a part of its NextGen plan, the FAA aims to reduce the minimum allowable spacing between runways used for simultaneous operations in poor visibility – currently 4300 ft. – by implementing revised standards and improved technologies [3]. Table 1 shows a partial listing of major US airports containing parallel runways with spacing below 4300 ft. All of these airports experience severe reduction in aircraft landing capacities under IMC.

A. Related Concepts

Several concepts that address and could benefit from the revision of separation standards and technologies include Simultaneous Offset Instrument Approaches (SOIA) [9], Airborne Information Lateral System (AILS) [5] and Terminal Area Capacity Enhancing Concept (TACEC) [6]. The role of the air traffic controllers during simultaneous approaches is different for each of the above concepts. However, all of these concepts assume that the air traffic controller will select the

A.H. Farrahi is with NASA Ames Research Center, Moffett Field, CA 94035, USA (phone: 650-604-5838; email: amir.h.farrahi@nasa.gov).

S.A. Verma is with NASA Ames Research Center, Moffett Field, CA 94035, USA (email: savita.a.verma@nasa.gov).

T.E. Kozon is with NASA Ames Research Center, Moffett Field, CA, 94035, USA (email: thomas.e.kozon@nasa.gov).

A preliminary version of this paper appeared in [1].

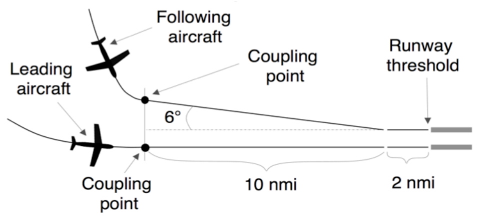

Fig. 1. Approach pattern for an aircraft pair.

aircraft for paired arrival with the assumption that they are properly equipped. Currently, there does not exist any formal tool or process in the NAS for pairing aircraft.

Recent HITL simulations investigated the allocation of tasks under TACEC for air traffic controllers [7]. This required an algorithm for pairing aircraft under different levels of automation to investigate the appropriate human/automation mix for the given task. The highest level of automation required a pairing algorithm that would identify and suggest aircraft pairs to the controllers. This paper describes the problem formulation and the development of the pairing algorithm needed for these HITL simulations. In these simulations, finding and suggesting pairs was handled by automation while controllers handled the final evaluation of the suggested pairs in order to accept or reject them. The pairing was achieved by building a feasible schedule in which as many aircraft as possible were scheduled for paired arrival.

B. Concept Description

The concept investigated in the current study is TACEC [6]. It allows multiple aircraft to fly in close formations during the final approach to enable simultaneous instrument approaches for landing aircraft onto very closely spaced parallel runways that are as close as 750ft apart. This would increase the landing capacity of the airports with closely spaced parallel runways during low visibility conditions, achieving arrival rates comparable to visual approach operations [6].

The process involves pairing the aircraft approximately 30 minutes before their arrival at the terminal boundary. As shown in Fig. 1, the actual coupling for the approach is set to occur 12nmi from the runway threshold. After this *coupling point*, the paired aircraft converge over a 10nmi distance at a 6° angle. For the last 2nmi prior to the runway threshold, the aircraft fly on parallel flight path segments.

After crossing the coupling point, the following aircraft in a pair must be flying within a *safe zone* [10], which is defined by a Lower Pairing Boundary (LPB) and an Upper Pairing Boundary (UPB) behind the lead aircraft, as illustrated in Fig. 2. The LPB is defined to minimize the risk of collision in case of a blunder by the lead aircraft, while the UPB is defined to keep the following aircraft ahead of the wake vortex of the lead aircraft. In this study, the LPB and UPB are set to 5s and 25s behind the lead aircraft, respectively. The pairing algorithm is instructed to schedule the pairs in order for the

TABLE I
SOME AIRPORTS WITH PARALLEL RUNWAYS BELOW 4300 FT SPACING

AIRPORT	RUNWAY SPACING	AIRPORT	RUNWAY SPACING
Los Angeles (LAX)	750 ft	St. Louis (STL)	1200 ft
San Francisco (SFO)	750 ft	Boston (BOS)	1300 ft
Seattle (SEA)	800 ft	Orlando (MCO)	1500 ft
Newark (EWR)	900 ft	New York (JFK)	1600 ft
Houston (IAH)	1000 ft	Minneapolis (MSP)	3000 ft
Las Vegas (LAS)	1000 ft	Memphis (MEM)	3380 ft
Atlanta (ATL)	1000 ft	Raleigh (RDU)	3400 ft
Pittsburgh (BIT)	1200 ft	Phoenix (PHX)	3650 ft

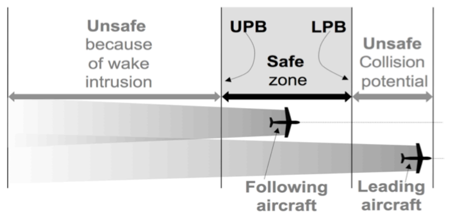

Fig. 2. Illustration of safe zone for the following aircraft.

follower aircraft to be situated in the middle of the safe zone. The concept assumes Differential Global Positioning System (DGPS), augmented ADS-B, four-dimensional flight management system (4D-FMS), wind detection sensors onboard the aircraft, and cockpit automation. To enable the following aircraft in a pair to reach the 15 sec. (+/-10 sec) window behind the leading aircraft, the flight deck merging and spacing algorithm ASTAR [11] was used and integrated into system.

To ensure safe operation, a minimum separation is maintained between landing aircraft that do not belong to the same pair. This separation is determined based on the wake separation category of the aircraft involved. The following four categories are recognized: Small (S), Light (L), B757 (7), and Heavy (H). The enforced wake separation in seconds for various combinations is provided in a wake separation matrix shown in Table 2.

Another important prerequisite for the HITL simulations was a careful redesign of the airspace so that the arrival traffic could safely follow their prescribed 4D trajectories from their respective arrival routes or *streams* [7]. This involved a split towards the end of the arrival streams to enable routing the aircraft arriving from either stream to land on either of the two runways involved. The aerial view of the redesigned airspace used in this study, along with the geometry of the five arrival streams are shown in Fig. 3 in the vicinity of runways 28L and 28R at San Francisco International Airport (SFO). The final portion of the paths starting from the coupling point and ending on the runways is shown in light green. This portion corresponds to the approach patterns of an aircraft pair illustrated in Fig. 1.

C. Scheduling Problem

Scheduling problems have numerous industrial applications and thus have been studied extensively. A general survey and classification of scheduling problems can be found in [12, 13], while [14, 15] present an overview of the scheduling problems

TABLE 2
WAKE SEPARATION MATRIX (SECONDS)

		Following Aircraft			
		S	L	7	H
Leading Aircraft	S	98	83	83	72
	L	147	83	83	72
	7	180	125	125	106
	H	213	152	152	106

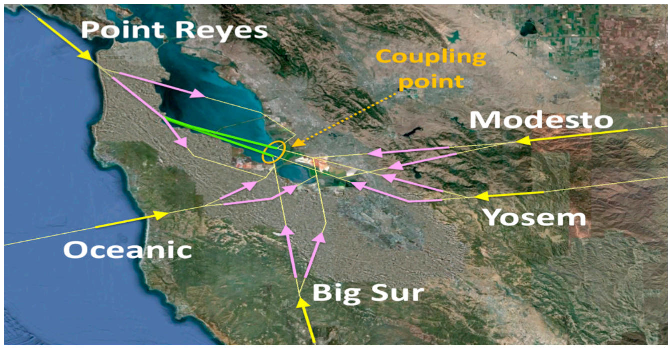

Fig. 3. Arrival stream geometries.

encountered in aircraft and airline scheduling. Scheduling problem for arrival aircraft has also been studied to a great extent. The idea of Constrained Position Shifts (CPS) was introduced in [15] as a means for improving the arrival sequence, while [16] presented a dynamic programming algorithm to optimize the aircraft arrival sequence and took advantage of CPS to increase efficiency. Other heuristics and approaches using GA [17-20], dynamic programming [21], graph search [22], mixed-integer linear programming with linear relaxations [23, 24], branch and bound techniques [25] or a combination of the above have also been applied to the arrival-scheduling problem.

Arrival-scheduling problems are often broadly categorized into static (offline) [24] and dynamic (online) [19], depending on whether or not the set of aircraft that are going to land is completely known ahead of time. In this paper, the focus is on the static case. Another classification of the arrival scheduling problems is into single vs. multiple runway landing. While much of the previous research on scheduling arrival landing focused on single runway, there are some studies including [23, 25] that discuss the multiple runway case or extend their single-runway approaches to the multiple-runway case.

The scheduling problem for simultaneous landing of aircraft to very closely spaced parallel runways is markedly different and more constrained than scheduling for independent landing onto multiple runways. A simplified version of his problem was formulated in [26] and solved using a mixed integer linear programming (MILP) formulation. As is often the case, the runtime achieved using the proposed MILP formulation is prohibitively long, making the approach unsuitable for real-time application. Therefore, a heuristic technique based on genetic algorithm [27] was also developed.

