

Temporal Decoupling and Determining Resource Needs of Autonomous Agents in the Airport Turnaround Process

Pim van Leeuwen
National Aerospace Laboratory NLR
Amsterdam, The Netherlands
Email: leeuwenp@nlr.nl

Cees Witteveen
Delft University of Technology
Delft, The Netherlands
Email: C.Witteveen@tudelft.nl

Abstract

Air traffic in Europe is getting more and more congested with the turnaround process at airports as one of the most constraining factors. During this turnaround process, a number of services need to be provided to aircraft at the gate: de-boarding, cleaning, catering, fueling, etc. These services are provided by different agents (the service providers), who have to coordinate their activities in order to respect the turnaround time slot, the required service times and existing regulations. Usually, a global turnaround plan respecting all temporal dependencies is constructed. Such a global plan, however, has several disadvantages. First of all, it contains several (time)dependencies between agents, preventing them to optimize the scheduling of their activities autonomously. Secondly, in case of disruptions (and delays are common at any airport), re-planning is complex, time-consuming and will often affect all agents. The contribution of this paper in addressing this problem is twofold. First of all, we propose a method to decouple the overall turnaround plan into local plans for each agent, allowing them to schedule their activities independently of one another. The decoupling method guarantees that the merging of all local schedules always satisfies the original set of plan constraints. Moreover, in case of disruptions, we could try to fix the local plans affected instead of repairing the global plan. Secondly, since by decoupling every agent now is free to choose its own schedule, each agent might select a schedule that optimizes its own objectives. Therefore, we show these benefits of decoupling in the turnaround process by developing a new algorithm that can be used by an agent to determine the minimum number of resources it requires to accomplish its ground handling task. We illustrate the application of this algorithm by determining the minimum number of vehicles a fueling agent will need in order to perform all its fueling services in the turnaround process.

1. Introduction

Despite a predicted 5% decrease for 2009 (see [6], [9]), air traffic in Europe is expected to grow again significantly in the longer term. To accommodate this growth, sufficient airspace and airport capacity needs to be ensured. As expansion of airports is expensive and often impossible due to noise and environmental regulations, airport capacity has become a major bottleneck in the air transport system. To still meet traffic demand, airport authorities are seeking improved planning methods to make more efficient use of existing resources.

Airport planning is generally subdivided into a number of domains: arrival management, departure management, stand allocation management, and taxi planning. In all of these areas, already extensive research has been conducted to improve planning and assist planners by means of decision support

tools (e.g., [1], [11], [12], [13]). Ground handling, denoting all processes that take place when an aircraft is at the gate or stand, is a notable exception. It is only recently that research has focused on a more efficient planning of ground handling processes [12]. This seems a promising direction, since ground handling is recognized as the second most common source of delays in the air transport system [14].

This paper presents a new approach to the planning of ground handling activities at airports. These activities (e.g., boarding, fueling, cleaning and baggage loading) are performed during the so-called *turnaround*, when an aircraft is serviced at the stand between two flights. Since these activities are performed by several agents (a boarding operator, a cleaning team, etc.), and are subject to time constraints, the problem domain can conveniently be modeled as a *multi-agent planning and scheduling problem*.

Usually, a global ground handling plan is provided, stating all the constraints that have to be satisfied in order to complete all the ground handling services. Using these constraints, the individual agents would like to schedule their own activities independently from the others, because each of them has his/her own objectives, preferences, business rules and resource constraints that they would like to keep private as much as possible. Due to the interdependencies of these activities, however, it is not directly possible for these actors to develop their own schedule autonomously.

Let's call this problem the *autonomous scheduling problem*: how to ensure that, given a global set of temporal constraints (temporal plan) for a set of agents, each agent is allowed to construct a schedule, satisfying its subset of constraints, without interfering with the schedules of the other agents.

To provide a solution for this autonomous scheduling problem, we present a method to represent such ground handling plans as Simple Temporal Networks (STNs) and show how a well-known method (Temporal Decoupling), developed by Hunsberger [7], can be applied to achieve a set of independently schedulable temporal plans, one for each of the service providers.

Then we show how this method can be used to find out, for each of the service providers involved in the turnaround

process, how a minimum amount of resources (e.g. vehicles) needed to perform the scheduled activities per time period can be determined. We present a very simple, but elegant algorithm that can be used for e.g. the fueling and catering providers to determine how many vehicles they minimally would need to perform their activities given a specification of their part of the turnaround process.

Note that in principle, this approach can be applied to more general problems than the turnaround process alone. Basically, the general version of the problem we discuss in this paper can be stated as follows:

Given: a set Ac of activities to be performed by a set Ag of autonomous agents, for each agent $ag_i \in Ag$ a disjoint subset $Ac_i \subseteq Ac$ of activities to be performed by ag_i and a set C of constraints imposed on the set Ac of activities, where $C_i \subseteq C$ is the set of constraints pertaining to the set of activities Ac_i . Question: (i) how to obtain for each agent $ag_i \in Ag$ a set C'_i of local constraints for its activities $Ac_i \subseteq Ac$, such that each agent ag_i can schedule its activities independently from the others, and (ii) how to provide each agent ag_i with its own resource plan, i.e., an estimation of the number of resources $r \in R$ it minimally needs to carry out its set of activities.

