Formal Verification of a Terminal Area Scheduler

Alfons Geser and César Muñoz, National Institute of Aerospace
Dieter Hofbauer, University of Kassel, Germany
Johannes Waldmann, University of Leipzig, Germany

The Terminal Area Scheduling Problem

Given:

- a terminal area topology
- aircraft entering at known times at known feeder gates

Wanted: A schedule for the aircraft *(when should it be where?)* such that

- all aircraft stay separated at all times, and
- each aircraft meets its constraints.

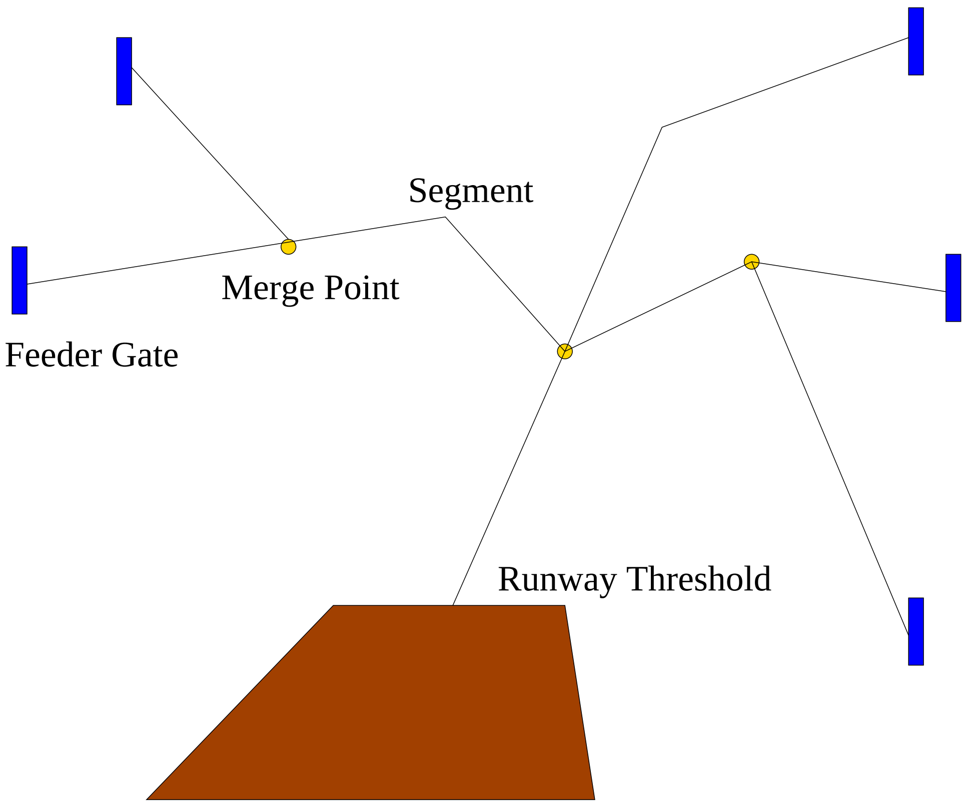
geser@nianet.org

Simplifying Assumptions

- algorithm only (no fault tolerance, system aspects, numerics, etc.)
- all aircraft have the same constraints
- the expected flight time from the feeder gates to the runway threshold is constant (temporal “freeze horizon”)
- acceleration, deceleration, and change of direction are instantaneous

 geser@nianet.org

The Approach Tree



Observations

1. If two unaccelerated aircraft are separated by Δ upon entry to and exit from a segment then so they are at all points in the segment:

$$t_1 + \Delta \leq t_2 \quad \wedge \quad t_1' + \Delta \leq t_2'$$
$$\Longrightarrow$$
$$\forall \beta \in [0,1] : \beta t_1 + (1-\beta)t_1' + \Delta \leq \beta t_2 + (1-\beta)t_2'$$

2. If the speed range of the aircraft is $[v_{min}, v_{max}]$ then the time to pass the distance $d(m)$ from merge point $m-1$ to m is in

$$[t_{min}(m), t_{max}(m)] = [d(m)/v_{max}, d(m)/v_{min}].$$

Reduced Problem Definition

Given: A tree structure of approach paths; a schedule of aircraft entries at the feeder gates.