III. Formulation of the Problem

Let $\boldsymbol{Y}=\{l, r\}$ represent the set of very closely spaced parallel runways (left and right) used for landing the aircraft, and $\boldsymbol{M}=\{m_1, m_2, \ldots, m_k\}$ represent the set of arrival streams. In the current study, we have $\boldsymbol{M}=\{big\_sur, modesto, oceanic, point\_reyes, yosem\}$, as shown in Fig 3. The set of arriving aircraft is denoted by $\boldsymbol{A}=\{\boldsymbol{a_1}, \boldsymbol{a_2}, \ldots, \boldsymbol{a_n}\}$, where $\boldsymbol{a_i} = (ETA_{N,i}, [ETA_{EL,i}, ETA_{LL,i}], [ETA_{ER,i}, ETA_{LR,i}], w_i, s_i, g_i)$ is the record corresponding to aircraft i. $ETA_{N,i}$ is the nominal estimated time of arrival (ETA) of aircraft i at the coupling point. $[ETA_{EL,i}, ETA_{LL,i}]$ and $[ETA_{ER,i}, ETA_{LR,i}]$ are the early and late ETA of aircraft i at the left and right coupling points, respectively. $w_i \in \{S, L, 7, H\}$, $s_i \in \boldsymbol{M}$, and g_i are the wake category, the arrival stream, and the pair-group category for aircraft i, respectively. The wake separation matrix $\boldsymbol{W}$ is a $4x4$ wake matrix as shown in Table 2, where entry w_{ij} is the required safe separation between non-paired aircraft of types i and j, with $i, j \in \{S, L, 7, H\}$ and i is ahead of j. The target separation window between the scheduled times of arrival (STAs) of two aircraft to be paired is denoted as t_p. For the purpose of this study, the wake separation matrix given in Table 2 is enforced, and the pair separation window is set to t_p= 15 seconds.

A scheduling solution consists of an STA value at the coupling point, and a runway assignment $r \in \boldsymbol{Y}$ for a (maximal) subset of A. A *feasible* scheduling solution is one satisfying a set of enforced *constraints*. Several classes of constraints are defined, including *temporal*, *sequencing*, *separation*, *pairing*, and *runway assignment* constraints.

A *temporal constraint* places a restriction on the acceptable range for the STA values of the aircraft being scheduled for landing. In the context of this study, the STA for an aircraft should lie within the ETA time window at the coupling point for its target runway.

A *sequencing constraint* places a restriction on the aircraft arrival sequence for the runways. In other words, it limits the acceptable order for the arriving aircraft as they pass through their respective coupling points. In this study, no overtaking is allowed among aircraft arriving from the same stream. Hence, within each arrival stream, the order of STA values for arriving aircraft to the two runways should correspond to the order of the aircraft's nominal ETA values at the coupling point.

A *separation constraint* defines the minimum safe distance among the aircraft as they fly through the airspace. In this study, the wake separation matrix and the wake categories of the arriving aircraft define the separation constraints. The separation between arriving and non-arriving aircraft is a complication that was not considered in the current study. However, part of the requirements for redesigning the airspace and the arrival trajectories was to minimize the potential occurrence of such separation conflicts.

Pairing constraints impose restrictions on the set of aircraft that are allowed to pair with one another. They can be used for different reasons, for instance, to prevent aircraft with grossly incompatible approach speeds to be paired with one another. In this study, pairing aircraft from the same arrival stream was not allowed.

Runway assignment constraints define the legal runways for the arriving aircraft to land on. In the two-runway case, two types of runway assignment rules are distinguished: single aircraft runway assignment rules, and paired aircraft runway assignment rules. The set $\boldsymbol{R}$ of runway assignment rules is the union of the set $\boldsymbol{R_1}=\{(m, y) \mid m \in \boldsymbol{M}, y \in \boldsymbol{Y}\}$ of single aircraft runway assignment rules and the set $\boldsymbol{R_2}=\{((m_1, y_1), (m_2, y_2)) \mid m_1, m_2 \in \boldsymbol{M} \text{ and } y_1, y_2 \in \boldsymbol{Y}\}$ of paired aircraft runway assignment rules, that is $\boldsymbol{R} = \boldsymbol{R_1} \cup \boldsymbol{R_2}$. A single aircraft runway assignment rule $(m, y) \in \boldsymbol{R_1}$ indicates that a single aircraft arriving from stream m should be scheduled to arrive at runway y, while an aircraft-pair runway assignment rule $((m_1, y_1), (m_2, y_2)) \in \boldsymbol{R_2}$ indicates that two arrival aircraft that are going to be paired with the leader and follower arriving from streams m_1 and m_2, should be assigned to runways y_1 and y_2 respectively. Additionally, when there is a conflict, a paired aircraft runway assignment rule overrides the single aircraft runway assignment rule. In general, we can define group aircraft runway assignment rules, to accommodate complex runway assignment rules for multiple runway systems.

The pairing and runway assignment rules for this study, were refined iteratively during several rounds of experiments and discussions among the researchers and subject matter experts. The final set of pairing rules adopted for this research can be expressed as the union of single aircraft runway assignment rules $\boldsymbol{R_1}$ and the paired aircraft runway assignment rules $\boldsymbol{R_2}$, as follows: $\boldsymbol{R} = \boldsymbol{R_1} \cup \boldsymbol{R_2}$, where:

$\boldsymbol{R_1}$ = { (*big_sur*, *l*), (*modesto*, *r*), (*oceanic*, *l*), (*point_reyes*, *l*), (*yosem*, *r*)}

$\boldsymbol{R_2}$ = { ((*point_reyes*, *l*), (*oceanic*, *r*)), ((*point_reyes*, *l*), (*big_sur*, *r*)), ((*oceanic*, *l*), (*big_sur*, *r*)), ((*yosem*, *l*),(*modesto*, *r*))}

The rules in $\boldsymbol{R_1}$ indicate that single aircraft arriving from **Big Sur**, **Oceanic**, and **Point Reyes** stream should be assigned to the left runway, while those arriving from **Modesto** and **Yosem** streams should be assigned to the right runway. The rules in $\boldsymbol{R_2}$ indicate how the rules in $\boldsymbol{R_1}$ are over-ridden when the aircraft are paired. For example, the first rule ((*point_reyes*, *l*), (*oceanic*, *r*)) indicates that for an aircraft pair arriving from streams **Point Reyes** and **Oceanic**, the aircraft arriving from **Point Reyes** should land on the left runway and the aircraft arriving from **Oceanic** should land on the right runway.

The Pair-Scheduling Problem (PSP) for landing single and paired aircraft on closely spaced parallel runways can be formulated as follows:

PSP *Instance*: A tuple of the form ($\boldsymbol{A}$, $\boldsymbol{W}$, t_p, $\boldsymbol{R}$), where $\boldsymbol{A}=\{\boldsymbol{a_1}, \boldsymbol{a_2}, \ldots, \boldsymbol{a_n}\}$ is a set of aircraft records, $\boldsymbol{W}$ is a *4x4* wake separation matrix, t_p is the *pairing window* or the target separation window between two paired aircraft as they cross their respective coupling points, and $\boldsymbol{R}$ is the set of runway assignment rules.

PSP *Objective*: To find a feasible scheduling solution of a maximal subset of $\boldsymbol{A}$ subject to the temporal, sequencing, separation, pairing, and runway assignment constraints.

The Pair-Scheduling Problem can be extended and generalized to the multiple-runway situation where there would be a set of *b* closely spaced parallel runways that could accommodate coordinated arrival of up to *b* aircraft simultaneously. We will refer to the resulting problem as the *b-runway Group-Scheduling Problem* or *b*–GSP.

IV. Computational Complexity

One of the fundamental tasks in studying any problem is to determine its computational complexity. In this section, we will show the PSP and *b*-GSP problems are $\mathcal{NP}$-hard in the strong sense [28], in general.

Recently, Helmke [22] proposed heuristics based on graph search, assuming that the objective function behaves *monotonically* in the STA values of the arriving aircraft. For such objective functions, the scheduling problem may be solved in two phases: sequencing followed by packing. In the sequencing phase, the best arrival sequence is identified, and in the packing phase, the aircraft are packed greedily in the order prescribed by the sequence identified earlier. Note, however, that in general the objective functions do not behave monotonically, and hence this two-phase solution strategy may not be directly applicable.