This problem can be viewed as a *coordination* problem, since it requires the establishment of a set of local temporal constraints for each agent in such a way that, while each agent can choose its own schedule and resource plan independently from the others, the feasibility of the overall solution is still ensured.

To illustrate its potential relevance and applicability, we discuss a solution of this problem in the context of the turnaround process at airports. In Section 2 the necessary background on the representation of the problem as an STN and the Temporal Decoupling method is discussed. In Section 3, the algorithm to solve the resource consumption problem is detailed. Finally, in Section 4 some possible extensions of this solution are discussed.

2. Background

2.1. The turnaround process

Ground handling concerns all activities that have to be performed during the turnaround of aircraft at the gate: between on-block (the time an aircraft arrives at the gate) and off-block (the time an aircraft is pushed back from the gate). For this time period, a so-called *turnaround plan* specifies which services (catering, cleaning, boarding, fueling, etc.) should be performed when for specific aircraft. The collection of these turnaround plans constitutes the overall turnaround plan for an airport for a specific day. An example high-level entry of such a turnaround plan is:

(KL1857, C06, 12:00, 13:28)

where the flight number (KL1857), the gate (C06), and the on-block (12:00) and off-block time (13:28) of a specific aircraft is specified. At a lower level, all services such as catering, cleaning, boarding and fueling that need to be planned between the on- and off-block times are listed. Usually, these services are constrained: for each service there is usually a minimal duration (minimum service time) and a maximal duration (norm time) specified. Moreover, there are constraints between several services. For example, fueling cannot take place when passengers are on-board, so fueling has to take place before boarding. Boarding itself has to end at most 15 minutes before off-block.

2.2. Modeling the turnaround process by an STN

To model the collection of turnaround plans we use a *Simple Temporal problem (STP)* [5]:

Definition 2.1: A Simple Temporal Problem S is a tuple $S = \langle X, C \rangle$ where X is a finite collection $\{x_0, \dots, x_n\}$ of time variables, and C is a finite collection of temporal constraints over these variables. Each constraint $c \in C$ is of the form $c_{ij} : x_j - x_i \leq b_{ij}$, for some $b_{ij} \in \mathbb{Z}$. The variable x_0 represents a special fixed time value, the temporal reference point, taking the value 0. ■

To use an STP to specify temporal constraints on activities, every service a for a given aircraft is represented by a pair (x_i, x_{i+1}) of time variables (events) indicating respectively the starting, and the finishing time of a . Moreover, per aircraft we use two additional time point variables indicating the on-block time and the off-block time of the aircraft. The temporal reference point x_0 will usually indicate the beginning of a day (00:00) or afternoon (12:00). So, given n aircraft and m services per aircraft, we need to specify one variable to indicate the temporal reference point, $2n$ variables to indicate the on- and off-block times and $2mn$ variables to indicate the start and finishing times of all the mn activities. Therefore, in total we need $2mn + 2n + 1$ variables.

To specify all constraints for the variables, it suffices to constrain the *duration* of the services by indicating minimum and maximum service times, their *temporal relation* and absolute constraints on the starting and ending times of the services. All these constraints can be specified as upper bounds on the difference of temporal variables. For example, if the start of service a is indicated by x_1 and its ending by x_2 , while the start of service b is indicated by x_3 , then specifying that a might take at least 3 but no more than 5 minutes can be expressed by the constraints $x_2 - x_1 \leq 5$ and $x_1 - x_2 \leq -3$. Requiring b to start after a has ended is specified by $x_2 - x_3 \leq 0$.

Without loss of generality¹ we can assume that for every pair of variables $x_i, x_j \in X$ there is a constraint $c_{ji} : x_i - x_j \leq b_{ji}$ and a constraint $c_{ij} : x_j - x_i \leq b_{ij}$. If these constraints are combined we obtain the *interval constraint* $-b_{ji} \leq x_j - x_i \leq$

1. Note that if x_i and x_j are temporally unrelated, the constraints $x_j - x_i \leq \infty$ and $x_i - x_j \leq \infty$ capture this relation.

b_{ij} , also written as $I_{ij} = [-b_{ji}, b_{ij}]$. In this paper we will use both notations. As a special case we mention the interval constraint $0 \leq x_i - x_j \leq \infty$ specifying that x_j has to occur before x_i .

Let us now present a simplified example of an STN in the turnaround domain.

