An Approximation Algorithm for Scheduling Aircraft with Holding Time*

Alexandre M. Bayen† Claire J. Tomlin‡ Yinyu Ye§ Jiawei Zhang¶

April 1, 2003

Abstract

We consider the problem of scheduling arrival air traffic in the vicinity of large airports. The problem is posed as a single queue problem, from which aircraft can be pulled out and put "on hold", in holding loops, each loop taking a fixed amount of time to traverse, before they join the queue again. The difficulty of deriving efficient solutions to this problem (which is currently controlled non optimally by human Air Traffic Controllers) is the minimization of "idle time" generated by traversing an integer number of loops.

We formulate this problem as a single machine scheduling problem where we are given N jobs denoted by $J_1, J_2, \cdots, J_N$. Each job J_j is characterized by two nonnegative numbers r_j and d_j. We are given two nonnegative numbers Δ (processing time), and T (holding time). In a feasible schedule (i) each job J_j is assigned a starting time τ_j, such that $r_j + l \cdot T \leq \tau_j \leq d_j + l \cdot T - \Delta$, for some integer $l \geq 0$; (ii) if $j \neq j'$ then $|\tau_j - \tau_{j'}| \geq \Delta$. Our goal is to find feasible schedules to alternatively minimize two objectives: the sum of the starting times of all jobs and the makespan (the time at which all jobs are finished).

To the best of our knowledge, this scheduling problem has not been studied before in the literature, at least from the approximation algorithm point of view. We present approximation algorithms which can alternatively approximate two objectives with factors of 5 and 3, respectively. Our main algorithm consists of solving two subproblems, one of which is solved optimally using dynamic programming, while the other is solved approximately using linear programming relaxation and rounding.

*Research supported by NASA under Grant NCC 2-5422, by ONR under MURI contract N00014-02-1-0720, by DARPA under the Software Enabled Control Program (AFRL contract F33615-99-C-3014), by NSF under the grant DMI-0231600 and by a Graduate Fellowship of the Délégation Générale pour l'Armement (France).

†Ph.D. Student, corresponding author. Aeronautics and Astronautics, Durand 028, Stanford University, Stanford CA, 94305-4035. Tel: (650)498-0530, Fax: (650)723-3738. `bayen@stanford.edu`

‡Assistant Professor, Aeronautics and Astronautics, Courtesy Assistant Professor, Electrical Engineering; Director, Hybrid Systems Laboratory, Stanford University.

§Professor, Management Science and Engineering and, by courtesy, Electrical Engineering; Stanford University

¶Ph.D. Student, Management Science and Engineering, Stanford University.

1 Introduction

1.1 Motivation

The almost uninterrupted growth of air traffic in the US in the last decades has motivated the design of a semi-automated *Air Traffic Control* (ATC) system to help Controllers managing the increasing complexity of traffic flow in the vicinity of major airports. However, some of the crucial tasks of ATC are still performed manually: conflict avoidance, and some aspects of scheduling. In the current ATC system, scheduling arriving flights incoming into busy airports is facilitated by CTAS [8], a software used to compute optimal descent routes for aircraft arriving to destination airports [11]. The combinatorial aspect of incoming traffic optimization in the direct vicinity of airports (i.e. up to 200 nautical miles before landing) has not been addressed in a global manner to this day: the solutions implemented by Controllers result from combinations of procedures established and validated over time with heuristics [10].

The ATC problem which the controllers currently solve manually is the following: each aircraft has an earliest possible arrival time, and a delayed arrival time (if it achieves minor path deviations on the way). The aircraft is allowed to arrive in the interval between these two times. Ideally, these intervals are such that there exists an assignment which guarantees the delivery of one aircraft at most every Δ time units at the airport. Δ is sometimes called the *metering time*. This problem is referred to as the *single interval problem*. If such an assignment is not possible, ATC decides to put certain aircraft "on hold" in order to meet the Δ metering constraint of the airport. The choice of aircraft which are put on hold today is not optimized, but obeys experienced-based rules (called *playbooks*).

This problem can be formulated as a single machine scheduling problem as follows. We are given N jobs denoted by $J_1, J_2, \cdots, J_N$. Each job J_j is characterized by two nonnegative numbers r_j and d_j. We are given two nonnegative numbers Δ (processing time), and T (holding time). In a feasible schedule (i) each job J_j is assigned a starting time τ_j, such that $r_j + l \cdot T \leq \tau_j \leq d_j + l \cdot T - \Delta$, for some integer $l \geq 0$ which represents the number of holding patterns used; (ii) if $j \neq j'$ then $|\tau_j - \tau_{j'}| \geq \Delta$. Our goal is to find a feasible schedule such that one of the two following objectives are minimized: the makespan $C_{\max} = \max_{1 \leq j \leq n} \tau_j + \Delta$ (the time at which all jobs are finished) and the sum of the starting times of all jobs $\sum_{j=1}^{n} \tau_j$.

1.2 Related Results

To the best of our knowledge, this scheduling problem has not been studied before in approximation algorithm literature.

In [4], the authors posed the general aircraft scheduling problem with holding time. They used an integer programming approach to solve the problem, using CPLEX, and showed the limit of this approach (the computational time grows exponentially with the number of aircraft, making this approach unattractive for online implementations). In [5], the authors present a polynomial time algorithm for solving the single interval case: the objective is to maximize $\delta := \min_{i<j} |\tau_j - \tau_{j'}|$ without putting the aircraft on hold. In the general case, $\delta < \Delta$, which forces ATC to put aircraft on hold: the present algorithm has been developed to address this situation.

Our problem generalizes those studied by Dantzig and Fulkerson [7], and Gertsbakh and Stern [9] where each job's starting time can only be in a single interval $[r_j, d_j - \Delta]$. To find a feasible schedule for the given jobs, one can schedule them on parallel identical machines. Here, the goal is to minimize the number of machines needed. One can easily see that this problem is equivalent to our problem when the objective is to minimize the number of periods we have to use and when all $d_j \in [0, T]$. The problem can be solved to optimality if $r_j = d_j - \Delta$ for for j. If $r_j < d_j - \Delta$, the complexity of the problem is still open.