Consider, the PSP problem formulated above, and define objective function so as to find the maximal subset of the aircraft that can be feasibly scheduled while minimizing the *makespan*, defined as the difference between the smallest and largest STA value among the feasibly scheduled aircraft. Consider the problem instance shown in Fig. 4. Assume pairing window of t_p=15 seconds, and that the three aircraft *A*, *B*, *C* in the problem instance are expected to arrive at the coupling point during ETA intervals I_A = [0, 50], I_B = [40, 150], and I_C = [100, 270], respectively. Assume *A*, *C* belong to the small (**S**) wake category, while *B* belongs to the heavy (**H**) wake category. Furthermore, assume that the arrival sequence *A* before *B*, before *C*. As illustrated in Fig. 4, in order to achieve the optimal makespan, it is best to delay aircraft *B* to pair it up with *C*, rather than to speed it up to pair it with *A*. This means that the makespan is not a monotonic function of the STA values.

Wake Separation Matrix

W	Following Aircraft				
		S	L	7	H
Leading Aircraft	S	98	83	83	72
	L	147	83	83	72
	7	180	125	125	106
	H	213	152	152	106

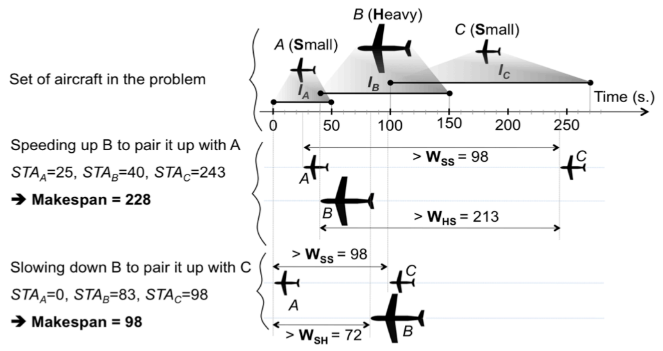
B (Heavy)
A (Small)
C (Small)
Set of aircraft in the problem
I_A
I_B
I_C
Time (s.)
0
50
100
150
200
250
Speeding up B to pair it up with A
STA_A=25, STA_B=40, STA_C=243
➔ Makespan = 228
$> W_{SS} = 98$
A
C
B
$> W_{HS} = 213$
Slowing down B to pair it up with C
STA_A=0, STA_B=83, STA_C=98
➔ Makespan = 98
$> W_{SS} = 98$
C
A
B
$> W_{SH} = 72$

Fig. 4. An example to illustrate that makespan minimization does not behave as a monotonic objective function.

Beasley et al. [24] studied scheduling of arrival aircraft. They formulated *aircraft-landing problem* and showed it is $\mathcal{NP}$-hard, by observing that it can be viewed as a job-shop scheduling problem. Note that the aircraft-landing problem involves independent landing of a set of arriving aircraft. On the other hand, problems studied in this paper allow scheduling for simultaneous landing of arriving aircraft, involving different sets of constrains.

In the rest of this section, we will make use of the three-field notation introduced by Graham [13] for classifying scheduling problems. Briefly, a scheduling problem can be formulated as one in which a set $\boldsymbol{M} = \{M_1, M_2, \ldots, M_m\}$ of *machines* (or processors) need to process a set $\boldsymbol{J} = \{J_1, J_2, \ldots, J_n\}$ of jobs subject to a set $\boldsymbol{C} = \{C_1, C_2, \ldots, C_k\}$ of constraints, in order to optimize certain objective function. Graham's notation or classification scheme uses three fields α, β and γ to identify a scheduling problem as α | β | γ, where α signifies the processor or machine environment and constraints, β signifies the job specification environment and/or constraints, and γ signifies the objective function. Each of these fields is a semicolon-separated list of attributes signifying the characteristics defining that field. In the context of scheduling for arriving aircraft, the set of jobs correspond to the set of aircraft that are to be scheduled for landing, and the runways correspond to the processors or machines. However, in the scheduling problem for parallel landing of aircraft to closely spaced parallel runways, the landing of the aircraft on the multiple runways needs to be coordinated. Brucker *et. al* [29] studied several variations of the scheduling problem on batch processors. A batch-processor is one that can accommodate up to *b* jobs simultaneously, where the jobs that are processed together form a batch, and all jobs in a batch start at the same time. Note that one can represent a set of closely spaced parallel runways as a batch-processor that can accommodate up to b jobs (aircraft) whose scheduled arrival times need to be coordinated. Here, b represents the number of closely spaced parallel runways. Given the above background, note that a variant of the PSP problem, formulated in section III, that tries to maximize the makespan, can be represented using Graham's notation as follows: $(1|\ b = 2\ ;\ r_i\ ;$ p-batch $|\ C_{max})$. In this variant, $b = 2$ represents the existence of two runways, r_i is the release time for job i (earliest time it can be scheduled), and C_{max} represents the maximum completion time among all scheduled jobs. The term p-batch stands for *parallel batch* indicating that jobs can be processed in parallel. Additional characteristics of this variant include:

i) No precedence or sequencing constraints, indicating overtaking is allowed on the arrival streams.

ii) No deadlines for the scheduling of the aircraft,

iii) The same ETA at the left, right runways, for all aircraft,

iv) A pairing window of zero ($t_p = 0$),

Brucker *et. al* [29] studied several variations of the scheduling for batch processors. They showed:

Theorem 1. [29] $(1\ |\ 2 \leq b < n;\ r_i;$ p-batch $|\ C_{max})$, where n is the number of jobs in the problem instance, is $\mathcal{NP}$–hard in the strong sense.

As indicated, the problem $1|\ b = 2;\ r_i;$ p-batch $|\ C_{max}$ is a special case of PSP, with the minimization of makespan as the objective. Similarly, observe that the problem $(1|\ 2 < b < n;\ r_i;$ p-batch $|\ C_{max})$ is a special case of b–GSP, for $2 < b < n$, where n is the number of aircraft in the problem instance. Thus, using $\mathcal{NP}$–hardness proof by restriction [28], we have the following result:

Theorem 2. With the objective defined as minimizing the makespan, the problems PSP and b–GSP with $2 < b < n$, are both $\mathcal{NP}$–hard, in the strong sense.

Brucker *et. al* [29] also used reduction from 3-PARTITION [29] to show that when jobs have deadlines (d_i), deciding whether all jobs can be feasibly scheduled, is $\mathcal{NP}$–complete for variants of the problem with $2 \leq b < n$. More specifically, they showed:

Theorem 3. [29] Given an instance of the problem $(1\ |\ 2 \leq b < n\ ;\ r_i\ ;\ d_i\ ;$ p-batch$|)$, it is $\mathcal{NP}$–complete in the strong sense to decide whether there is a solution in which all jobs can be feasibly scheduled.

Again, by observing that the above are special cases of PSP and GSP, we have the following:

Theorem 4. Given a PSP or a b–GSP instance, with $2 < b < n$ instance, it is $\mathcal{NP}$–complete in the strong sense, to decide whether there is a scheduling solution in which all aircraft can be feasibly scheduled.

The fact that special cases of PSP belongs to the strongly $\mathcal{NP}$–hard and $\mathcal{NP}$–complete class of problems, means that unless $\mathcal{P}=\mathcal{NP}$, there is no efficient algorithm that can solve an arbitrary instance of the problem optimally. On the other hand, many practical instances of the problem may not be very difficult to solve. For one thing, from *Theorem 1*, note that the hard instances of the problem include those with no deadline and they allow overtaking among the aircraft in the arrival streams. In practice, however, the arriving aircraft have strict deadlines by which they have to be scheduled for landing, and in general overtaking is not allowed in the arrivals streams. These additional constraints reduce the feasible solution space, suggesting that it might be more manageable to look through the feasible solution space for the optimal (or close to optimal) scheduling solutions. In order to do this, however, we need efficient schemes for effective navigation and exploration of the feasible solution space. In the remainder of this paper, we discuss the development of an algorithm for solving this problem and demonstrate its performance characteristics in practice.

V. DEVELOPING THE PAIRING ALGORITHM

The emergence of different requirements and constraints during the early phases of this work, suggested that the best course of action would be to seek algorithmic solutions that are practical, adaptable, and capable of producing feasible scheduling results efficiently. The task was compounded by the fact that the set of requirements and constraints defining feasible scheduling solutions were in continual state of flux and revision during the earlier phases of the research. This required a flexible solution strategy that could be adjusted as the requirements and constraints were being defined. At the same time, a prototype was needed that could be developed in parallel and integrated into the system for conducting real-time HITL simulations that were planned [7].

The availability of Kupfer's prototype [26] and its implementation based on the GA paradigm [27] provided a starting point, suggesting GA as a suitable development framework. As is typical in GAs, our algorithm consists of three main phases. In the first phase, an initial population of feasible (scheduling) solutions is created. The second phase forms the evolutionary optimization phase and is comprised of a (fixed) number of iterations. During each iteration, current population of feasible solutions undergoes one generation of evolutionary optimization followed by a greedy packing heuristic. In the final phase, the best solution encountered so far is reported. Although both our solution and Kupfer's lie within the GA framework, there were significant differences in terms of problem formulation, scalability and expected solution quality suitable for real-time HITL simulations. Simple extensions to Kupfer's algorithm proved quite inadequate, as explained below, forcing us to develop the main components of the GA from scratch.