Example 2.1: Suppose we have a flight with an on- and off-block time of 13:00 and 15:30, respectively. Therefore, all required ground services like fueling, (de)boarding and cleaning have to be done between 13:00 and 15:30. It is required that deboarding has to start within 15 minutes after on-block and it takes at least 10 and at most 20 minutes. Fuelling can only start if deboarding has ended. Finally, we know that fuelling takes at least 20 and at most 40 minutes and has to be completed 30 minutes before off-block. We can model these constraints using an STP as follows:

Consider the following set of variables $X = \{x_0, x_1, x_2, x_3, x_4, x_5, x_6\}$, where

- x_0 = temporal referential point (0 = 12:00)
- x_1 = on block time
- x_2 = begin time deboarding
- x_3 = end time deboarding
- x_4 = begin time fueling
- x_5 = end time fueling
- x_6 = off-block time

The following constraints should be specified:

- | | |
|---------------------------------|--|
| $60 \leq x_1 - x_0 \leq 60$ | on-block is exactly 60 minutes after 12:00 |
| $150 \leq x_6 - x_1 \leq 150$ | the time between on- and off-block is 150 minutes |
| $0 \leq x_2 - x_1 \leq 15$ | deboarding has to start within 15 minutes after on-block |
| $10 \leq x_3 - x_2 \leq 20$ | duration of deboarding is between 10 and 20 minutes |
| $20 \leq x_5 - x_4 \leq 40$ | fueling takes 20 to 40 minutes |
| $0 \leq x_4 - x_3 \leq \infty$ | fueling starts after deboarding has ended |
| $30 \leq x_6 - x_5 \leq \infty$ | fueling has to be completed at least 30 minutes before off-block |

Given an STP $S = \langle X, C \rangle$, a direct labeled graph representation $G_S = \langle N_X, E_C, l \rangle$ of S can be obtained by using

- N_X as the set of nodes n_i representing the time points x_i and
- E_C as the set of labeled directed arcs, where $e = (n_i, n_j) \in E$ has label $l(e) = b_{ji}$ whenever $c_{ji} : x_j - x_i \leq b_{ji}$ occurs in C .

Figure 1 contains a graphical representation (a Simple Temporal Network) derived from this STP S . ■

A *solution* of an STN $S = \langle X, C \rangle$ is a specification of suitable values for time variables $x \in X$:

Definition 2.2: [8] A *solution* for a STN $S = \langle X, C \rangle$ is a complete set of assignments $\{x_0 = 0, x_1 = v_1, \dots, x_n = v_n\}$

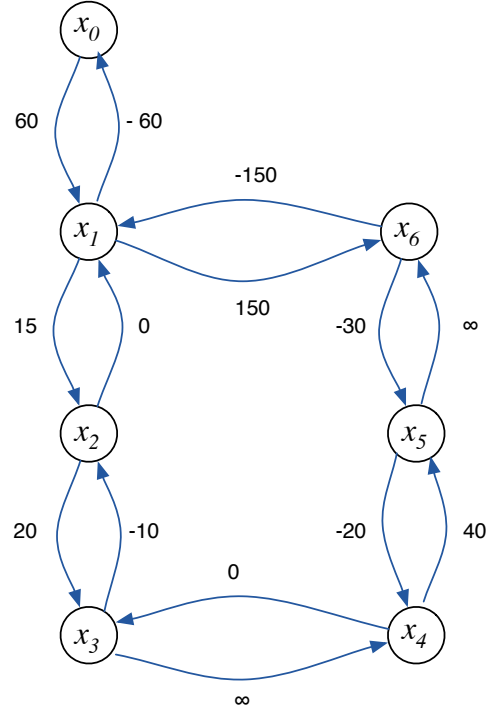


Fig. 1. A Simple Temporal Network G_S derived from the STP S discussed in Example 2.1.

of values $v_i \in \mathbb{Z}$ to variables $x_i \in X$, such that all constraints $c \in C$ are satisfied. ■

Example 2.2: The following assignment is a possible solution to the STP S discussed above:

$$\{ x_0 = 0, x_1 = 60, x_2 = 70, x_3 = 90, \\ x_4 = 105, x_5 = 135, x_6 = 210 \}$$

If such a solution exist, we say that the STN S is *consistent*, else it is said to be *inconsistent*.

There is an efficient algorithm to check (in)consistency of a STN $S = \langle X, C \rangle$ based on the following idea (see [4]): If the labels $l(e)$ on the edges e in the associated graph $G_S = \langle N_X, E_C, l \rangle$ are interpreted as *distances* between nodes, the well-known $O(n^3)$ Floyd-Warshall All-Pair-Shortest-Path (APSP)[3] algorithm can be used to determine the shortest distance $d(i, j)$ between all nodes n_i and $n_j \in N_X$. It is not difficult to show (see [4]) that now the inconsistency of S can be decided by checking whether $d(i, i) \geq 0$ holds for all nodes $n_i \in N_X$.

(To see this note that, if there exists some n_i such that $d(i, i) < 0$, it implies that x_i should occur before itself, which is clearly impossible. Hence, there is no assignment to x_i that satisfies all constraints, implying that S is inconsistent.

