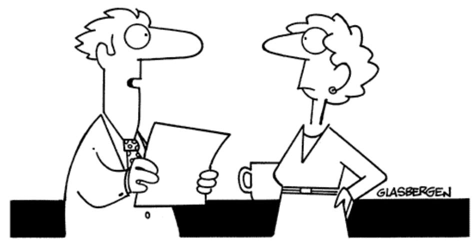
Temporal Constraints Techniques for Autonomous Scheduling

a play in 4 parts

Cees Witteveen,
Algorithmics group,
Delft University of Technology



"It took us five days to figure out how to finish our project two days early. That's why we're three days late."

Scheduling: basics

Given

- a set of *activities* (events) whose durations and resource requirements are known
- a set of temporal constraints between the activities
- and a given cost function

scheduling is about deciding *when* to perform *each activity* in a *cost optimal* way.

Scheduling: examples

project planning & scheduling
planning of activities to be executed in software projects

machine scheduling
allocation of jobs to processors

transportation scheduling
arrival & departure scheduling of flights

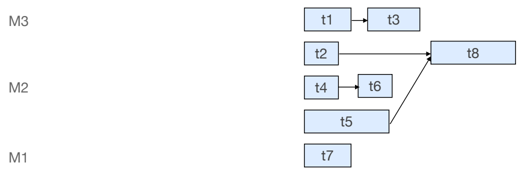
employee scheduling
crew rostering on flights

educational timetabling
timetables at schools and universities

assembly system scheduling
production planning for cars

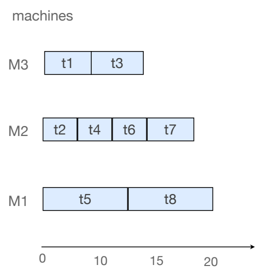
Example: machine scheduling

machines (resources) | events (tasks) with durations & precedences



assign jobs to the sequential machines such that the makespan (maximum completion time of a machine) is minimized.

Simple example



schedule realising a total completion time (makespan) of 20

can't do better in this case:
t8 has to be executed after t5 has been completed

Scheduling and constraints

an activity or event *e* is a process taking space and time.

we characterize events by *time point variables* and their constraints

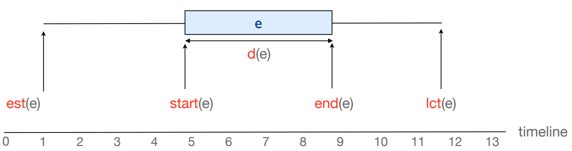
start(e) : starting time of event e

end(e) : ending time of event e

d(e) : duration of event e

est(e) : earliest starting time = min(start(e))

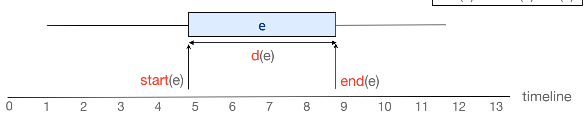
lct(e) : latest completion time = max(end(e))



preemptive vss non-preemptive

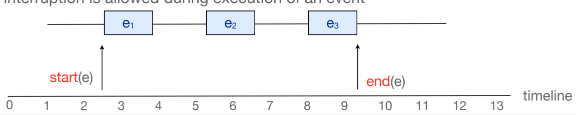
non-preemptive scheduling
no interruption is allowed during execution of an event

end(e) = start(e) + d(e)



preemptive scheduling
interruption is allowed during execution of an event

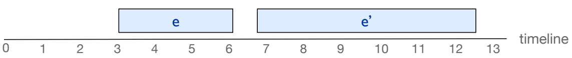
end(e) ≥ start(e) + d(e)



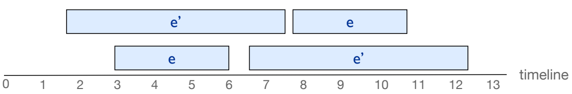
precedence constraints

sequencing of events
event e has to be completed before event e' can start, denoted by e ≺ e', if end(e) ≤ start(e')

end(e) ≤ start(e')



exclusive events:
e and e' are *exclusive* if e ≺ e' or e' ≺ e



performance measures

makespan:
the makespan F = C_{max} of a schedule is the completion time of the last event scheduled.

maximum tardiness
the maximum tardiness is the completion time of the last event scheduled minus the deadline given

average waiting time
the average waiting time is the sum of the starting times of all the jobs divided by the number of jobs

We are most interested in minimizing the makespan of schedules

How difficult is that?

scheduling:
some complexity results

Basic cases

Given a scheduling problem with
one machine (agent) and a set of jobs with *durations*
there is an efficient algorithm for computing the minimal makespan.