Wanted: A schedule for the aircraft at the merge points. The schedule is subject to

1. temporal separation of all aircraft at each merge point;
2. lower and upper bounds for the flight times between two successive merge points.

This is a linear, temporal constraint problem.

Balanced Approach Trees

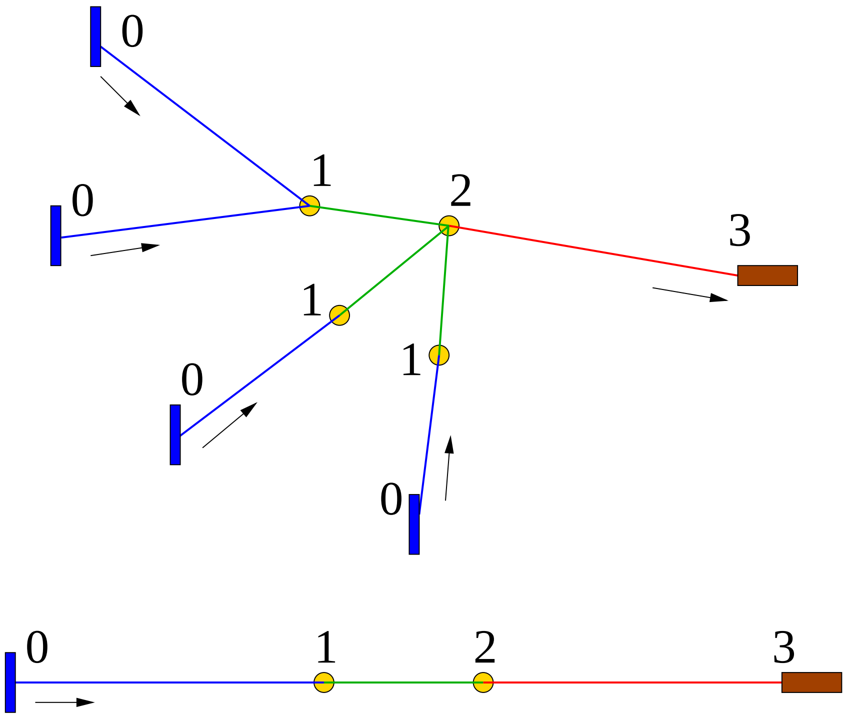
An approach tree is called *balanced* if:

1. all approach paths have the same number of merge points.
2. the timing constraint for the segment between two successive merge points is independent of the approach path and of the aircraft. The time bounds may depend on the merge point number m (new!).

Every approach tree can be made balanced: add virtual merge points as necessary.