Conversely, suppose that for all n_i we have $d(i, i) \geq 0$. Then the shortest path $d(i, j)$ between every pair of vertices is well-defined. We show that S is consistent. Consider the following

assignment: $x_0 := 0$ and for every $i > 0$, $x_i := d(0, i)$. Take an arbitrary constraint $c_{ij} : x_j - x_i \leq b_{ij}$. Since by the shortest path property $d(0, j) \leq d(0, i) + d(i, j)$ and $d(i, j) \leq b_{ij}$, it follows that $d(0, j) - d(0, i) \leq d(i, j) \leq b_{ij}$ and the constraint $c_{i,j}$ is satisfied. Therefore, every constraint is satisfied by this assignment. Hence, S is consistent.)

The graph obtained by applying the APSP algorithm to the STN G_S containing all shortest distances between the nodes in N_X will be denoted by G_S^D . This graph, also called the *d-graph* associated with the STN S , specifies the *tightest* constraints between the time variables in X . Note that the d-graph is a complete graph and contains for every node pair one edge with the shortest distance between them as its label.

2.3. The Temporal Decoupling Method

We model the collection of turnaround plans using an STN. This STN contains the specifications of all activities to be performed by the different agents (service providers) for a collection of aircraft while these are on-block. In order to provide a solution for the complete turnaround process we have to specify the values of all begin and end points of these activities. Finding such a solution (i.e. *joint schedule*) would require either a centralized solution process or a rather elaborate coordination process requiring negotiation between the different agents involved. As we already remarked in the introduction, both these approaches to find a solution are not acceptable, since the service providers require a specification of the constraints between the activities *they* have to perform that enables them to come up with a schedule *independently* of the others.

This requires a modification of the original STN S such that

- S is split into k sub STNs S_i , $i = 1, 2, \dots, k$, where each S_i contains all the constraints for the services of service provider i in the turn round process. We assume that all STNs have the variable x_0 in common.
- each of the agents is allowed to solve its own sub STN S_i by specifying an arbitrary solution s_i (schedule) for it.
- whatever solutions s_j are chosen by the agents, their *merge*, i.e., a complete solution $s = s_1 \cup s_2 \cup \dots \cup s_k$, always constitutes a valid solution to the overall problem S .

This is exactly the idea behind the so-called *Temporal Decoupling* method specified by Hunsberger [8]. This method can be described in a more formal way as follows:

Definition 2.3: (Temporal Decoupling)

Let $X_1, X_2 \subseteq X$ two subsets of the set of variables X such that both X_1 and X_2 contain x_0 and $X_1 - \{x_0\}$ and $X_2 - \{x_0\}$ partition $X - \{x_0\}$. Such a (near) partitioning is called a *z-partition* of X .

A temporal decoupling of the STN $S = \langle X, C \rangle$ using X_1 and X_2 is a pair of STNs $S_1 = \langle X_1, C_{X_1} \rangle$ and $S_2 = \langle X_2, C_{X_2} \rangle$ such that:

- S_1 and S_2 are consistent,

- the merging of solutions for S_1 and S_2 always is a solution for S .

Here, $S_i = \langle X_i, C_{X_i} \rangle$ is the sub-STN generated by X_i by selecting the constraints $c \in C$ that contain variables only occurring in X_i .

The definition of the z-partition and the temporal decoupling for more than two sets is analogous to that for two sets. In our application we choose the z-partition in such a way that all time points belonging to activities of one service provider occur in one z-partition. We will now give an example of z-partitions and the essence of the temporal decoupling process.

Example 2.3: Suppose we have two time point variables x_1 and x_2 with the constraints: $0 \leq x_1 \leq 60$, $0 \leq x_2 \leq 120$ and $0 \leq x_2 - x_1 \leq \infty$, i.e., $x_1 \leq x_2$. Let $x_0 = 0$ (See Figure 2). We consider the following z-partition: $X_1 = \{x_0, x_1\}$ and $X_2 = \{x_0, x_2\}$. We can't solve for x_1 independently from x_2 , since the inter-block constraint $x_1 \leq x_2$ has to be satisfied. For example, if we take $x_1 = 50$ and $x_2 = 30$ then they both satisfy the local constraints $0 \leq x_1 \leq 60$ and $0 \leq x_2 \leq 120$, respectively, but $x_1 \leq x_2$ is not satisfied. Temporal decoupling now essentially comes down to make this latter constraint obsolete by *tightening* the local constraints. In this case, this can be easily accomplished by changing the constraint $0 \leq x_2 \leq 120$ into $60 \leq x_2 \leq 120$. The effect of this change is that the constraint between x_1 and x_2 is always satisfied and can be removed. As a consequence, the values for x_1 and x_2 meeting the local constraints now can be chosen independently from each other, while still guaranteeing a correct total solution: every value chosen for x_1 cannot be larger than an independently chosen value for x_2 . ■

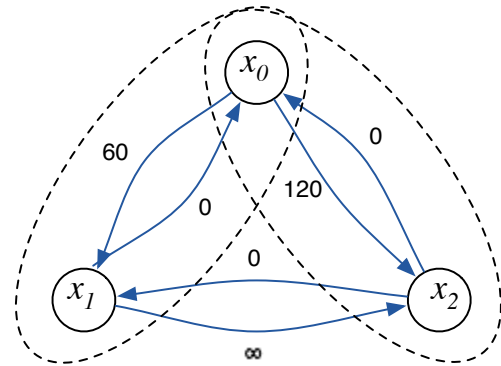


Fig. 2. Example of a z-partitioning used for temporal decoupling in Example 2.3.