A. Implementation of the Objective Function

The objective function is used to determine the relative merit of a given scheduling solution, and operates by associating a scalar merit value to each candidate scheduling solution. Due to the significant differences between our problem model and that of [26], and the existence of various soft and hard requirements and constraints, we decided to redesign the objective function from scratch, with all the requirements and constraints in mind. The revised objective function associated a merit value to each scheduling solution evaluated during the course of the optimization, and the GA was setup to find the scheduling solution of maximum merit value. For a given scheduling solution, the following merit function was used:

$$merit = a_0 + a_1 P + a_2 S + a_3 STA_{max}$$

where P is the *effective number* of scheduled aircraft pairs, S is the number of scheduled aircraft singles, STA_{max} is the largest STA among the scheduled aircraft, and a_0, a_1, a_2, a_3, are parameters used to shift and scale the merit factors appropriately. Among these, a_0 was chosen to shift the merit values to a meaningful range, while a_1, a_2, a_3 were selected in order to provide the relative significance of the other factors. The following values were used for these parameters to run the experiments:

$$a_0 = 10{,}000; \quad a_1 = 100; \quad a_2 = 10; \quad a_3 = -0.05$$

B. *Implementing Solution Space Control and Preferences*

An important feedback received during initial pilot study of the system was the need to allow manual override by the air traffic controllers on the pairing solution. Throughout, the controllers had the option not to accept a suggested pair. However, there was a need to communicate an undesirable pair to the pairing algorithm so that the same undesirable pair would not be suggested repeatedly. This was implemented by maintaining a list of *forbidden pairs*. A forbidden pair is one with designated leader and follower, which the pairing algorithm should not suggest. Once the controllers identify a pair as forbidden, the pairing algorithm would add it to its list of forbidden pairs. To implement this during the optimization, the forbidden pairs list was consulted during the creation of the initial population seed, during the course of the optimization, and once again before reporting the pairing results. In order to gradually lead the exploration out of the undesirable portion of the search space, the objective function was used to discourage forbidden pairs from appearing in a solution by giving such pairs a much lower weight. To ensure no forbidden pairs would leak into the final pairing solution, a pruning of the pairing solution was done just before reporting the pairing result to the controllers.

To allow smooth exploration of the search space, the contribution of such pairs to P were reduced to guide the algorithm to look for scheduling solutions not involving undesirable or forbidden pairs. Some of the undesirable pairs were those involving aircraft that would require a change in their assigned runways. For example, it was desirable not to pair aircraft that are by default destined to the same runway, when a more or less equivalent choice seemed to exist. As an example, it would be generally undesirable to pair aircraft approaching from the eastern arrival streams **Modesto** and **Yosem**. Each such pair would involve a change in the runway assignment, requiring manual intervention and coordination that would lead to an increase in controller and pilot workload. Other factors contributing to the controllers identifying a pair as forbidden were somewhat subjective and based on developing circumstances that were not quite amenable to exact qualification for the pairing algorithm to be able to avoid them altogether, especially in the early phases of the study, when the need to implementing forbidden pairs was envisaged.

In order to implement undesirable or forbidden pairs, the contribution of such undesirable pairs to P was reduced to 0.8 compared to the contribution of 1.0 for regular pairs. For the forbidden pairs, the contribution was significantly reduced to 0.1 making the impact of such pairs relatively insignificant.

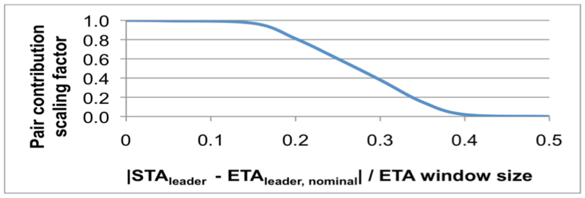

Fig. 5. Cooling function to lower a pair's contribution to the number of pairs, as the leading aircraft in the pair is scheduled farther from its nominal ETA.

This would cause the algorithm to look harder for scheduling solutions that do not involve forbidden pairs. It was decided not to assign a negative contribution to forbidden pairs, as this would interfere with other cost factors and cause potential instability during the search process. As a final assurance, all remaining forbidden pairs were always filtered out of the pairing solution before reporting the list of aircraft pairs to the controllers.

Another soft preference was for the algorithm to schedule aircraft close to their nominal ETA, and thus to look for scheduling solutions that would disturb the arrival stream as little as possible, leading to increased savings in fuel consumption, more predictable arrival times, and more accurate planning of arrival and departure times. Since there is coordinated arrival of two aircraft in each pair, this issue becomes much more pronounced for the leading aircraft in a pair. To accommodate this preference into the objective function, a scaling factor was introduced to control the contribution of a pair to the total number of pairs (P) that are used for the merit function in evaluating a scheduling solution. A pair's contribution is lowered based on a cooling function as its leading aircraft is scheduled farther from its nominal ETA. An example of the cooling schedule is shown in Fig. 5, where the horizontal axis represents how far the STA of the leading aircraft is from its nominal ETA, as a fraction of its overall ETA window size, and the vertical axis shows the contribution of such a pair to P.

C. *The GA Initialization Algorithm*

An effective pairing scheduler is one that successfully finds a large number of aircraft pairs. During the early phases of our research, however, it was observed that the initial prototype, which was an extension of Kupfer's algorithm, had considerable difficulty finding aircraft pairs, and consistently produced results of inferior quality. Careful examination suggested that a major contributor to the poor solution quality was the algorithm used for producing the initial population pool for the GA. More specifically, two issues were identified: a) the initial population was obtained by repeated calls to a deterministic scheduling algorithm, b) the algorithm did not explicitly seek to find solutions containing aircraft pairs; instead, it settled for any feasible solution. As a result, the initial population had no diversity, and was often quite suboptimal as it contained very few or no aircraft pairs. This placed much limitation on the ability of the GA to escape out of local minima in search of solutions of superior quality.

To address these issues a new GA initialization algorithm was developed. The new algorithm was designed to be non-deterministic, producing different feasible solutions on consecutive calls. Moreover, the algorithm was explicitly designed to maximize the number of feasibly scheduled aircraft pairs. The proposed algorithm is a non-deterministic greedy algorithm that uses First-Come-First-Served (FCFS) heuristic; hence it is called *NDGreedyFCFS*. Its pseudo code is provided in **Algorithm 1.**

The algorithm consists of three phases. In the initialization phase (steps 1–15), an empty schedule is created and the algorithm proceeds with creating arrival stream queues, with each queue containing the list of aircraft arriving from the corresponding arrival stream. Each queue is sorted in the increasing value of the nominal ETA of the aircraft. The algorithm then forms (and maintains) ***candSet***, which constitutes the set of unscheduled aircraft from which the next aircraft to be scheduled is to be picked. Since overtaking is not allowed, at the beginning, this set is formed from the first arriving aircraft from each arrival stream (steps 11–15).

After the initialization phase, the algorithm goes through a loop forming the iterative phase of the algorithm (steps 15–30). During each iteration ***candSet*** is pruned and updated by removing aircraft that cannot be feasibly scheduled given the partial schedule ***S*** that is formed thus far. Each pruned aircraft is then replaced by the next arriving aircraft from the same arrival stream (steps 17–22). Once the ***candSet*** is pruned and updated, the algorithm proceeds by greedily finding the best candidate aircraft (*opt*) among those in ***candSet*** (step 23). This is the earliest aircraft that can be legally scheduled among all the aircraft in ***candSet***, given the partial schedule ***S*** that is formed so far. Once this aircraft is identified, it is assigned a runway, based on the runway assignment rules and scheduled (steps 25–27). Non-determinism is introduced in step 26, where the best candidate identified (*opt*) is scheduled randomly in its legal range, thus diversifying the seed population. This diversity is then used for more effective exploration of the solution space during genetic optimization. Prior to continuing to the next iteration of the loop, ***candSet*** is amended by removal of the aircraft that has just been scheduled (step 24), and addition of the next aircraft arriving from the same arrival stream as the one that has just been scheduled (steps 28, 29).