Another closely related scheduling problem is the one studied by Carlier [6] and Baptiste [3] where the job's starting time is restricted to a single interval $[r_j, d_j - \Delta]$ and the objective is to maximize the number of jobs which can be scheduled. This problem can be solved in polynomial time [3]. Their results imply that for our problem, if all the jobs can be scheduled in the first available interval for each aircraft, the problem is polynomial time solvable.

Notice that when the processing time for each job is the same, minimizing the sum of starting times of jobs is equivalent to minimizing the sum of completion times of jobs. Designing approximation algorithms for various scheduling problems with the objective minimizing the sum of completion times has received considerable attention during the last decades, see for example an excellent survey by Chen, Potts and Woeginger [2]. However, because of the structure of our problem, it seems that none of the algorithms can be applied to give an approximation for our problem. In most of the models, only release times of jobs are present. However, in the present case, each job has a deadline, although the job can be scheduled in a prescribed later range if this deadline cannot be met.

1.3 Our Results and Techniques

We present an approximation algorithm which approximates the sum of completion times with factor 5. We extend it to approximate makespan with factor 3.

Our first attempt at this problem is based on a simple observation that the starting time of the jobs can be restricted to a polynomial bounded set. Therefore, the problem can be formulated as a constrained assignment problem: assign the jobs to those possible starting points, with the constraints that at most one job can be assigned to any interval of length Δ. As a standard approach, we solve the linear programming (LP) relaxation of this constrained assignment problem. The key in our algorithm is the rounding of the LP solution x, which is fractional in general, into an integer solution for the original problem. Here we observe that there is a way to modify the fractional solution x and obtain another fractional solution $\bar{x}$ (which still satisfies the constraints of the LP) such that the corresponding costs (both makespan and the sum of the starting times) are increased by a factor of at most 3 if all $r_j \geq T$. Then we construct an instance of the (unconstrained) assignment problem such that $\bar{x}$ is a feasible solution of the assignment problem and any integer solution of the new instance will automatically satisfy the constraints of the original problem. Using the fact that we can find an integer solution for the (unconstrained) assignment in polynomial time, whose cost is not worse than that of $\bar{x}$, we find an approximation solution for the original problem with guaranteed error factor if all $r_j \geq T$. However, bounding the ratio of the algorithm if there are jobs with r_j less than T requires an initial step to schedule these jobs.

Motivated by the results of Baptiste [3] that for the single interval case, there exists a polynomial time algorithm that can schedule the maximum number of jobs, we treat the jobs with $r_j < T$ separately. We thus develop a greedy approach: we schedule the maximum number of jobs in the first period $[0, T]$, and then apply the above mentioned LP-based algorithm to the rest of the jobs. Two issues need to be addressed. (i) After we schedule the jobs in the first interval, how do we bound the optimal cost of the rest of the jobs? Indeed, when we apply the LP-based algorithm to the rest of the jobs, we are comparing their cost in this schedule with their cost if they were scheduled by the optimum solution. Our proof provides a bound on the ratio of these two costs. (ii) Scheduling the maximum number of jobs in the first period does not necessarily imply that the sum of starting times of this set of jobs is minimized. Here, we generalize the algorithm and result of Baptiste [3] and apply it to the set of aircraft which can arrive before T: we derive a polynomial-time algorithm for the single interval problem, which schedules (feasibly) the maximum number of jobs; furthermore, we show that among all feasible schedules that schedule the same number of jobs, the algorithm produces a solution that has the minimum sum of starting times. The algorithm is based on dynamic programming.

Combining the above algorithms, we obtain an approximation algorithm which approximates the optimal solution of minimizing the sum of starting times with factor 5. For makespan, we modify both parts of the algorithm and obtain a ratio 3.

1.4 Organization of the Paper

This article is organized as follows. Section 2 describes the optimization program formulation of the physical problem. Section 2.1 shows a trivial 2 approximation algorithm for the case in which $T \leq \Delta$, which works when the objective function is sum of arrival times or makespan. Section 2.2 provides a high level description of the approximation algorithm for the sum of arrival times in the case $T > \Delta$. The part of the algorithm which schedules the first set of jobs before T (dynamic programming) is explained in detail in Section 3. Section 4 presents the part of the algorithm which schedules the rest of the jobs after T (LP rounding). Section 5 explains how to modify the algorithm to provide a 3-approximation algorithm for makespan. The proof of the algorithm of Section 3 is presented in the Appendix.

2 The Algorithm: a High-Level Description

We call a_i the earliest arrival time of aircraft i, and b_i the latest arrival time of aircraft i without holding pattern. In the absence of holding pattern, the problem of scheduling aircraft i in $[a_i, b_i]$ for all $i \leq N$ such that two aircraft are separated by at least Δ is equivalent to scheduling N jobs on a single machine, with processing times Δ, release times $r_i = a_i$, deadlines $d_i = b_i + \Delta$. We call T the time of a holding pattern, and let $[a_j, b_j] + T\mathbb{N} := \bigcup_{k=0}^{\infty}[a_j + kT, b_j + kT]$. The problem can be formulated as follows:

$$\begin{array}{lll} \textbf{Minimize:} & \sum_{j\in\{1,\cdots,N\}} \tau_j & \\ \textbf{Subject to:} & \tau_j \in [a_j, b_j] + T\mathbb{N} & \forall j \in \{1,\cdots,N\} \\ & |\tau_j - \tau_j'| \geq \Delta & \forall (j,j') \in \{1,\cdots,N\}^2, \text{ such that } j \neq j' \end{array} \quad (1)$$

2.1 A Trivial Algorithm for the $T \leq \Delta$ Case

In this case, the following first come first served algorithm reads:

1. Sort the aircraft by earliest possible time of arrival: without loss of generality, we can write $a_1 \leq a_2 \leq \cdots \leq a_{N-1} \leq a_N$.
2. Construct the following schedule:

$$\begin{array}{l} \tau_1 = a_1 \\ \tau_j = \min\{x \mid x \in [\tau_{j-1} + \Delta, +\infty) \cap ([a_j, b_j] + T\mathbb{N})\} \quad \forall j \in \{2,\cdots,N\} \end{array}$$