Example of a Balanced Approach Tree

equal colors = equal time bounds



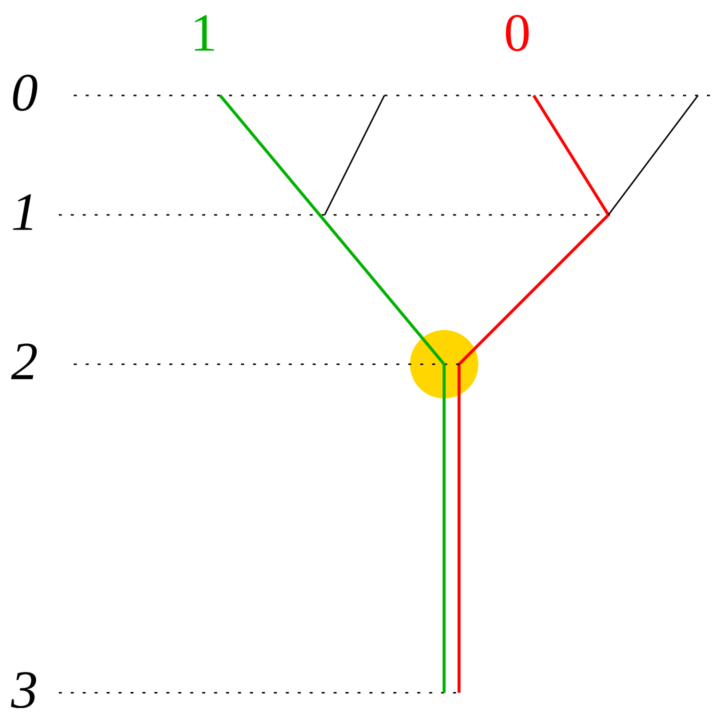
geser@nianet.org

The Parameters

- *aircraft* = sequence number $0, \ldots, N-1$ ($0 =$ first, $N-1 =$ last)
- *merge point* = number $0, \ldots, M-1$ ($0 =$ feeder gate, $M-1 =$ runway threshold)
- *earliest merge point* $\mathrm{mp}(i,k) =$ merge point where aircraft i and k merge
- *entry schedule* $\mathrm{es}(i) =$ arrival time of aircraft i at its feeder gate
- *required time separation* Δ
- *time bounds* $t_{min}(m)$ and $t_{max}(m)$ for trip between merge points $m-1$ to m

Example

$M = 4 \quad N = 2 \quad \mathrm{mp}(0, 1) = 2$



geser@nianet.org

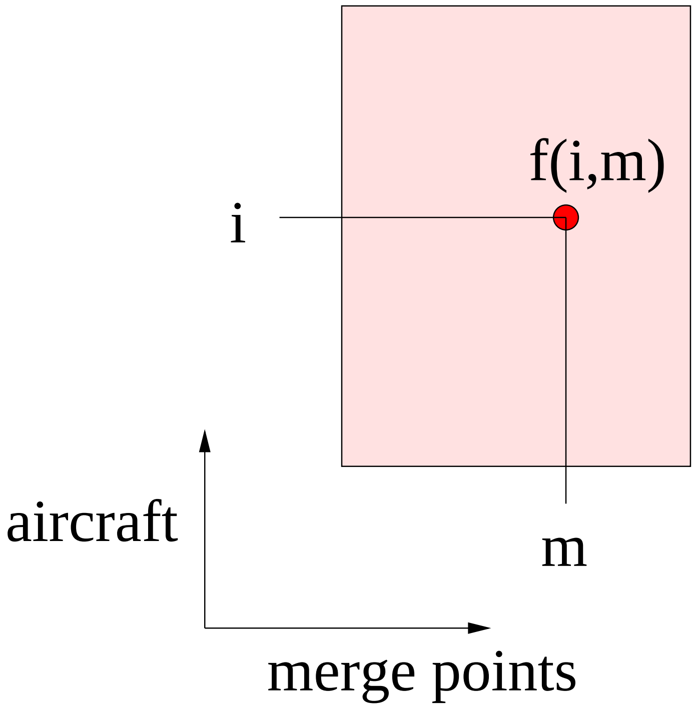
Properties of the Earliest Merge Point Function

The tree structure and the arrival locations are implicitly given by mp. We assume:

- $\mathrm{mp}(i, i) = 0$ for all aircraft i,
- $\mathrm{mp}(i, k) = \mathrm{mp}(k, i)$ for all aircraft i, k *(commutativity)*,
- $m \geq \mathrm{mp}(i, j) \wedge m \geq \mathrm{mp}(j, k) \Rightarrow m \geq \mathrm{mp}(i, k)$ for all merge points m and aircraft i, j, k *(transitivity)*.