Algorithm 1. *NDGreedyFCFS* (***A***, ***W***, t_p, ***R***)

Input:	***A***:	The set of aircraft in the problem instance.
	W:	The wake separation matrix
	t_p:	The pairing window.
	R:	The runway assignment rules.
Output:	***S***:	The scheduling result.

```
1.  S = ∅;
2.  for each arrival stream i do
3.      Q[i] = ∅;
4.  end for
5.  for each a ∈ A do
6.      Q[arrivalStream (a)] = Q[arrivalStream (a)] ∪ {a};
7.  end for
8.  for each arrival stream i do
9.      Sort Q[i] in increasing order of the nominal ETA of the aircrafts;
10. end for
11. candSet = ∅;
12. for each arrival stream i do
13.  a = Pop Q[i].head;
14.     candSet = candSet ∪ a;
15. end for
16. while (candSet ≠ ∅) do
17.     for each aircraft a ∈ candSet do
18.         if (a cannot be legally scheduled given partial schedule S)
19.             candSet = candSet – {a};
20.             candSet = candSet ∪ Q[arrivalStream(a)].head;
21.         end if
22.     end for
23.     opt = the aircraft in candSet that can be legally scheduled
        earliest, given the partial schedule S;
24.     candSet = candSet – {opt};
25.     Assign runway (opt) based on the rules in R;
26.     Assign STA (opt) randomly in its legal range;
27.     S = S ∪ {opt};
28.     next = pop the next aircraft in Q[arrivalStream(opt)];
29.     candSet = candSet ∪ {next};
30. end while
31. return S;
```

Note that if an aircraft cannot be legally scheduled, given the partial schedule that is constructed so far, due to the constructive and greedy nature of the algorithm, there is no need to consider it in the future. Therefore, we can safely eliminate these aircraft from consideration. The following theorem characterizes the time complexity of the algorithm:

Theorem 3. The algorithm *NDGreedyFCFS* can be implemented to run in $O(n \times \max\{m, \log n\})$ time, where n and m are the number of aircraft and the number of arrival streams in the problem instance, respectively.

Proof: Note that in the initialization phase, each aircraft is processed exactly once, resulting in $O(n)$ run time, while creating sorted arrival stream queues ***Q*** may take $O(n \log n)$. The iterative phase, on the other hand, runs in $O(nm)$ time, since during each iteration, one more aircraft is scheduled, or at least one aircraft is eliminated from future consideration, and the amount of time required to complete each iteration is proportional the number of elements in ***candSet***. Since there can be at most one aircraft from each arrival stream in ***candSet***, each iteration of the while loop can be implemented in $O(m)$ time, with the while loop taking $O(nm)$ overall. The total runtime of the algorithm is therefore $O(n(m + \log n))$, or equivalently $O(n \times \max\{m, \log n\})$ □

The GA used for solving PSP uses *NDGreedyFCFS* in order to seed its initial solution population. An initial population size of 20 scheduling solutions was used to seed the GA. After generating the pool of initial solutions, the algorithm goes through N_{GA} iterations of genetic evolution, during each of which the population pool is evolved and then undergoes a greedy packing step to further enhance each individual solution. The best individual solution, as determined by the objective function, is then returned.

The way the pairing algorithm is used in the HITL experiments is as follows: The automation system scans the airspace for the set of aircraft within the pairing zone that are not yet scheduled, and sends the list of such aircraft to the pairing algorithm, which then returns the proposed scheduling solution and pairs back to the automation system. The proposed solution is then shared with the controllers who will make the final determination as to which pairs are accepted. The aircraft whose proposed schedules are accepted, are then removed from future consideration. This whole processing cycle repeats every few minutes as new aircraft enter the pairing zone.

VI. Experimental Results

The pairing algorithm described earlier was implemented in C/C++ using GALib [30] and integrated into the ground air traffic control system for HITL simulation. A stand-alone version was also implemented to conduct off-line performance and stress testing. The stand-alone tests were run on a Mac Pro machine with 2 × 2.8 GHz Quad-Core Intel Xeon processor. This section presents the results of experiments conducted to measure performance, scalability, and quality of the pairing results produced by the algorithm. Additional experiments were conducted to illustrate the flexibility of the algorithm and its objective function in targeting different quality metric to be optimized.

A. Performance Results

The first set of results shown in Table 3, summarize the impact of the (improved) GA initialization, compared to the initial implementation (original) on the number of pairs that were discovered. The results are shown for both the *HITL test scenarios* used in [7] as well as a suite of *stress test scenarios*. Note that the HITL test scenarios were designed to exercise the system and provide coverage for the intended operational procedures to study the proper allocation of tasks between the automation and the controllers [7]. They were not developed to stress the pairing algorithm sufficiently. In order to conduct a more detailed evaluation of the pairing algorithm, a suite of stress test scenarios was generated using a randomized parametric algorithm. The parameters included the numbers of arrival streams, aircraft, pair groups, as well as the minimum and maximum spacing requirements between the arrival times of consecutive aircraft on the same arrival stream.

In Table 3, each row in the HITL tests section corresponds to a specific scenario, while each row in the stress test section corresponds to over two hundred different scenarios, each with the same number of aircraft, for which the average runtime and the average number of discovered pairs are reported. Results are presented for the pairing using the original and the improved GA initialization algorithm, showing significantly more number of aircraft pairs for the latter, while the run-time stayed well below a minute, even for scenarios containing as many as 100 aircraft.

To see the overall effectiveness of the pairing algorithm, we compared the number of feasibly scheduled single, and paired aircraft per run in the HITL simulations in two different modes using the same exact scenarios: once with the pairing algorithm, and once without it, where aircraft pairs were identified and established completely by the controllers. Fig. 6 shows the result, indicating a considerable increase in runway throughout, when the pairing algorithm was used.

B. Efficiency and Scalability Results

A pairing algorithm intended for real-time applications must be efficient and scalable. This allows its use on instances of practical size, and results in response times that are acceptable for real-time applications. However, an efficient and scalable algorithm that produces results of poor or questionable quality is of little interest. So, striking the right balance between the quality of results and the efficiency of the pairing algorithm is of great importance.

A reasonable measure for the quality of results is the number of aircraft that were successfully scheduled. In order to find the right balance between solution quality and runtime, experiments were conducted to determine the number of iterations in the evolutionary phase of the algorithm. This

TABLE 3
IMPACT OF GA INITIALIZATION ALGORITHM

	#	Original		Improved	
	A/C	#Pairs	Runtime(s)	#Pairs	Runtime(s)
HITL Test Scenarios	11	0	1	4	2
	12	1	3	5	3
	14	5	4	7	4
	16	0	2	6	3
	18	0	2	5	2
	20	0	3	6	3
	34	0	6	7	5
Total		**6**	**21**	**40**	**22**
Stress Test Scenario	20	0.02	2.16	4.30	4.64
	40	0.00	4.26	11.76	10.66
	60	0.00	6.64	18.75	18.28
	80	0.00	9.04	25.02	26.70
	100	0.00	11.50	32.74	35.54
Total		**0.02**	**33.60**	**92.57**	**95.82**

number was varied while the percentage of feasibly scheduled single and paired aircraft were measured as the algorithm was run on the stress test scenarios. It was observed that iterating the genetic evolutionary phase for more than 1,000 times did not result in any noticeable increase in the percentage of successfully scheduled aircraft. Therefore, the number of generations for the genetic evolution was conservatively set to 10,000. Having set the number of generations in the genetic evolutionary phase, the runtime scalability of the overall algorithm was tested using the stress test scenarios. Fig. 7 and 8 show the percentage of aircraft successfully scheduled for landing, and the runtime of the pairing algorithm, respectively, as a function of the number of aircraft in the scenario. In Fig. 7, the percentage of aircraft successfully scheduled is further broken down into its paired-landing and single-landing constituents. As it is shown, the percentage of the total aircraft successfully scheduled for landing stays well above 95% even for problem instances containing up to one hundred aircraft. At the same time, the percentage of the aircraft that are scheduled for paired landing comprises a significant portion of the total, ranging from 30% to more than 60% of all the aircraft. This suggests that the algorithm performs in a scalable manner, with no sign of solution quality degradation, as the number of aircraft in the problem instance approaches

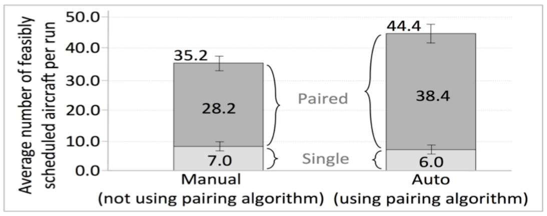

Fig. 6. Overall effectiveness of the pairing algorithm.

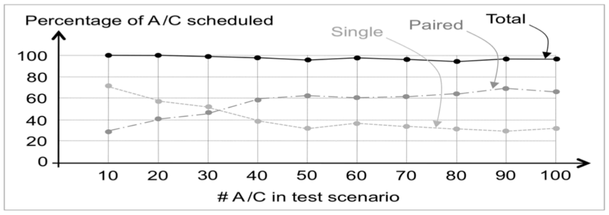

Fig. 7. Percentage of scheduled aircraft in HITL simulation runs.

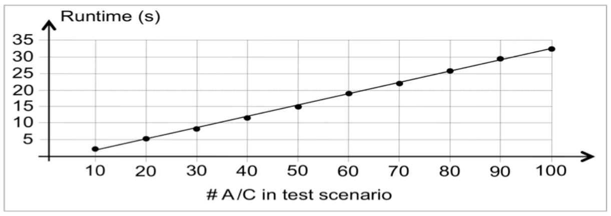

Fig. 8. Runtime profile of the pairing algorithm.