Theorem 1. *The first come first served algorithm is a 2 approximation algorithm for makespan and sum of arrival times.*

Proof: Sum of arrival times: Divide the N aircraft into K subsets: $S_1 = \{N_1, \cdots, N_2 - 1\}$, $S_2 = \{N_2, \cdots, N_3 - 1\}$, $\cdots$, $S_K = \{N_K, \cdots, N\}$, where

$$N_1 = 1$$
$$N_k = \min\{p | a_{N_{k-1}} + (p - N_{k-1})(\Delta + T) < a_p\}$$

For all $k \in \{1, \cdots, K-1\}$, for all $j \in S_k$, the algorithm provides $\tau_j \leq a_{N_k} + (j - N_k)(\Delta + T)$. Indeed, if it were not true, we would have a $j \in S_k$ such that $\tau_j > a_{N_k} + (j - N_k)(\Delta + T)$, which would contradict the definition of N_{k+1}. We call C_{fcfs} the cost of this algorithm:

$$C_{\text{fcfs}} \leq \sum_{k=1}^{K} \sum_{j \in S_k} \tau_j \leq \sum_{k=1}^{K} \sum_{j \in S_k} (a_{N_k} + (j - N_k)(\Delta + T)) \leq \sum_{k=1}^{K} \sum_{j \in S_k} \frac{a_{N_k} + (j - N_k)(\Delta + T)}{a_{N_k} + (j - N_k)\Delta} (a_{N_k} + (j - N_k)\Delta)$$

Since we know that $\text{OPT} \geq \sum_{k=1}^{K} \sum_{j \in S_k} (a_{N_k} + (j - N_k)\Delta)$, we get the following approximation ratio β for the sum of arrival times:

$$\beta = \max_{k=1}^{K} \max_{j \in S_k} \left\{ \frac{a_{N_k} + (j - N_k)(\Delta + T)}{a_{N_k} + (j - N_k)\Delta} \right\} = \max_{k=1}^{K} \max_{j \in S_k} \left\{ 1 + \frac{(j - N_k)T}{a_{N_k} + (j - N_k)\Delta} \right\} \leq 1 + \frac{T}{\Delta} \leq 2 \quad (2)$$

Makespan: We know that $\tau_N \leq a_{N_K} + (N - N_K)(\Delta + T)$. Since the jobs have been ordered such that $a_1 \leq a_2 \leq \cdots \leq a_{N-1} \leq a_N$, the arrival time τ_N^* of aircraft N in the optimum schedule has to satisfy $\tau_N^* \geq a_{N_K} + (N - N_K)\Delta$. The approximation ratio α for makespan thus satisfies:

$$\alpha \leq \frac{a_{N_K} + (N - N_K)(\Delta + T) + \Delta}{a_{N_K} + (N - N_K)\Delta + \Delta} \leq 1 + \frac{(N - N_K)T}{a_{N_K} + (N - N_K)\Delta + \Delta} \leq 2$$

□

2.2 The Case $T > \Delta$

Before presenting our algorithm, we introduce some notation and initial steps.

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

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

(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 *(i)* is used for notational convenience. Step *(ii)* enables us to construct a grid of polynomial size: it separates aircraft into groups for which we have at least a feasible solution (which we can construct using the trivial algorithm of Section 2.1). In step *(iii)*, the K groups are gridded

(σ_k) and assembled in a single grid Σ of size $O\left(\frac{T}{\Delta}N^2\right)$. The set $\bigcup_{l\in S_k}\{a_l + \Delta\mathbb{N} + T\mathbb{N}\}$ is the set of earliest arrival times plus an integer number of T (holding patterns) and/or Δ (time to process one aircraft). It follows from Lemma 3 that the optimum schedule lies on this grid (the optimum is sometimes referred to as *left shifted*).

Main Algorithm

1. 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 algorithm of Section 3. If this number is equal to N, stop.

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

3. Modify the x_{ij} using the procedure $\overline{x}_{ij} = \mathsf{TransformLPsol}(x_{ij})$.

4. Compute the integer solution $\tilde{x}_{ij} = \mathsf{Matching}(\overline{x}_{ij})$ of the matching problem constructed by the Matching procedure. The result is a feasible schedule for the remaining aircraft.

In the previous algorithm, $I(i)$ represents the set of aircraft for which the arrival times are less than Δ time units after t_i, and thus not allowed if t_i is chosen by an integer solution of (3). $G(j)$ represents the set of i such that t_i is available for aircraft j.

We call n the number of aircraft scheduled before T by the algorithm of Step 1, and $m = N - n$ the number of aircraft scheduled after T. We use the following notation:

$C(n)$	cost of scheduling n jobs before T with the algorithm of Step 1 (cost means sum of arrival times)
$C(LP, m)$	minimum cost of the relaxed LP (3) for the m jobs
$C(IP, m)$	minimum cost of a feasible integer solution to (3) for the m jobs
$C(\overline{x}, m)$	cost of the fractional solution (for the m jobs) $\overline{x}_{ij}$ of (3) produced by Step 3
$C(\tilde{x}, m)$	cost of the fractional solution (for the m jobs) $\tilde{x}_{ij}$ produced by Step 4
$\mathrm{OPT}(m)$	cost of the m jobs when they are scheduled by the optimal solution OPT
$\mathrm{OPT}(n, \text{first})$	cost of the first n jobs when they are scheduled by OPT
$\mathrm{OPT}(m, \text{last})$	cost of the last m jobs when they are scheduled by OPT

Lemma 1. *Consider the $|F|$ aircraft of F (step 1 of the algorithm). Step 1 schedules the largest number of them in $\bigcup_{i=1}^{|F|}[a_i, b'_i]$, with minimum sum of arrival times. The complexity of Step 1 is $O(|F|^9)$.*

Lemma 2. *The sum of starting times for the $m = N - n$ other aircraft scheduled after T is bounded above by $5 \cdot OPT(m, last)$. The complexity of scheduling these m aircraft is dominated by solving the LP (3).*