In [7], [8] an algorithm is given that given a z-partition performs a suitable *minimal* temporal decoupling. This algorithm can be easily applied to STN specifications of the turnaround process to obtain sub-STNs, one per service provider. These decoupled STNs can be given to each of the service providers. Whatever solution is provided by a particular service provider,

it will never create an infeasibility, since merging the individual solutions will always provide a total solution.

3. Determining the number of resources needed

Once by temporal decoupling an independently schedulable temporal plan per agent (service provider) has been obtained, a service provider would like to choose an optimal schedule, for example a schedule that minimizes the number of resources needed in order to complete all the activities specified in his temporal plan. For example, agents such as the fueling or catering company, need vehicles as resources to service an aircraft. A service provider would be interested to use as few vehicles as possible by letting them service multiple aircraft if the activities take place after each other and there is sufficient time between them to travel from one aircraft to another. Hence, we need to take into account the *travelling time* between the activities to be performed by each provider.

To give an example, let us consider the sub STN (temporal plan) of the fueling agent. In such a decoupled STN the service provider can find the Earliest Starting Time ($EST(a)$), Latest Starting Time ($LST(a)$), Earliest End Time ($EET(a)$) and the Latest End Time ($LET(a)$) per fueling activity a , since for each such an activity the tightest constraints for the start and end time of a are specified. Table 1 shows an example of information derived from a decoupled temporal plan where for each flight to be handled it is indicated at what time the fueling agent must start (earliest) and when the agent must end (latest).

TABLE 1. Decoupled plans for fueling service

call sign	gate	EST	LST	EET	LET
KL1857	C06	12:00:00	12:00:00	12:12:00	13:28:31
KL1013	B13	12:10:00	12:10:00	12:22:00	13:39:37
KL1667	D87	12:14:25	12:14:25	12:45:25	13:59:31
KL1577	F04	12:25:23	12:25:23	12:56:23	14:10:37
.....

In order to determine the number of fueling vehicles for this fueling agent i , we model the set of activities specified in the decoupled temporal plans as the set of nodes V_i in a directed *reachability graph* $G_i = \langle V_i, E_i \rangle$, where there is a directed edge between two activities $a_1, a_2 \in V_i$ if it is possible to accomplish activity a_2 after a_1 using the same resource. That is, $(a_i, a_j) \in E_i$ if agent i , using the same vehicle, is able to complete activity a_2 without violating the time constraints after it has completed a_1 without violating its time constraints. Here, we might use two methods to determine whether a_2 is serviceable after a_1 : the *minimum* and the *maximum* method.

- minimum or *pessimistic* method:
 $(a_1, a_2) \in E_i$ iff $EET(a_1) + distance(a_1, a_2) \leq$

$LST(a_2)$

- maximum or *optimistic* method:

$$(a_1, a_2) \in E_i \text{ iff } LET(a_1) + distance(a_1, a_2) \leq EST(a_2)$$

Here, $distance(a_1, a_2)$ is the travel time (distances) between the gate where activity a_1 and the gate where activity a_2 has to be performed. In the next example we show how to construct such a graph G_i using the *maximum* method.

Example 3.1: We construct the graph $G_i = \langle V_i, E_i \rangle$ for a fueling agent i . First, we need its decoupled plan with a list of activities with all the flights and the EST, LST, EET and LET for the fueling service for these flights. Table 1 shows a part of this list. We also need a table with the travel times (distances) between gates. Using the maximum method, we check whether for every possible pair of activities a_1 and a_2 the constraint $LET(a_1) + distance(a_1, a_2) \leq EST(a_2)$ holds. If so, we add a directed edge (a_i, a_j) to G_i . If not, no edge is added. For example, checking for flight KL1857 and KL1577, suppose that the distance between these activities is 2 minutes travel time. From Table 1 and this distance we derive $13:28:31 + 00:02:00 > 12:25:23$. It follows that it is not possible to service these two flights after each other and consequently no edge is added. Using a full table of distances and applying this method to all pairs of services for agent i results in Figure 3.