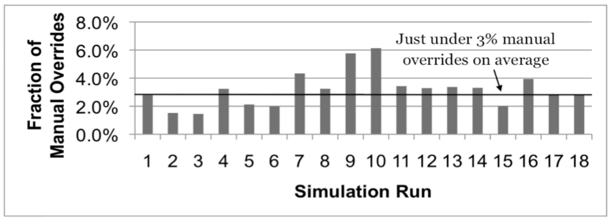

Fig. 9. Percentage of manual overrides in the HITL simulation runs.

one hundred or more. The average runtime of the pairing algorithm for the stress test scenarios as a function of the number of aircraft in the problem instance is plotted in Fig. 8.

Note that a 100-aircraft instance of the problem required less than 40 seconds to solve, on average, an acceptable response time, given that all scenarios in the HITL simulation runs contained no more than 45 aircraft, requiring no more than 15 seconds to solve, on average. The number of aircraft in the HITL scenarios was in the range 13–41, averaging around 25. Since pairing occurs approximately 30 minutes prior to aircraft reaching the terminal area boundary. A pairing algorithm with less than half a minute of turn-around time proved quite adequate in the HITL simulations. In most instances, the number of aircraft was 35 or less and the pairing result was available in less than 10 seconds.

C. *Quality of the Scheduling Results*

The final judgment about the suitability of an aircraft pair for simultaneous landing is the responsibility of the human operators. In order to measure the quality of pairs suggested by the pairing algorithm, the HITL simulation results were examined to see how often the forbidden pair feature was exercised. Fig. 9 summarizes the results.

The data collected during HITL simulation consisted of 18 runs, representing more than ten hour of simulation, indicating that controllers accepted over 97% of the pairs suggested by the algorithm. During the course of development of the pairing algorithm, and as the algorithm and the set of constraints it could handle matured, the need to exercise forbidden pair feature arose progressively less often. Note that in the final implementation of the algorithm, by definition, all the pairs suggested by the algorithm were "good pairs", meaning they met all the problem constraints. However, real-time and asynchronous nature of HITL simulation resulted in situations where not all the suggested pairs were evaluated and/or accepted immediately or in a timely fashion. Occasionally, pairs that once looked appropriate became obsolete, as other pairs were made or accepted, thus prompting the human operators to mark some originally suggested pairs as forbidden. Other contributing factors included controller overload and changing circumstances from the time the original suggestions were made until the time the controllers got the chance to evaluate the pair. Overall, the team was pleased with the quality of the pairing suggestions in the final implementation, and low level invocation of forbidden pair feature, thus confirming the effectiveness of the objective function in guiding the search process, and the ability of the algorithm in handling all constraints and preferences successfully.

D. *Smooth Tradeoff between Runtime and Solution Quality*

The next sets of results illustrate how the scheduling solution quality evolves during the course of the algorithm, as it goes through more iterations of genetic evolution. It is desirable for the algorithm to provide solutions of acceptable quality very quickly within a few dozen genetic iterations. Additional iterations would be applied as needed while maintaining the runtime within acceptable levels. To illustrate how the characteristics of the scheduling solution evolved during the course of the GA, the pairing algorithm was exercised on all arrival scenarios with a fixed number n of aircraft and snap shots of the best scheduling solutions were taken after 1, 10, 100, 1000, and 10,000 iterations of the GA. This is equivalent to running the pairing algorithm 5 times on each instance, with N_{GA} set to 1, 10, 100, 1000, and 10,000. The key quantities that were recorded and reported for different values of n and N_{GA} are as follows:

- Percentage of feasibly scheduled aircraft, whether or not they were paired. That is, $100 * (2P + S) / n$, where P and S represent the number of feasibly scheduled aircraft pairs and singles, respectively (see Fig. 10),
- Percentage of the aircraft that were feasibly scheduled as pairs. That is, $100 * (n_P / n)$, where $n_P = 2P$ represents the number of feasibly scheduled aircraft that were paired. It is desirable to maximize this quantity (see Fig. 11),
- The normalized makespan of the scheduling result, which is the makespan divided by total number of feasibly scheduled aircraft: $(sta_{max} - sta_{min}) / (2P + S)$. Note that between two scheduling solutions with the same number of scheduled aircraft, the one with smaller makespan is preferred (see Fig. 12),
- A measure of how far, on average, the lead aircraft in a pair was scheduled from its nominal ETA. To normalize this into a meaningful quantity, this average was divided by the width of the ETA window. In our study, this was of secondary importance, intended to find solutions disturbing the nominal arrival times as little as possible (see Fig. 13),
- The overall runtime of the algorithm (see Fig. 14.)

In order to illustrate how the scheduling results evolved over time, the above experiment was conducted on arrival scenarios from both the HITL simulation runs as well as those generated to conduct stress tests. Arrival scenarios from the HITL simulation runs with n aircraft in the scenarios, $n \in \{15, 20, 25, 30, 35, 40\}$ were run, and the metrics listed above were recorded and averaged for all the test cases with the same number of aircraft. Fig. 10 through 14 illustrate how the solution quality evolved over time as the GA executed more evolutionary iterations. Note that within roughly 100 generations the algorithm found scheduling solutions in which most of the aircraft were feasibly scheduled. The remaining portion of the evolutionary process was mainly used to inch up the number of feasibly scheduled aircraft, and to minimize the makespan and how far from the nominal ETA the leading aircraft in a pair was scheduled.

Note that the number of pairs in the solution occasionally decreased in order to find solutions containing more feasibly scheduled aircraft. This is because, although the number of feasibly scheduled aircraft pairs plays a significant role, the number of feasibly scheduled single aircraft also carries some weight. Besides, some pairs do not carry as much weight as others, based on the soft pairing rules explained earlier.

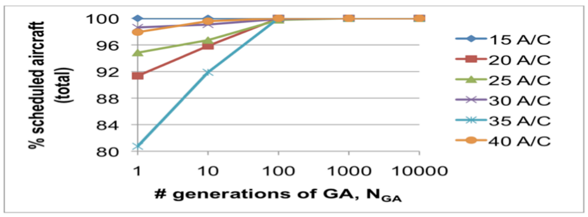

Fig. 10. Percentage of the aircraft that were feasibly scheduled, on average, as a function of the number of generations of the genetic algorithm in the arrival scenarios from the HITL simulations.

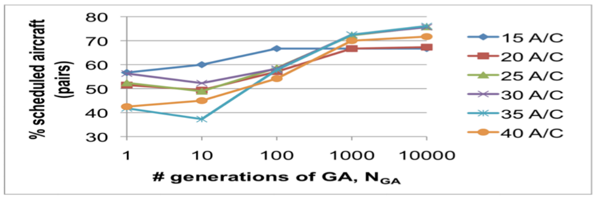

Fig. 11. Percentage of the aircraft that were feasibly scheduled and paired, on average, as a function of the number of generations of the genetic algorithm in the arrival scenarios from the HITL simulations.

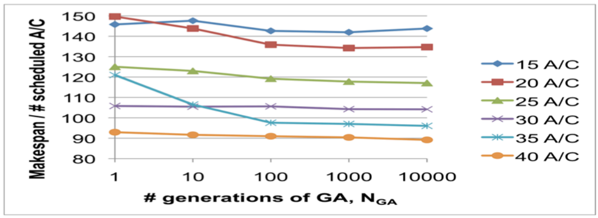

Fig. 12. Makespan divided by the number of feasibly scheduled aircraft, as a function of the number of generations of the genetic algorithm in the HITL simulation runs.

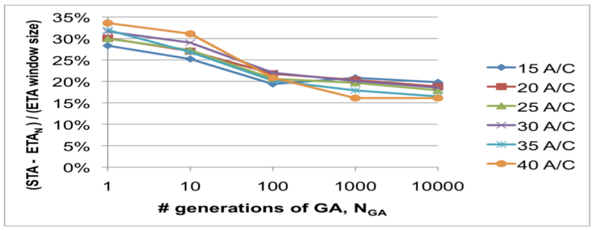

Fig. 13. How far, on average, the leading aircraft in each pair was scheduled from its nominal ETA, as a function of the number of generations of the genetic algorithm in the HITL simulation runs.

Similar experiments were conducted for stress test scenarios with n aircraft per scenario, where $n \in \{20, 40, 60, 80, 100\}$. The results of these experiments are qualitatively very similar to those conducted for the HITL scenarios illustrated in Fig. 10 through 14, and hence not shown. Note that each curve in these figures represents the average value of the measured quantity over all the scenarios with the same number of aircraft.