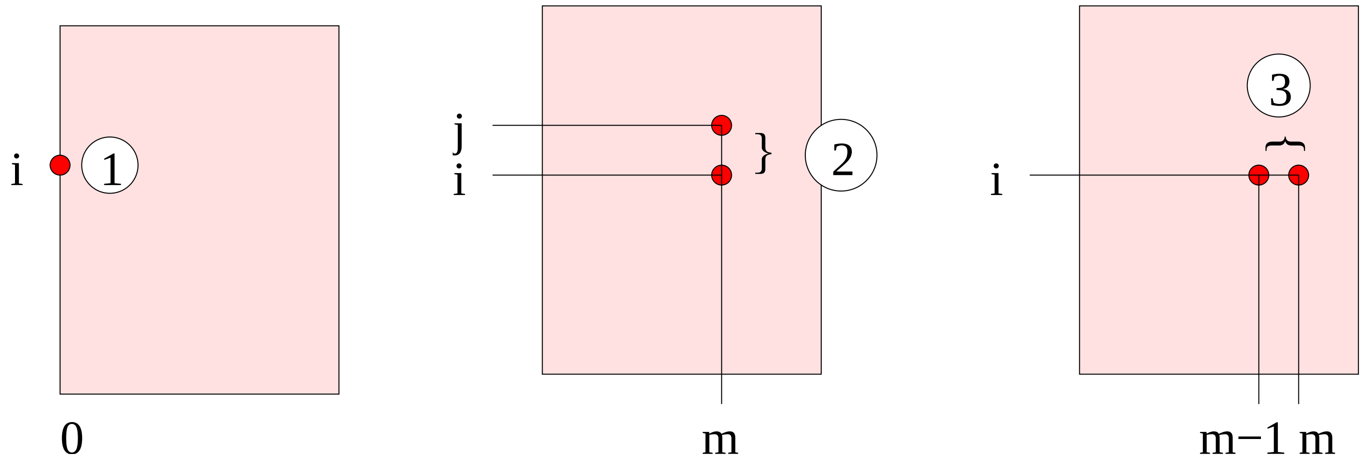
For any given merge point m, the relation $\{(i, k) \mid m \geq \mathrm{mp}(i, k)\}$ is an equivalence relation.

Schedules



A schedule f shall assign each aircraft i and merge point m the time $f(i,m)$ when aircraft i is to arrive at merge point m.

Constraints on Schedules



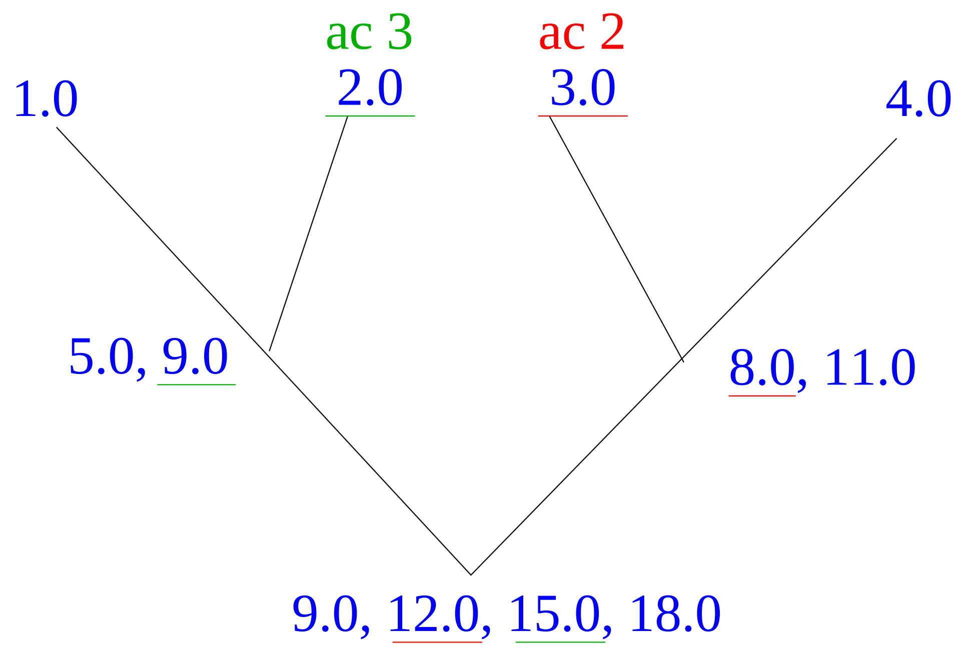
$$\left.\begin{array}{lll} ① = \mathrm{es}(i) & & f \ \textit{founded} \\ ② \geq \Delta, & \text{if } m \geq \mathrm{mp}(i,j) & f \ \textit{safe} \\ ③ \in [t_{min}(m), t_{max}(m)] & & f \ \textit{feasible} \end{array}\right\} \quad f \ \textit{admissible}$$

The *weight* of f is defined by $\mathrm{wt}(f) = \sum_{i=0}^{n-1} f(i, M-1)$.