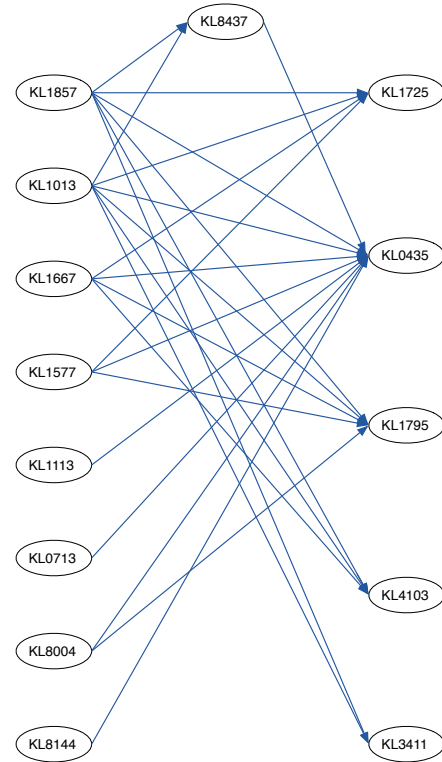


Fig. 3. Reachability graph G_i for the fueling agent i created with the maximum method. There is an edge between two nodes (flights) if these flights are serviceable after each other using the same vehicle.

In Figure 3 there is a path from flight KL1857 via KL8437 to KL0435. This means that only one resource is needed to service these three flights. In order to determine the minimum number of resources needed, we now have to find out how many paths we need if we want to cover all nodes exactly once². In the literature, this problem is called the *Minimal Node Disjoint Path Cover* problem (e.g., [10]). In general, this problem is intractable, because the decision variant (*Node Disjoint Path Cover*) is a special case of the NP-complete HAMPAD problem (which comes down to covering all nodes with one path). In our case, however, it is easy to see that the graph G_i is *acyclic*, which, as we will show, implies that the problem can be solved in polynomial time.

We can reduce the acyclic Node Disjoint Path Cover problem to the polynomially solvable *Maximum Flow* problem³ as follows: First, we construct a flow graph G_i^f from the graph G_i (as e.g., presented in Figure 3). Secondly, we prove that a solution of this *Maximum Flow* problem can be easily converted into a solution of our Minimal Node Disjoint path Cover problem, i.e., the minimum number of resources needed.

The flow graph $G_i^f = (V_i^f, E_i^f)$ is constructed from the directed graph $G_i = (V_i, E_i)$ as follows:

Let $V_i = \{x_1, x_2, \dots, x_n\}$.

- $V_i^f = \{s, x_1, x_2, \dots, x_n\} \cup \{y_1, y_2, \dots, y_n, t\}$, where s (the source) and t (the sink) are two nodes not occurring in V_i and for every $x_i \in V_i$, y_i is a new node in V_i^f ;
- $E_i^f = \{(s, x_i) : x_i \in V_i\} \cup \{(y_i, t) : x_i \in V_i\} \cup \{(x_i, y_j) : (x_i, x_j) \in E_i\}$.
- each edge $e \in E_i^f$ is given capacity 1.

Example 3.2: Performing this transformation on the graph G_i , we get the flow graph G_i^f depicted in Figure 4. The top row is called the X-row with the x_i nodes and the bottom row of nodes is the Y-row of copies y_i . Executing a Maximal Flow algorithm on G_i^f gives the maximum amount of flow from s to t in the graph and the edges through which it flows (See Figure 4, the edges in the maximal flow are marked bold). The flow f is a maximum $s - t$ flow with value 6. ■

We now show that, in general, the value of the maximal flow in G_i^f determines the solution specifying the minimum amount of resources needed (i.e. the solution to the minimum disjoint path cover problem).

Proposition 3.1: Given a reachability graph $G_i = (V, E)$ and the constructed flow graph $G_i^f = (V', E')$, let f_{max} be the value of the maximal $s - t$ flow of G_i^f . Then the minimum number of paths needed for a disjoint path cover of G_i is $|V| - f_{max}$.

2. Note that the graph is transitive: if there exists an edge between node a and b then, according to the maximal method, $a_{let} + distance(a, b) \leq b_{est}$ holds. If there also exists an edge (b,c) then $b_{let} + distance(b, c) \leq c_{est}$ also holds. By definition $b_{est} \leq b_{let}$, from which follows that $b_{est} + distance(b, c) \leq c_{est}$ and thus $a_{let} + distance(a, b) + distance(b, c) \leq c_{est}$ and $a_{let} + distance(a, b) \leq c_{est}$. This transitivity of the graph ensures that the requirement that each flight is serviced (fuelled) exactly once doesn't restrict the solution set.