As it can be seen, even with no evolutionary optimization, the GA initialization algorithm has was able to find scheduling solutions with more than 70 percent of the aircraft scheduled, out of which roughly half of them (40 to 60 percent of all the aircraft) were paired. With roughly 100 generations of genetic evolution, spending only a fraction of a second, the algorithm found solutions with most of the aircraft feasibly scheduled. The remaining 9,900 generations of the evolutionary process looked further for scheduling solutions that minimized the normalized makespan and those in which the aircraft were scheduled as close as possible to their nominal ETA. These results demonstrate that the pairing algorithm exhibits desirable characteristic that would allow one to smoothly trade-off runtime to find higher quality scheduling solutions.

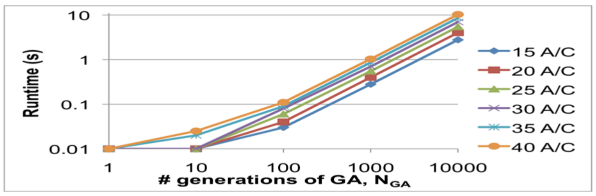

Fig. 14. Runtime of the pairing scheduler as a function of the number of generations of the genetic algorithm in the HITL simulation runs.

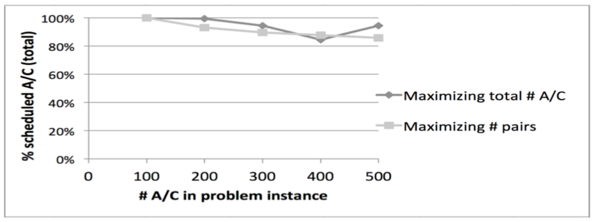

Fig. 15. Comparison of the total number of aircraft scheduled with the default objective function as well as an objective function that tries to maximize the number of scheduled pairs only.

E. *Flexibility for Targeting Different Objectives*

In this section we illustrate flexibility of the algorithm for targeting different objectives. Experiments were conducted on arrival scenarios containing from 100 to 500 aircraft each, to show the flexibility of the algorithm across a wide range of problem sizes, while limiting the runtime of the algorithm to 30 seconds. Scalability of the algorithm is also demonstrated on problem instances containing up to 500 aircraft. For this purpose, additional stress test scenarios with up to 500 aircraft were generated and ran through the algorithm, while limiting the runtime to 30 seconds. Total numbers of feasibly scheduled aircraft, as well as total number of feasibly scheduled aircraft pairs were recorded. The same experiment was repeated using the following merit function to maximize the number of aircraft pairs (ignoring the number of single aircraft):

$$merit = a_0 + a_1 . P$$

Results of these experiments for the two merit functions are shown in Fig. 15 and 16. Note that even for scenarios with up to 500 aircraft, the algorithm found scheduling solutions in which more than 80% of the aircraft were feasibly scheduled, and more than 45% of the aircraft were paired. With the objective function set to maximize the number of pairs, the algorithm found scheduling solutions in which from 5% to 20% more aircraft were paired, while the total number of feasibly scheduled aircraft decreased by an average of about 5%.

Another flexible feature of the algorithm stems from the ability to control the runtime while still getting a reasonably accurate estimate of the number of aircraft pairs and singles that can be feasibly scheduled given a specific arrival scenario. This means that the algorithm can serve as a good prediction and planning tool to test what-if scenarios for early flight planning, if needed. To illustrate this feature, the algorithm was tested on arrival scenarios with 100 to 500 aircraft, letting it to run for over one minute. Two snapshots were taken of the best scheduling solution found, one at t_1 = 0.1 seconds and once at t_2 = 60 seconds into the run. Let $n(t_1)$ and $n(t_2)$ represent the number of feasibly scheduled aircraft in the best solution found after running the algorithm for t_1 = 1 second and t_2 = 60 seconds, respectively. Fig. 17 shows the plot for

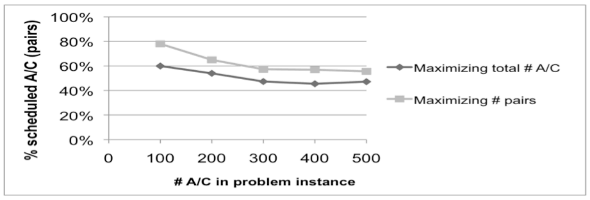

Fig. 16. Comparison of the number of aircrafts that were paired with the default objective function as well as an objective function that tries to maximize the number of scheduled pairs only.

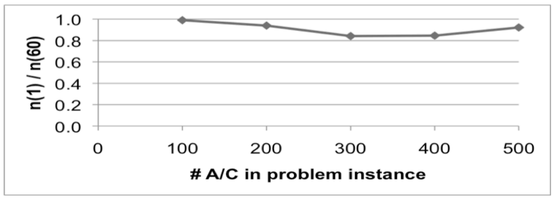

Fig. 17. Ratio of the number of aircraft that could be feasibly scheduled in the best scheduling solution found after 1.0 second and one minutes into the pairing algorithm.

the ratio $n(t_1)$ / $n(t_2)$ on average, as the number of aircraft per scenario goes from 100 to 500. Note that even for problem instances containing as many as 500 aircraft, the algorithm can make a reasonably accurate prediction on the number of aircraft that could be feasibly scheduled, spending no more than one second. For smaller problem instances, the accuracy of such a prediction after 1.0 second is even better. This means the algorithm can serve as a fast prediction tool to estimate the total number of aircraft that could be feasibly scheduled, a feature that could be exploited for planning and traffic flow management purposes.

VII. Discussion

Note that the focus in this paper has been on the formulation of the pair scheduling problem, its computational complexity, the development of the pairing algorithm, and analyzing its performance characteristics. The functional allocation of the roles between the human controllers and the automation for simultaneous approach is detailed in [7], as is the detailed analysis of the HITL simulation data. The material in [7] also addresses some of the important human factors issues that are not discussed in this paper, and thus complements the discussions presented here.

Some generalizations of the problem that are of interest for future research include extensions to more than two parallel runways, and inclusion of additional constraints and factors in objective function such as balancing the traffic on the runways, as well as simultaneous consideration of arrival and departure traffic. In addition, conducting a comparative study using different scheduling algorithms, for example those reported in [22], might provide valuable data on the relative strengths and weaknesses in terms of runtime, flexibility of the algorithms, and the quality of the scheduling results.

The scope of the current study was limited in several respects. For instance, no accommodations were provided for vectoring or re-routing aircraft that could not be legally scheduled for landing. Also, in practice, the early and late ETA at the coupling points would be a function of several factors including the aircraft capabilities, speed profiles, wind conditions, and how early the aircraft is available for pairing consideration. In the current study, these issues were not considered and a simple range was used around the nominal ETA at the coupling point to continue with the proof of concept study. Additionally, one could argue that the earlier the arriving aircraft are available for pairing, the higher is the chance of finding scheduling solutions that maximize the runway utilization, while this would increase the uncertainty involved in meeting the prescribed pairings and the associated schedule. Thus, robustness of the overall system to the variations in the ETA times is another important question that was left for potential future studies, as is the impact of earlier scheduling and pairing solution on the pairing potential of aircraft much later in the arrival sequence. Conducting research into paired arrival and departure scheduling has other practical long-term applications. The insights gained from these studies could help extend the capabilities of new and improved tools for terminal area scheduling.

Considerations for handling off-nominal conditions and accommodating the aircraft that may not be feasibly scheduled by the pairing algorithm are important questions that are left for future studies. Potential extensions that might be needed to accommodate these aircraft include directing these aircraft out of the arrival stream, maintaining some empty slots in the arrival sequence to allow vectoring these aircraft back into the arrival stream in such a way that the pairing scheduler can consider them for paired or single landing.

VIII. Conclusion

The problem of scheduling aircraft for simultaneous landing onto very closely spaced parallel runways is studied. The contributions of this work include: *i*) presenting a general formulation for the Pair-Scheduling and Group-Scheduling problems, *ii*) demonstrating that the objective function in these problem do not behave monotonically, in general, *iii*) showing that these problems, in general, are $\mathcal{NP}$–hard in the strong sense, *vi*) developing a scalable pairing scheduler using GA development framework that could accommodate a large variety of practical constraints and preferences, suggesting its suitability for real-time HITL simulations.

The algorithms were implemented and integrated into the ground air traffic control for HITL simulations [7]. In addition to the scheduling scenarios that were part of the HITL simulations, an extensive set of stress test scenarios were developed to exercise the pairing algorithm more thoroughly. Experimental results indicate:

- The runtime of the proposed algorithm scales almost linearly with the number of aircraft in the problem instance, resulting in an acceptable response time, thus making it suitable for real-time application.
- The proposed pairing algorithm succeeded in finding significantly more pairs, as compared to an algorithm developed using simple extensions of an earlier prototype.
- Despite the short runtime, the algorithm produced pairing results of superior quality. Analysis of the HITL simulation results indicated that over 97% of the pairs suggested by the algorithm were acceptable to the participating air traffic controllers.
- By controlling the number of generations of evolutionary optimization in the GA, we can achieve an algorithm that provides a smooth tradeoff between runtime and the quality of

the scheduling solution. Even with about 100 generations of the evolutionary optimization, we achieve scheduling solutions with more than 95% of the aircraft feasibly scheduled, with about half of the aircraft paired. Additional iterations improve the scheduling solution by minimizing the makespan and scheduling the aircraft close to their nominal ETA.