Theorem 2. *The Main Algorithm has a 5-approximation ratio for the sum of arrival times. Its complexity is $O(N^9)$, and is dominated by step 1.*

Proof of Theorem 2: Let c_n be the cost of the n jobs scheduled before T by Step 1. Let c_m be the cost of the m jobs scheduled after T by Step 4. By Lemma 1, we have $c_n \leq \mathrm{OPT}(n, \text{first})$. By Lemma 2, we have $c_m \leq 5 \cdot \mathrm{OPT}(m, \text{last})$. Now combining the two bounds for c_n and c_m, we get:

$$\begin{aligned} c_n + c_m &\leq \mathrm{OPT}(n, \text{first}) + 5 \cdot \mathrm{OPT}(m, \text{last}) \\ &\leq 5 \cdot \mathrm{OPT} \end{aligned}$$

The complexity of Step 1 is $O(|F|^9)$. The worst case is when $|F| = N$ (all jobs can be scheduled in F), therefore the complexity of Step 1 is $O(N^9)$. The complexity of the other steps is determined by solving the linear program (3), so the overall complexity is $O(N^9)$. □

3 Dynamic Programming Algorithm for the first subproblem

We call $|F| = f \in \mathbb{N}$. We are given a set of time intervals $[r_i, d_i]$, $i \in \{1, \cdots, f\}$, and assume that the d_i have been sorted in chronological order: $d_1 \leq d_2 \leq \cdots \leq d_f$. A schedule $\{t_i\}_{i \in \{1,\cdots,l\}}$ is said to be admissible if for all i in $\{1, \cdots, l\}$, $t_i \geq r_i$ (jobs start after their release time), $t_i + \Delta \leq d_i$ (jobs are finished before their deadline), and $\forall i \neq j$, $|t_i - t_j| \geq \Delta$ (two jobs cannot be processed simultaneously). For l given ($l \leq f$), we want to compute $\min \sum_{p=1}^{l} t_{i_p}$, the minimum of the sum of starting times of l of the f jobs.

Definition 1.
- *We call $\Theta = \{t \mid \exists i \in \{1, \cdots, f\},\ \exists l \in \{0, \cdots, f\},$ such that $t = r_i + l\Delta\}$.*
- *We call $U_k(s, e) = \{J_i \mid i \leq k$ and $s \leq r_i < e\}$ the set of jobs of index less than k released between s and e.*
- *For a given l, we call $C_k(s, e, l)$ the minimum sum of starting times among the set of admissible schedules containing l jobs in $U_k(s, e)$, which satisfy*

(i)	*for all i, $t_i \geq s + \Delta$*	*(job i starts after $s + \Delta$)*
(ii)	*for all i, $t_i + \Delta \leq e$*	*(job i is processed before e)*
(iii)	*for all i, $t_i \in \Theta$*	*(starting times are in Θ).*

We pose $C_k(s, e, l) = +\infty$ for all $l > k$.

Lemma 3. *(Carlier [6], Baptiste [3]) There exists an optimal schedule such that $t_i \in \Theta$ for any i scheduled.*

Proof: The proof of Lemma 3 is simple and thus omitted. □

Theorem 3. *$C_k(s, e, l)$ can be computed recursively by the following formula:*

$$C_k(s, e, l) = C_{k-1}(s, e, l)$$

if $r_k \notin [s, e)$ for the k^{th} job J_k, and

$$C_k(s, e, l) = \min \begin{cases} C_{k-1}(s, e, l), \\ \\ \displaystyle\min_{\substack{s' \in \Theta \\ 1 \leq q \leq l-1 \\ \max(r_k, s+\Delta) \leq s' \leq \min(d_k, e) - \Delta}} \left(C_{k-1}(s, s', l-q) + s' + C_{k-1}(s', e, q-1) \right) \end{cases}$$

otherwise.

The proof of Theorem 3 is provided in the Appendix. Now we are ready to prove Lemma 1.

Proof of Lemma 1: The aircraft scheduling problem can be transformed into a job scheduling problem by letting $r_i = a_i$ and $d_i = b_i' + \Delta$, for $i \in F$, as defined in the Main Algorithm. We call $f = |F|$. For notational convenience, we can relabel these jobs from 1 to f. Note that f can range from 1 to N. We can now apply Theorem 3. For all $k,\ s,\ e,\ l$, such that

$$\begin{aligned} & k \leq f \\ & s \geq \min_{i=1}^{f} r_i - \Delta \\ & e \leq \max_{i=1}^{f} d_i \\ & l \leq f \end{aligned}$$

one can compute recursively $C_k(s, e, l)$, and find l_0, the largest l such that $C_f(\min_{i=1}^{f} r_i - \Delta, \max_{i=1}^{f} d_i, l_0)$ is finite.

k and l range from 1 to f, e and s are on the grid Θ of Lemma 1, thus they are in a set of size $O(f^2)$. The size of the dynamic program table to build is thus $O(f^6)$. In building the table, we take the minimum over a set of size $O(f^3)$, since it is indexed by $s' \in \Theta$ and $q \in \{1, \cdots, l-1\}$. Thus, the total cost of the algorithm is $O(f^9)$. $\square$