3. For a description of this problem, see[3].

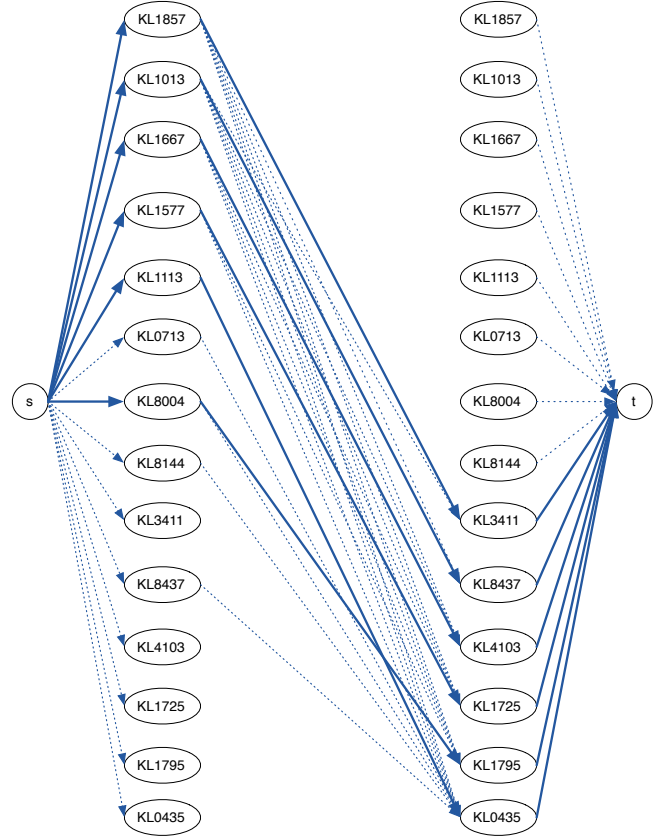


Fig. 4. Flow graph G_i^f constructed from the graph G_i in Figure 3

Proof: See Appendix. □

Example 3.3: Applying the Maximum Flow algorithm to our example in Figure 4, we find a maximum flow of 6. Thus, the minimum number of resources needed to carry out all activities is 8 (the number of activities (14) minus the value of the maximum flow (6)). ■

Note that in addition to the value of the disjoint path cover the algorithm also determines *which* flights can be serviced by the same resource. Each edge with positive flow is a part of a path. In Figure 4 these edges are marked bold. Constructing the paths is now easy. We take for each edge (x_i, y_j) with non-zero flow the edge (v_i, v_j) in G_i and add these to the path cover. Succeeding nodes are on the same path. Nodes not belonging to a path belong to their own path with length zero. In Figure 3 the paths found are the following ones:

- Paths with length 1:
 $\{(KL1857, KL3411)\}$, $\{(KL1013, KL8437)\}$,
 $\{(KL1667, KL4103)\}$, $\{(KL1577, KL1725)\}$,
 $\{(KL8004, KL1795)\}$, $\{(KL1113, KL0435)\}$
- Paths with length 0:
KL0713 and KL8114

In Figure 5 the number of resources needed is plotted against time. This figure demonstrates that the maximum capacity of

8 vehicles is only needed between 13:00 and 16:00 - not for the complete time period.

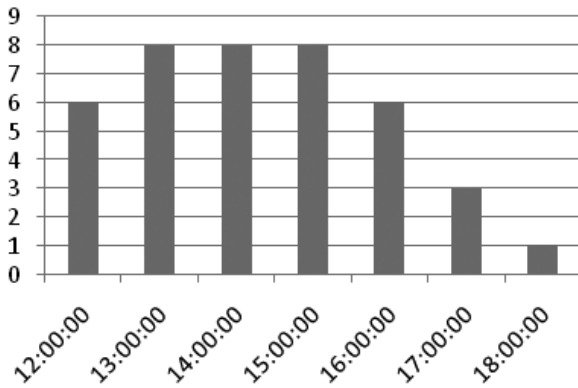


Fig. 5. Number of resources needed dependent upon time

As a final remark, note that the reachability graph G_i can be constructed using the maximum or the minimum method. This enables a service provider to come up with a pessimistic and an optimistic estimation of the number of vehicles (s)he has to use to perform all the activities.

4. Conclusions and Further Research

In this paper, a multi-agent approach has been presented to solve the planning of ground handling processes at airports. In particular, a temporal decoupling method has been introduced to partition the overall plan into several independent sub-plans that can be solved locally and merged again into a conflict-free plan. This approach offers one major advantage: it allows service providers at airports to plan their activities independently of other service providers as much as possible. This renders complex and time-consuming re-planning coordination between agents superfluous in case of small disruptions to the original plan. Given the large number of delays occurring daily at airports, this seems a valuable approach.

Based on the decoupling approach, a new algorithm has been presented to determine the number of resources needed to execute all activities in a decoupled service plan. Apart from the travelling time between activities, this algorithm also takes the earliest and latest start and end times of each activity into account. By doing so, the algorithm offers a minimum and maximum number of resources required to perform the service. The minimum may correspond to minimal costs for the service provider, the maximum to maximum service reliability through maximum flexibility in re-pairing plan disruptions. The service provider is thus given an upper and lower bound: the choice, depending on his preferences and business model, is his.

Another advantage of the algorithm is that it not only specifies the amount of resources needed, but also the time and place of their allocation. For example, Figure 4 shows that the minimum capacity of eight resources is only needed

for three hours; during the rest of the day, fewer resources will suffice. Other solutions with the same minimum capacity may exist and could be calculated. The time complexity of this algorithm is rather modest: it is bounded above by the complexity of the Maximum Flow algorithm and below by $O(n^2)$.⁴

Future research may focus on a number of extensions. First, resource determination may be made more realistic by including information about the availability of resources or travel routes. Fuelling vehicles may for instance become temporarily unavailable during the re-fuelling of these vehicles themselves.