Example

$t_{min} = 4.0 \qquad t_{max} = 7.0 \qquad \Delta = 3.0$

$\mathrm{es}(1) = 1.0 \quad \mathrm{es}(2) = 3.0 \quad \mathrm{es}(3) = 2.0 \quad \mathrm{es}(4) = 4.0$



$f(3, 2) = 15.0 \quad \mathrm{wt}(f) = 54.0$

First Come, First Serve

- The landing sequence may be permuted. A different landing sequence entails a different instance of the scheduling problem.

- A *permutation* π is called a FCFS sequence if permuted arrival sequence at the feeder gates is sorted: $\mathrm{sorted?}(\mathrm{es} \circ \pi^{-1})$.

- an *inversion* is a pair $i < k$ of indices (aircraft) such that $\mathrm{es}(i) > \mathrm{es}(\mathrm{k})$. The number of inversions measures the distance to an FCFS sequence. (This is the basis of the correctness proof of *bubblesort*.)

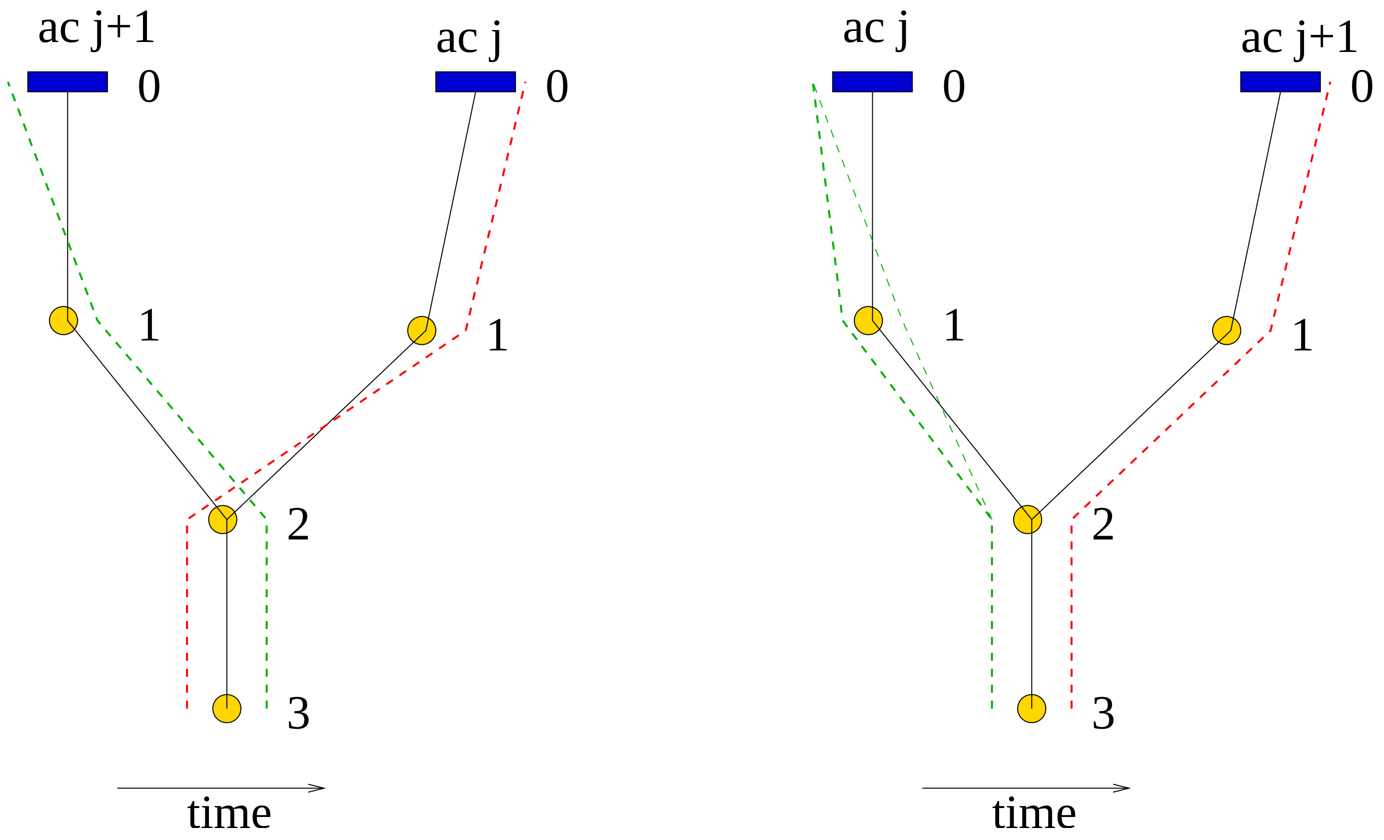
- π can be decomposed into *transpositions* of successive aircraft j and $j+1$ that decrease the number of inversions.