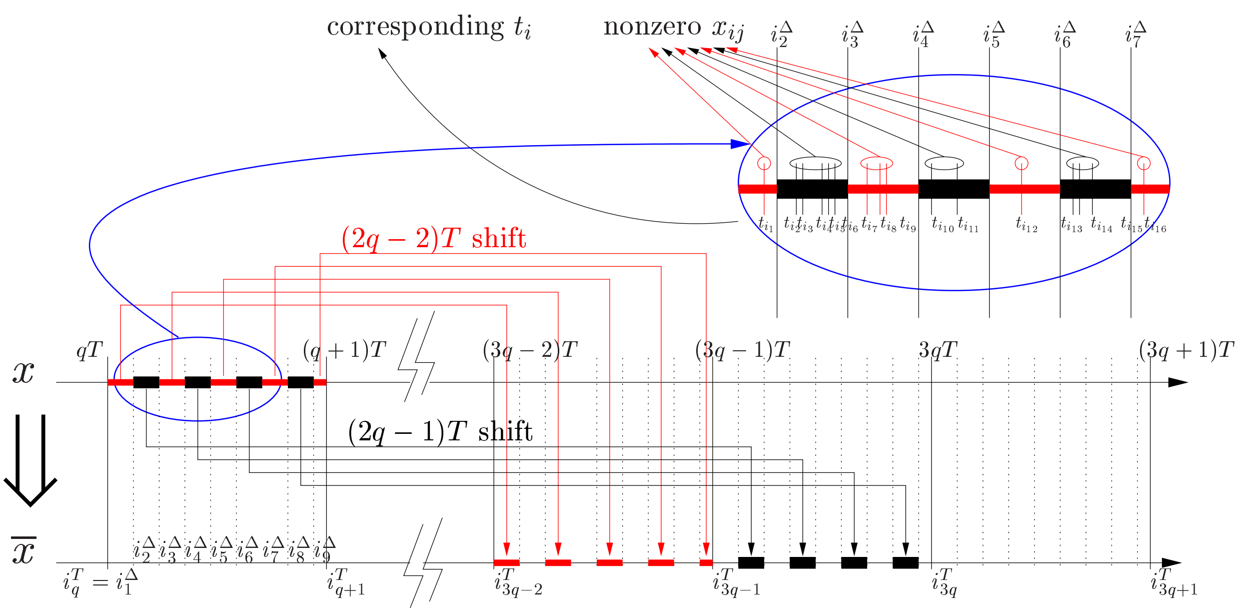

Figure 1: Illustration of the procedure TransformLPsol. This procedure transforms the feasible fractional solution $x = x_{ij}$ of (3) into another feasible fractional solution $\overline{x} = \overline{x}_{ij}$ with idle time. Half of the fractional x_{ij} in the interval $[qT, (q+1)T)$ is shifted to $[(3q-2)T, (3q-1)T)$ (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.

4 Linear Programming Based Algorithm for the Second Subproblem

4.1 Description of the algorithm

This section summarizes steps 2-3-4 of the Main Algorithm. We first solve the relaxed LP (3). This LP is a constrained assignment problem: aircraft j are assigned arrival times t_i; the last constraint means that any integer solution of this problem can only assign one t_i every Δ (by definition of the interval $I(i)$). In the relaxed version, it means that the x_{ij} (fractional assignments) sum to at most 1 over a period of length Δ.

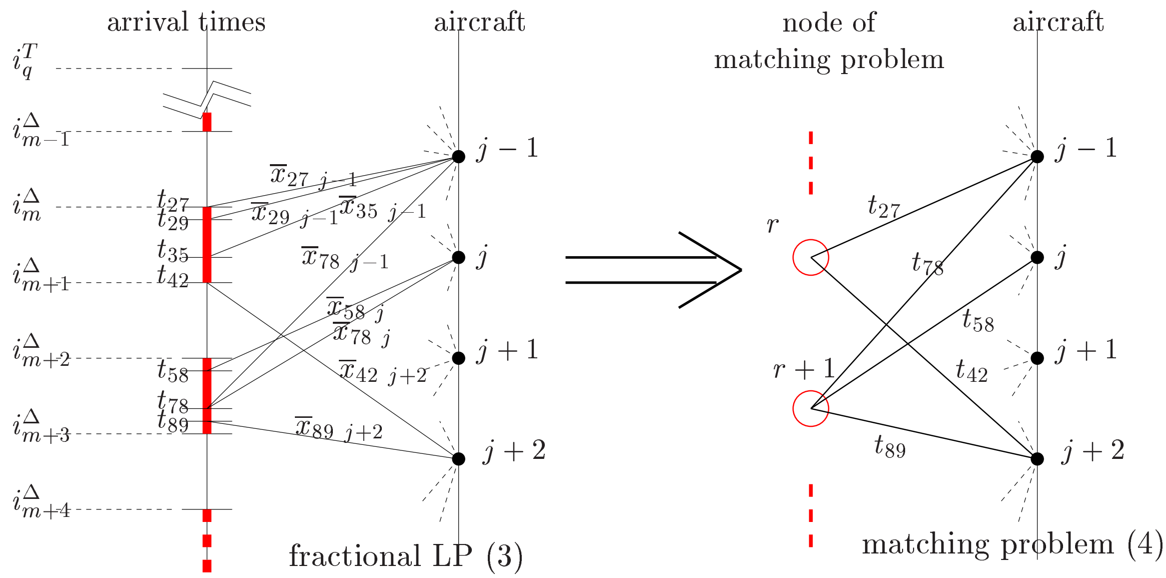

Figure 2: Illustration of the procedure Matching. This procedure transforms the feasible fractional solution $\overline{x}_{ij}$ into a weighted matching problem. In the fractional LP solution, only the nonzero $\overline{x}_{ij}$ are represented (solid if they are in the range of the figure, dashed 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, $\Theta_{rj-1} = t_{27}$.

The fractional solution x is transformed to another fractional solution $\overline{x}$ by the procedure TransformLPsol (illustrated in Figure 1). The fractional solution is decomposed into sets of t_i such that $t_i \in [qT, (q+1)T)$, where $q \in \mathbb{N}$. Each of these sets is decomposed into chunks of length Δ. Figure 1 shows one of these intervals (eight chunks of length Δ, and the last chunk of length less than Δ). The x_{ij} in each of these chunks is shifted by $(2q-2)T$ or $(2q-1)T$ depending on parity, in order to insert idle time of length Δ between the corresponding chunks (see arrows in Figure 1). Because of the last chunk (interval $\{i_9^\Delta, \cdots, i_{q+1}^\Delta\}$ in Figure 1), it is necessary to insert a period of idle time of length at least Δ after the highest shifted x_{ij} (the black chunk coming from $\{i_8^\Delta, \cdots, i_9^\Delta\}$ in Figure 1). This is done by adding a full interval $[3qT, (3q+1)T)$ of idle time after $3qT$.[1] The result is another fractional solution $\overline{x}$ which satisfies (3) and has idle time periods of length Δ alternating with non idle time periods of length Δ. This means sets in which $\bar{x}_{ij} = 0$ for t_i ranging in an interval of length Δ alternate with sets of the same length with nonzero $\bar{x}_{ij}$.