- The algorithm can be used as a fast prediction tool to provide quick estimates of the number of aircraft that could be feasibly scheduled, allowing the algorithm to be used as a tool for early planning and evaluation of different arrival scenarios in the larger air traffic management context.

References

[1] A. H. Farrahi, S. A. Verma, "A Pairing Algorithm for Landing Aircraft to Closely Spaced Parallel Runways," Proceedings of the 29th IEEE/AIAA Digital Avionics Systems Conference, Salt Lake City, UT, pp. 4.B.4-1–4.B.4-15, October 3–7, 2010.

[2] C. L. Scovel, III, "Actions Needed to Meet Expectations for the Next Generation Air Transportation System in the Mid-Term", Testimony before the Committee on Transportation and Infrastructure Subcommittee on Aviation, US House of Representatives, October 28, 2009.

[3] Federal Aviation Administration, "FAA's NextGen Implementation Plan", March 2011, Available from: http://www.faa.gov/

[4] R. Bone, A. Mundra, B.O. Olmos, 2001, "Paired Approach Operational Concepts," *Proc. 2001 Digital Avionics Systems Conference*, Daytona Beach, FL, October 2001, pp. 5B3/1–5B3/14, Vol. 1.

[5] T. Abbott, D. Elliot, "Simulator Evaluation of Airborne Information for lateral Spacing (AILS) Concept," NASA/TP-2001-210665.

[6] M. E. Miller, S. Dougherty, J. Stella, P. Reddy, "CNS Requirements for Precision Flight in Advanced Terminal Airspace," *Proc. 2005 EEE Aerospace Conference*, Big Sky, MT, pp. 1–10.

[7] S. Verma, T. Kozon, D. Ballinger, A. Farrahi, "Functional Allocation of Roles Between Humans and Automation for a Pairing Tool Used for Simultaneous Approaches", *International J. of Aviation Technology*, Vol. 23, No. 4, pp. 335–367., October 2013.

[8] G. Kulesa, "Weather and Aviation: How Does Weather Affect the Safety and Operations of Airports and Aviation, and How Does FAA Work to Manage Weather-Related Effects?" *The Potential Impacts of Climate Change on Transportation Research Workshop*, U.S. Dept. of Transportation, Center for Climate Change, October 2002, pp. 199–208.

[9] U.S. Department of Transportation, Federal Aviation Administration, Order 8260.49A, "Simultaneous Offset Instrument Approach (SOIA)", June 2006, Available at http://www.faa.gov/

[10] A. Pritchett, "Pilot Performance at Collision Avoidance During Closely Spaced Parallel Approaches," ATC Quarterly, Vol. 7, No. 1, 1999, pp. 47–75.

[11] B.E. Barmore, T.S. Abbott, W.R. Capron, B.T. Baxley, "Simulation Results for Airborne Precision Spacing Along with Continuous Descent Arrivals", 8th AIAA Aviation technology and Operation (ATIO) Conference, September 2008, Anchorage, AK.

[12] P. Brucker, *Scheduling Algorithms*, Fifth Edition, Berlin, Germany, Springer-Verlag.

[13] R. L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan, "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey," *Annals Discrete Mathematics*, Vol. 5, 1979, pp. 278−326.

[14] R. Gopalan, K.T. Talluri, "Mathematical Models in Airline Schedule Planning: A Survey," *Annals of Oper. Res.*, Vol. 76, 1998, pp. 155−185.

[15] M. M. Etschmaier, D.F.X. Mathaisel, "Aircraft Scheduling – The State of the Art," *Proc. 24th AGIFORS Annual Symp.*, 1984, pp. 181− 225.

[16] H. N. Psaraftis, "A Dynamic Programming Approach to the Aircraft Sequencing Problem," MIT Flight Transportation Laboratory, Research Report R78–4, 1978, Cambridge, MA.

[17] V. Ciesielski, P. Scerri, "An Anytime Algorithm for Scheduling Aircraft Landing Times Using Genetic Algorithms," *Australian J. Intelligent Information Processing Systems*, Vol. 4, 1997, pp. 206–213.

[18] T. Grosche, A. Heinzl, F. Rothlauf, "A Conceptual Approach for Simultaneous Flight Schedule Construction with Genetic Algorithms," *Lecture Notes in Computer Science*, *Applications of Evolutionary Computing*, Vol. 2037/2001, pp. 257–267, Springer.

[19] J. E. Beasley, M. Krishnmoorthy, Y.M. Sharaiha, D. Abramson, "Displacement Problem and Dynamically Scheduling Aircraft Landings," *J. Operational Research Society*, Vol. 55, Issue 1, 2004, pp. 54–64, Palgrave, Macmillan.

[20] X.-B. Hu, E. Di Paolo, "An Efficient Genetic Algorithm with Uniform Crossover for Air Traffic Control," *Computers and Operations Research*, Vol. 36, Issue 1, 2009, pp. 245–259.

[21] H. Lee, H. Balakrishnan, "A Study of Tradeoffs in Scheduling Terminal Area Operations," *Proceedings of the IEEE*, Vol. 96, No. 12, 2008, pp. 2081–2095.

[22] H. Helmke, "Scheduling Algorithms for ATM Applications – Tools and Toys", *Proc. Digital Avionics Systems Conference*, October 16–20, 2011, Seattle, WA, pp. 3C2-1–3C2-15.

[23] A. T. Ernst, M. Krishnamoorty, R.H. Storer, "Heuristic and Exact Algorithms for Scheduling Aircraft Landings," *Networks International J.*, Vol. 34, No. 3, 1999, pp. 229–241.

[24] J. E. Beasley, M. Krishnamoorthy, Y.M. Sharaiha, D. Abramson, "Scheduling Aircraft Landings – The Static Case," *Transportation Science*, Vol. 34, Issue 2, 2000, pp. 180–197.

[25] M. Jani, "Modeling the Capacity of Closely-Spaced Parallel Runways using Innovative Approach Procedures," *Transportation Research, Part C: Emerging Technologies*, Vol. 16(6), 2008, pp. 704–730.

[26] M. Kupfer, "Scheduling Aircraft Landings to Closely Spaced Parallel Runways," *Proc. Eighth USA/Europe Air Traffic Management Research and Development Seminar*, 2009, Napa, CA.

[27] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1989, Addison-Wesley Professional.

[28] M. R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 1979, W.H. Freeman and Co., New York.

[29] P. Brucker, A. Gladky, J.A. Hoogeveen, M.Y. Kovalyov, C.N. Potts, T. Tautenhahn, S.L. Van de Velde, "Scheduling a batching machine" *J. Scheduling*, Vol. 1, No. 1, 1998, pp. 31–54.

[30] M. Wall, 1996, "GAlib: A C++ Library of Genetic Algorithm Components," Mechanical Engineering Departement, Massachusetts Institute of Technology, http://lancet.mit.edu/ga/.

Amir H. Farrahi received his B.S., M.S., and Ph.D. in electrical and computer engineering from Sharif Institute of Technology, Illinois Institute of Technology, and Northwestern University, in 1988, 1992, and 1997, respectively.
He has held R&D positions at IBM Research, Synplicity, Sun Microsystems, and Dash Navigation. Since 2008, he has been with UC Santa Cruz, working on various aspects of Next Generation Air Transportation System. His research interests include the theory of computation, design and analysis of algorithms, electronic design automation, shortest paths and graph algorithms, and air traffic management.
Dr. Farrahi is a member of the ACM and has served on the program and organizing committees for a number of ACM and IEEE conferences.

Savita A. Verma received her M.S. in human factors and ergonomics from San Jose State University in 2001.
She has been a researcher in the Airspace Systems Division at NASA Ames Research Center for the last 13 years. Her research interests include human performance modeling, data link, surface operations and very closely spaced parallel runways.
Ms. Verma is a senior member of the AIAA. She has published about 20 papers in the field and was the recipient of the NASA Aeronautics Directorate Technical Excellence in Publications Award in 2010.

Thomas E. Kozon received his M.S. From the California State University and his B.S. from Michigan State University.
He has been employed at NASA Ames Research Center as a Systems Engineer for over 27 years and is currently affiliated with UC Santa Cruz. His work experience includes fast-time, full-mission and part-task simulation research in airspace, surface and air traffic control domains. His current work interests include NextGen and Super Density Operations.
Mr. Kozon has received many awards during his tenure at NASA Ames, including the Turning Goals Into Reality (TGIR) award, NASA Certificate of Recognition for Creative Development of a Technical Innovation, Aeronautics Directorate Technical Excellence in Publications Award, and a number of best paper awards at professional conferences.