Transposing two Aircraft

Given: an admissible schedule f and two successive aircraft j, $j+1$ such that $\mathrm{es}(j) \geq \mathrm{es}(j+1)$.

Wanted: an admissible schedule f' that "is like f" except that the roles of $j+1$ and j are exchanged.

How to Swap Two Successive Aircraft



Schedule Change for Transposition of Aircraft

Let $n = \mathrm{mp}(j, j+1)$.

$$\mathrm{local\_change}(f, \mathrm{es}, \mathrm{mp}, j)(k, m) =$$

$$\begin{cases} \min(f(j+1, m), f(j, n) - \sum_{p=m}^{n-1} t_{min}(p)), & \text{if } m < n \text{ and } k = j, \\ \max(f(j, m), f(j+1, n) - \sum_{p=m}^{n-1} t_{max}(p)), & \text{if } m < n \text{ and } k = j+1, \\ f(k, m), & \text{else.} \end{cases}$$

geser@nianet.org

Transposition Preserves Admissibility

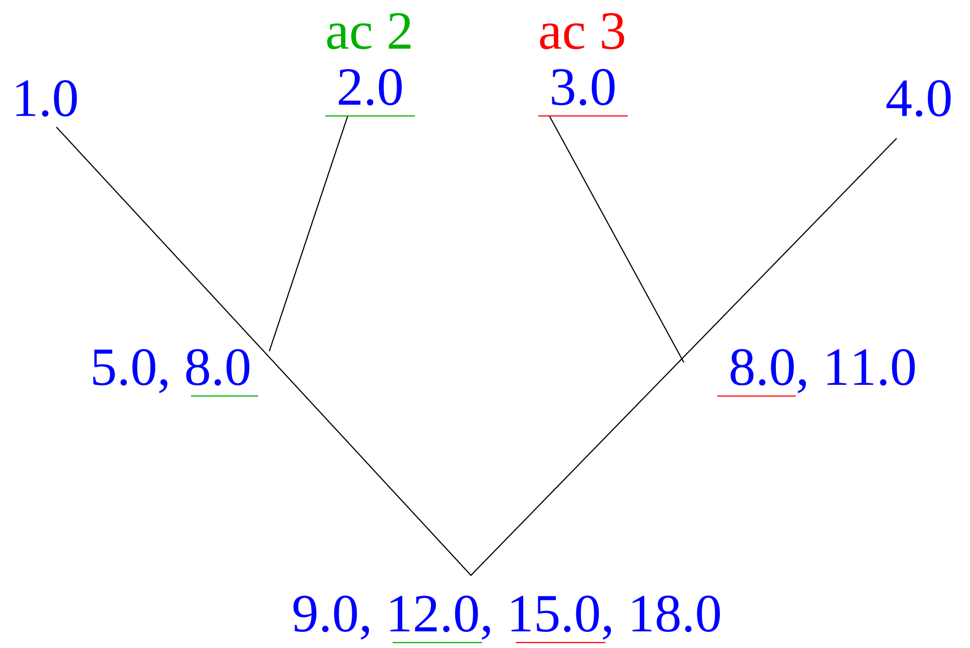
```
fcfs_admissible: LEMMA
  admissible?(f,es,mp) & j < N-1 & es(j+1) <= es(j) =>
    admissible?(local_change(f,es,mp,j),
                es o transpose(j,j+1),
                ap(transpose(j,j+1),mp))
```

In any admissible schedule, any aircraft that enters no earlier than its successor may be transposed with its successor.