[1]This is not optimal, but enables us to construct $\overline{x}$ systematically. This might be improved in the future.

Procedure TransformLPsol

Input: fractional solution x_{ij} of (3).
Output: $\overline{x}_{ij}$, another feasible fractional solution of (3).

Perform the following assignment:

- $\overline{x}_{ij} = x_{i'j}$ where $t_i = t_{i'} + (2q-2)T$, if $\exists q \geq 0$, $\exists l \geq 0$, such that $l < \lfloor \frac{T}{\Delta} \rfloor$ is even, and $t_{i'} \in [qT + l\Delta, qT + (l+1)\Delta)$ or such that $l = \lfloor \frac{T}{\Delta} \rfloor$ is even, and $t_i = t_{i'} + (2q-2)T$ for $t_{i'} \in [qT + l\Delta, (q+1)T)$.
- $\overline{x}_{ij} = x_{i'j}$ where $t_i = t_{i'} + (2q-1)T$, if $\exists q \geq 0$, $\exists l \geq 0$, such that $l < \lfloor \frac{T}{\Delta} \rfloor$ is odd, and $t_{i'} \in [qT + l\Delta, qT + (l+1)\Delta)$ or such that $l = \lfloor \frac{T}{\Delta} \rfloor$ is odd, and $t_i = t_{i'} + (2q-1)T$ for $t_{i'} \in [qT + l\Delta, (q+1)T)$.

The two last items correspond to the last (incomplete) chunk of $[qT, (q+1)T)$.

Procedure Matching

Input: fractional solution $\overline{x}_{ij}$ of (3).
Output: integer solution of a matching problem solving (3).

1. Construct the following set of Θ_{qj}:

   $q = 1, r = 1$
   while $\exists t_i > qT$ **such that** $\exists j \in \{1, \cdots, N\}$ **such that** $\overline{x}_{ij} \neq 0$
   $\quad i_1^{\Delta} = i_q^T,\ \ m = 1$
   $\quad$ **while** i such that $t_i = t_{i_m^{\Delta}} + \Delta$ is less than i_{q+1}^T
   $\quad\quad i_{m+1}^{\Delta} = i$ such that $t_i = t_{i_m^{\Delta}} + \Delta,\ \ r = r+1$
   $\quad\quad$ **for** $j = 1$ **to** N
   $\quad\quad\quad$ **if** $\exists i \in \{i_m^{\Delta}, \cdots, i_{m+1}^{\Delta} - 1\}$ s.t. $\overline{x}_{ij} \neq 0$
   $\quad\quad\quad\quad \Theta_{rj} = \min\{t_i \mid i \in \{i_m^{\Delta}, \cdots, i_{m+1}^{\Delta} - 1\}, \overline{x}_{ij} \neq 0\}$ **end if**
   $\quad\quad$ **end for**
   $\quad$ **end while**
   $\quad$ **for** $j = 1$ **to** N
   $\quad\quad$ **if** $\exists i \in \{i_m^{\Delta}, \cdots, i_{q+1}^T - 1\}$ s.t. $\overline{x}_{ij} \neq 0$
   $\quad\quad\quad r = r+1,\ \Theta_{rj} = \min\{t_i \mid i \in \{i_m^{\Delta}, \cdots, i_{q+1}^T - 1\}, \overline{x}_{ij} \neq 0\}$ **end if**
   $\quad$ **end for**
   $\quad q = q+1$
   end while
   for all j, call $H(j)$ the set of r for which Θ_{rj} has been assigned.

2. Solve for the integral solution $\tilde{x}_{ij}$ of the following weighted matching problem:

$$\begin{array}{lll} \textbf{Minimize:} & \sum_j \sum_{q \in H(j)} \Theta_{qj} \tilde{x}_{qj} & \\ \textbf{Subject to:} & \sum_{q \in H(j)} \tilde{x}_{qj} = 1 & \forall j \in \{1, \cdots, N\} \\ & 0 \leq \tilde{x}_{qj} \leq 1 & \forall j \in \{1, \cdots, N\},\ \forall q \in H(j) \\ & \sum_j \tilde{x}_{qj} \leq 1 & \forall q \in \bigcup_{j=1}^N H(j) \end{array} \tag{4}$$

The procedure $\mathsf{Matching}$ takes the new fractional solution $\overline{x}$, and constructs a feasible instance of a weighted assignment problem. An illustration is given in Figure 2. Every chunk $\{i_m^\Delta, \cdots, i_{m+1}^\Delta - 1\}$ with nonzero $\overline{x}_{ij}$ is reduced to a single node indexed by r. Any aircraft j which has one (or more) nonzero $\overline{x}_{ij}$ with $i \in \{i_m^\Delta, \cdots, i_{m+1}^\Delta - 1\}$ is now linked to r (Figure 2 right). The weight on the corresponding link is the smallest t_i with nonzero $\overline{x}_{ij}$ in $\{i_m^\Delta, \cdots, i_{m+1}^\Delta - 1\}$. For example in Figure 2 right, the weight on the link $r \to (j-1)$ is t_{27} because in Figure 2 left, node $j-1$ was such that x_{27}, x_{29} and x_{35} are nonzero. The corresponding weighted assignment problem is (4). A fractional feasible solution $\tilde{x}$ is obtained from $\overline{x}$ by adding all $\overline{x}_{ij}$ emanating from the same chunk towards aircraft j. We know that (4) has an integer optimal, which is therefore less or equal to our fractional feasible solution.

We are now able to prove Lemma 2 (notations are defined on page 6).