Second, support may be given to the re-planning of resource allocation in case things go wrong. As a first step, an overview could be generated listing which extra resources may be available in case of disruptions. Further steps may be directed towards increasing the level of decoupling. The underlying idea here is to merge certain service providers (e.g., cleaning and catering) in case local re-planning is not feasible. In such a mechanism, the new decoupling combines the planning margins of both agents, thus increasing the total margin that can be used to re-plan their activities. Current research focuses on implementing this merge mechanism; other efforts address right-shifting and swapping resources in order to re-plan services.

After adding more realism to the current algorithm and implementing re-planning decision support, we intend to proceed towards an all-encompassing prototype. We then hope to validate this multi-agent prototype and its underlying ideas during real-time simulation test trials. In the near future, a decision support application based on this prototype may become operational at a real airport.

References

- [1] D. Böhme. Tactical departure management with the eurocontrol/dlr dman. In *Proceedings of the 6th USA/Europe Seminar on ATM R&D*, Baltimore, USA, 2005.
- [2] B. V. Cherkassy and A. V. Goldberg. On implementing push-relabel method for the maximum flow problem. In *Proceedings of the 4th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 157–171, London, UK, 1995. Springer-Verlag.
- [3] Th. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.
- [4] R. Dechter. *Constraint Processing*. Morgan Kaufmann Publishers, 2003.
- [5] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artif. Intell.*, 49(1-3):61–95, 1991.
- [6] Eurocontrol. Prediction of air traffic growth in europe. <http://www.centreforaviation.com/news/2009/03/25/eurocontrol-predicts-5-reduction-in-european-flights-in-2009—weak-growth-likely-in-2009> page1, 2009.
- [7] L. Hunsberger. Algorithms for a temporal decoupling problem in multi-agent planning. In *Eighteenth national conference on Artificial intelligence*, pages 468–475, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.
- [8] L. Hunsberger. *Group Decision Making and Temporal Reasoning*. PhD thesis, Cambridge, MA, USA, 2002.
- [9] IATA. Prediction that cargo traffic in the world will fall by 13%, passenger traffic will contract 5.7%. <http://www.iata.org/pressroom/pr/2009-03-24-01.htm>, 2009.

4. In [2] the fastest algorithms available up till now are presented.

- [10] J. M. Kleinberg. *Approximation algorithms for disjoint paths problems*. PhD thesis, 1996.
- [11] H. Oberheid and D. Söffker. Cooperative arrival management in air traffic control - a coloured petri net model of sequence planning. In *Applications and Theory of Petri Nets, Proceedings of the 29th International Conference on PETRI NETS*, pages 348–367, Berlin, Germany, 2008. Springer-Verlag.
- [12] P. van Leeuwen. Requirements and design document for LEONARDO’s collaborative decision making multi-agent system CDMMA. Technical Report NLR-TR-2003-653, National Aerospace Laboratory NLR, January 2004.
- [13] P. van Leeuwen, L. I. Oei, P. Buzing, and C. Witteveen. Adaptive temporal planning at airports. In *Proceedings of the International Multi-conference on Computer Science and Information Technology*, Wisla, Poland, 15 - 17 October, 2007.
- [14] Cheng-Lung Wu and Robert E. Caves. Modelling and optimization of aircraft turnaround time at airports. *Transportation Planning and Technology*, 27(1):47–66, 2004.

Appendix

Proposition A.1: Given a graph $G = (V, E)$ and the constructed flow graph $G' = (V', E')$, let f be the value of the maximal flow of G' . Then the minimum number of paths needed for a disjoint path cover of G is $|V| - f$.

Proof: We show that G' has a flow with value $|V| - k$ if and only if k is the size of the set of disjoint paths covering V . From this correspondence it follows immediately that *maximal* flows correspond to *minimum* disjoint path covers.

(\Leftarrow) Given graph G with a path cover consisting of k disjoint paths, there are k nodes as the starting point of a covering path and $|V| - k$ nodes which are not. Because the covering paths are node disjoint, each of these $|V| - k$ nodes must have a unique predecessor on the covering path at which they occur. We construct a flow graph G' according to the method described earlier. For each node v_{i+1} with predecessor v_i there is an edge in G' between x_i from the X-row and y_{i+1} from the Y-row over which one unit of flow can be pushed. Because each node has a unique predecessor, the capacity constraint holds because each node x_i has at most one outgoing flow and each y_{i+1} has at most one incoming flow. So G' has as flow of $|V| - k$.

(\Rightarrow) Consider flow of size $|V| - k$ in the graph G' constructed from graph $G = (V, E)$. It is not difficult to see that this implies that there are k disjoint paths from the source s to the sink t . Hence, there are k disjoint nodes (note that the edge capacity is 1) in the X-row from which 1 unit of flow flows to a unique node in the Y-row. From this follows that there are k nodes in G that are a successor of some node on a flow path, and $|V| - k$ that are not. Clearly, these $|V| - k$ nodes are the starting point of a covering path. Therefore, there are $|V| - k$ paths in the path cover for G . \square