Example (after Transposition)

$t_{min} = 4.0 \qquad t_{max} = 7.0 \qquad \Delta = 3.0$

$\mathrm{es}(1) = 1.0 \quad \mathrm{es}(2) = 3.0 \quad \mathrm{es}(3) = 2.0 \quad \mathrm{es}(4) = 4.0$



Schedule Change to an FCFS Landing Sequence

```
global_change(f: schedule, es: entry_schedule, mp: merge_function,
  pi: permutation | sorted?(es o inverse(pi))): RECURSIVE schedule =
IF pi = id THEN f
ELSE
  LET j =
    epsilon! j: j < N-1 & pi(j) > pi(j+1) IN
  global_change(local_change(f,es,mp,j),
                es o transpose(j,j+1),
                ap(transpose(j,j+1),mp),
                pi o transpose(j,j+1))
ENDIF
MEASURE card(inversions(pi))
```


geser@nianet.org

FCFS is Optimal

```
all_fcfs_admissible: THEOREM
  admissible?(f,es,mp) & sorted?(es o inverse(pi)) =>
    admissible?(global_change(f,es,mp,pi),
                es o inverse(pi),
                ap(inverse(pi),mp))

all_fcfs_lighter: THEOREM
  admissible?(f,es,mp) & sorted?(es o inverse(pi)) =>
    wt(global_change(f,es,mp,pi)) = wt(f)
```

If there is an admissible schedule for a given landing sequence then there is an admissible schedule, of the same weight, for a FCFS landing sequence. A tie may be broken arbitrarily.

geser@nianet.org

A Greedy Algorithm; Auxiliary Predicates

- $\text{infrontof?}(i, j, m, \text{mp}) = (i < j \wedge m \geq \text{mp}(i, j))$
- $\text{traffic?}(h, j, m, \text{mp}) =$
  $(\text{infrontof?}(h, j, m, \text{mp}) \wedge \neg\exists k : h < k \wedge \text{infrontof?}(k, j, m, \text{mp}))$
- $\text{expect\_traffic?}(k, m, \text{mp}) = \exists(h, n) : n \geq m \wedge \text{infrontof?}(h, k, n, \text{mp})$

The Greedy Scheduling Algorithm in PVS

```
greedy(es: entry_schedule, mp: merge_function)(k,m): RECURSIVE real =
  IF m = 0 THEN es(k)
  ELSIF expect_traffic?(k,m,mp)
  THEN max(greedy(es,mp)(k,m-1)+t_min(m-1),
           max({r | EXISTS (h,n):
                    n >= m & traffic?(h,k,n,mp) &
                    r = greedy(es,mp)(h,n) + Delta - t_max_star(m,n)}))
  ELSE greedy(es,mp)(k,m-1)+t_min(m-1)
  ENDIF
MEASURE k*M+m
```

Greedy is Optimal

```
greedy_admissible: THEOREM
  admissible?(f,es,mp) => admissible?(greedy(es,mp),es,mp)

greedy_lighter: COROLLARY
  admissible?(f,es,mp) => wt(greedy(es,mp)) <= wt(f)
```

If there is an admissible schedule for a given landing sequence then greedy yields the lightest.

New PVS Libraries

- `permutation_ops`: permutation operations
- `inversions`: transpositions and inversions
- `bubblesort`: (bonus) correctness proof of bubble sort
- `minmax_props`, `finite_sets_minmax_props`: inequations with min and max
- `finite_cross`: types A, B finite implies $A \times B$ finite

geser@nianet.org

Conclusion

- PVS models and formal proofs finished:
  - optimality of first-come-first-serve
  - optimality of the greedy scheduler
- new PVS libraries

Future Work

- introduce aircraft ids invariant to landing order permutation
- introduce feeder gate ids to decouple aircraft from topology
- include geometry; show spatial separation; add speed profiles ...

geser@nianet.org