Proof of Lemma 2: Let us call c_m the cost of the m jobs scheduled after T. For each of these jobs, we compute an upper bound of the ratio by which the cost is increased by the procedure $\mathsf{TransformLPsol}$ for each nonzero x_{ij}. The minimum cost $C(IP, m)$ of scheduling the m jobs after T is less than the cost of scheduling these m jobs within OPT and shifting them by T. Therefore:

$$C(LP, m) \leq C(IP, m) \leq \mathrm{OPT}(m) + mT$$

Let us define B_q as the sum of fractional x_{ij} in interval $[qT, (q+1)T)$:

$$B_q = \sum_{i: qT \leq t_i < (q+1)T} \sum_j x_{ij}.$$

Let us call r the largest q such that $B_q \neq 0$. By construction of the LP (3), we have

$$\sum_{q=1}^{r} B_i = m \qquad \text{and} \qquad \sum_{q=1}^{r} B_q \cdot qT \leq C(LP, m)$$

Let us define $C(B_q)$ as the cost of the fractional LP solution of (3) corresponding to the interval $[qT, (q+1)T)$. As shown in Figure 1, the amount by which x_{ij} is shifted is either $(2q-2)T$ or $(2q-1)T$, and therefore the increase in cost of the x_{ij} in interval $[qT, (q+1)T)$ due to the procedure $\mathsf{TransformLPsol}$ is at most $B_q \cdot (2q-1)T$. Thus, we have the following upper bound on the cost c_m of the m jobs scheduled after T by our algorithm.

$$\begin{array}{rcl}
C(\overline{x}, m) & \leq & \sum_{q=1}^{r} \left(C(B_q) + B_q \cdot (2q-1)T\right) \\
& = & C(LP, m) + \sum_{q=1}^{r} B_q \cdot (2q-1)T \\
& \leq & \mathrm{OPT}(m) + mT + \sum_{q=1}^{r} B_q \cdot (2q-1)T \\
& = & \mathrm{OPT}(m) + 2\sum_{q=1}^{r} qB_qT + mT - \sum_{q=1}^{r} B_qT \\
& = & \mathrm{OPT}(m) + 2\sum_{q=1}^{r} qB_qT \\
& \leq & \mathrm{OPT}(m) + 2C(LP, m) \\
& \leq & \mathrm{OPT}(m) + 2\left(\mathrm{OPT}(m) + mT\right) \\
& \leq & 3 \cdot \mathrm{OPT}(m) + 2mT \\
& \leq & 5 \cdot \mathrm{OPT}(m, \mathrm{last})
\end{array}$$

c_m is the optimal solution of the weighted matching problem (4) obtained by the procedure $\mathsf{Matching}$. We know from [1] that there exists at least one integral solution which achieves c_m, which we can find in polynomial time. For this solution, we thus have $c_m \leq C(\overline{x}, m)$ $\square$

5 Modification of the algorithm for makespan

Theorem 4. *The following modification of the* Main Algorithm *is a 3-approximation algorithm for makespan.*

1. *In Step 1, replace the algorithm of section 3 by Carlier's algorithm [6]. If this algorithm schedules N aircraft, Stop. Else, apply Step 2 directly to the N aircraft.*

2. *In step 2, given $C_{\max}$, let us replace (3) with the following feasibility problem:*

$$\begin{array}{ll} \sum_{i\in G(j)} x_{ij} = 1 & \forall j\in \{1,\cdots,N\} \\ x_{ij} \geq 0 & \forall j\in \{1,\cdots,N\},\ \forall i \in G(j) \\ \sum_{i'\in I(i)} \sum_j x_{i'j} \leq 1 & \forall i\in \{1,\cdots,|\Sigma|\} \end{array} \tag{5}$$

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 $\forall j\in \{1,\cdots,N\}$, $G(j) = \Sigma \cap \{[a_j, b_j] + T\mathbb{N}\} \cap [0, C_{\max}]$. If there exists a fractional solution to the set (5) of constraints with corresponding $C_{\max}$-dependent $G(j)$, it represents a fractional schedule of makespan less than $C_{\max}$, which provides a lower bound on the makespan of the original problem. We can compute the smallest possible $C_{\max}$ solving (5) to ϵ using bisection.[2]

Steps 3 and 4 are identical with those in the Main Algorithm*. The complexity of the algorithm is dominated by the solution of the LPs (5) and (4).*

Proof: If Carlier's algorithm schedules the N aircraft before T, it follows from [6] that this schedule provides the optimum makespan. If $n < N$, Carlier's algorithm implies that it is not possible to schedule N aircraft in $[0, T]$. Therefore the optimum makespan $C^*_{\max}$ satisfies $C^*_{\max} > T$. Applying Steps 2-3-4 directly provides a feasible solution $\overline{C}_{\max}$. A similar proof as for Lemma 2 provides the ratio 3 for makesan. □

References

[1] R. K. Ahuja, T .L. Magnanti, and J. B. Orlin. *Network Flows, Theory, Algorithms and Applications.* Prentice Hall, Upper Saddle River, NJ, 1993.

[2] C. N. Potts B. Chen and G. J. Woeginger. A review of machine scheduling: Complexity, algorithms and approximability. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, volume 3, pages 21–169. Kluwer Academic Publishers, Boston, MA, 1998.

[3] 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.

[4] A. M. Bayen and C. J. Tomlin. Real-time discrete control law synthesis for hybrid systems using MILP: applications to congested airspaces. To appear in the *Proceedings of the 2003 American Control Conference.*

[5] A. M. Bayen, J. Zhang, C. J. Tomlin, and Y. Ye. MILP formulation and polynomial time algorithm for an aircraft scheduling problem. Submitted to the *2003 IEEE Conference on Decision and Control.*

[6] J. Carlier. Problèmes d'ordonnancement à durées égales. *QUESTIO*, 5(4):219–229, 1981.

[7] G.B. Dantzig and D.R. Fulkerson. Minimizing the number of tankers to meet a fixed schedule. *Naval Res. Logist. Quart.*, 1:217–222, 1954.

[8] H. Erzberger, H. T. Davis, and S. M. Green. Design of Center-TRACON Automation System. In *AGARD Meeting on Machine Intelligence in Air Traffic Management*, Berlin, Germany, May 1993.

[9] I. Gertsbakh and H. I. Stern. Minimal resources for fixed and variable job schedules. *Operations Research*, 26:68–85, 1978.

[10] J. Histon and R. J. Hansman. The impact of structure on cognitive complexity in air traffic control. Technical Report ICAT-2002-4, MIT, June 2002.

[11] D. Jackson and J. Chapin. Redesigning air traffic control: An exercise in software design. *IEEE Software*, 17(3):63–70, 2000.

[2] In fact, we can compute it in $O(\log|\Sigma|)$ bisection steps, since the total number of grid points is $|\Sigma|$.

Appendix

Proof of Theorem 3

Case 1: $r_k \notin [s, e)$

We have $J_k \notin U_k(s, e)$ by definition of $U_k(s, e)$. We know that $U_k(s, e) \subseteq U_{k-1}(s, e) + \{J_k\}$. Since $J_k \notin U_k(s, e)$, this implies $U_k(s, e) \subseteq U_{k-1}(s, e)$. The reverse inclusion $U_{k-1}(s, e) \subseteq U_k(s, e)$ holds by definition of $U_k(s, e)$, and we therefore have $U_k(s, e) = U_{k-1}(s, e)$. By definition of $C_k(s, e, l)$, this implies that $C_k(s, e, l) = C_{k-1}(s, e, l)$.

Case 2: $r_k \in [s, e)$

We have $J_k \in U_k(s, e)$. We call

$$C' = \min \left\{ C_{k-1}(s, e, l) \ , \min_{\substack{s' \in \Theta \\ 1 \le q \le l-1 \\ \max(r_k, s+\Delta) \le s' \le \min(d_k, e) - \Delta}} \left(C_{k-1}(s, s', l-q) + s' + C_{k-1}(s', e, q-1)\right) \right\}$$

We want to prove that $C' = C_k(s, e, l)$. We do it in two steps.

Step 1: $C' \geq C_k(s, e, l)$.

- If $C' = C_{k-1}(s, e, l)$, the inclusion $U_{k-1}(s, e) \subseteq U_k(s, e)$ implies $C' = C_{k-1}(s, e, l) \geq C_k(s, e, l)$
- If $C' = C_{k-1}(s, s', l-q) + s' + C_{k-1}(s', e, q-1)$ for some s' and q achieving the min, we separate the jobs into three subsets: X, the $l-q$ first jobs; job J_k; and Y, the set of $q-1$ jobs after s'. We first show that $X \cup Y \cup \{J_k\} \subseteq U_k(s, e)$.

  1. $\forall j \in X$, $J_j \in U_{k-1}(s, s') \subseteq U_k(s, s') \subseteq U_k(s, e)$.
  2. $\forall j \in Y$, $J_j \in U_{k-1}(s', e) \subseteq U_k(s', e) \subseteq U_k(s, e)$.
  3. $J_k \in U_k(s, e)$ by assumption.

  Therefore, $X \cup Y \cup \{J_k\} \subseteq U_k(s, e)$, and $|X| + |Y| + 1 = l - 1 + q - 1 + 1 = l$. From this, we see that $X \cup Y \cup \{J_k\}$ is a particular element of the set among which the min is taken in the definition of C' above. Therefore, $C' \geq C_k(s, e, l)$ which is the min among all elements of this set.

Step 2: $C' \leq C_k(s, e, l)$ Call B the instantiation of the l jobs which realizes $C_k(s, e, l)$.

- If $J_k \notin B$, then $B \subseteq U_k(s, e) \backslash \{J_k\} \subseteq U_{k-1}(s, e)$, and $|B| = l$. B is also an instantiation of $C_{k-1}(s, e, l)$, therefore $C_k(s, e, l) \geq C_{k-1}(s, e, l)$. By definition of C', we have $C_{k-1}(s, e, l) \geq C'$. Therefore $C_k(s, e, l) \geq C'$.
- If $J_k \in B$, we first show that $\forall j \in Y$, $r_j > s'$, where s' is the starting time of job J_k.

  Let j be in Y. Assume $r_j \leq s'$. We know that $r_k \leq s'$ by construction. We know that $j \leq k$, because $J_j \in Y$ (and $Y \subseteq B \subseteq U_k(s, e)$; in $U_k(s, e)$, all jobs have index less than k). By assumption, the deadlines have been indexed chronologically, therefore, $d_j \leq d_k$. Call z the completion time of job j. We have $d_j \geq z$. Summarizing all information above, this means that both J_j and J_k can start at s' and finish at z. Therefore switching J_j and J_k is possible and will not change the sum of the starting times of all jobs. The conclusion of this paragraph is that *any job of Y with release time less than s' can be put in X without change of cost.* We can therefore assume that all jobs J_j of Y have a starting time $r_j > s'$

Let J_j be in Y. Because $J_j \in U_k(s,e)$, $j \leq k$. Since $J_j \in Y$, J_j is scheduled after J_k by the previous paragraph. Therefore, $j \leq k-1$. This implies $J_j \in U_{k-1}(s',e)$. Therefore $Y \subseteq U_{k-1}(s',e)$. Let us call $C(Y)$ the cost of scheduling all jobs of Y ($C(Y)$ is the sum of their starting times). $C(Y) \geq C_{k-1}(s',e,|Y|)$ by definition of $C.(,\cdot,\cdot,\cdot)$. For X, similarly, $J_j \in X \subseteq B \subseteq U_k(s,e)$ implies $r_j \geq s$. Also, $r_j < s'$ (because $r_j \leq s' - \Delta$). $j \neq k$ because $J_j \in X$, therefore $j \leq k-1$. From this we deduce $X \subseteq U_{k-1}(s,s')$. This implies $C(X) \geq C_{k-1}(s,s',|X|)$ where $C(X)$ is the cost of scheduling the jobs of X.

$$C(X) \geq C_{k-1}(s,s',|X|)$$
$$C(Y) \geq C_{k-1}(s',e,|Y|)$$

Therefore, writing explicitly the contributions of the different terms in B, we have:

$$C_k(s,e,l) = C(X) + s' + C(Y) \geq C_{k-1}(s,s',|X|) + s' + C_{k-1}(s',e,|Y|) \geq C'$$

The last inequality results from the min in the definition of C'.