Given a scheduling problem with
two machines (agents) and a set of jobs with *durations*
there is *no efficient* algorithm for computing the minimal makespan, unless P=NP. (the problem is NP-hard)

Given a scheduling problem with
one machine (agent) and a set of jobs with *durations* + *precedence constraints*, there is no efficient algorithm for computing the minimal makespan, unless P=NP.

(will be extended)

Autonomous scheduling: the problem

The problem

Let E be a set of (time-constrained) events (activities) to be performed by several agents. Every activity *e* is given to exactly one agent A_i.

Every agent A_i is allowed to schedule its own set of activities $E_i \subseteq E$ *completely independent from the others.* Its schedule has to satisfy all the constraints belonging to the set E_i

The autonomous scheduling problem is the following problem:

How to ensure that the merging of all individual schedule is a schedule that is
- feasible, i.e., satisfies all the constraints
- makespan efficient

autonomous scheduling examples

airport planning

agent planning systems for arrival, departure, gate assignment, ground handling, taxi-route planning

how to provide a feasible total airport schedule?

patient schedules

individual scheduling of treatments of patients by doctors

how to ensure a feasible total patient schedule?

multi-modal logistics planning

company specific transportation planning systems

how to ensure a feasible intermodal transportation schedule?

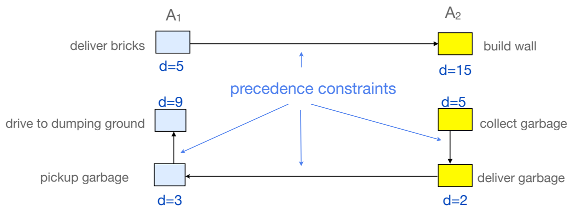
autonomous scheduling example

There are two agents A1 and A2 involved in a construction task.

A1 has to deliver bricks to A2, who has to use them to build a wall.

A2 has to ensure that garbage will be collected and has to deliver it to A1 who will pick it up and bring it to a dumping ground.

For each task t , the duration d(t) and the precedence relation with other tasks is known.



autonomous scheduling example

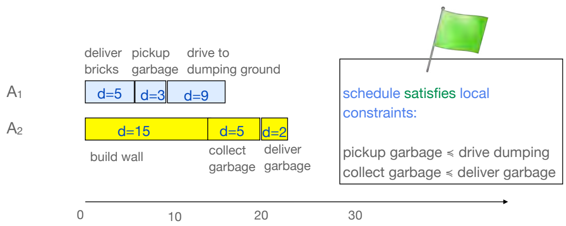
We would like to enable the agents to make their own schedule, starting from a given point in time *z=0.*

Let's look what might happen if only local constraints+ durations are given

autonomous scheduling example

We would like to enable the agents to make their own schedule, starting from a given point in time *z=0.*

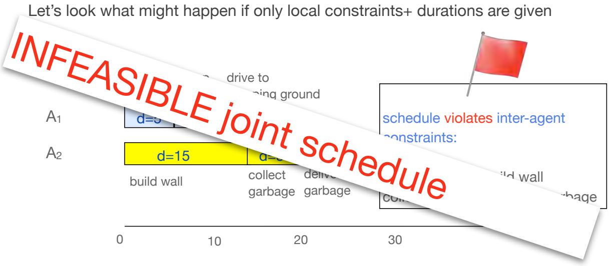
Let's look what might happen if only local constraints+ durations are given



autonomous scheduling example

We would like to enable the agents to make their own schedule, starting from a given point in time *z=0.*

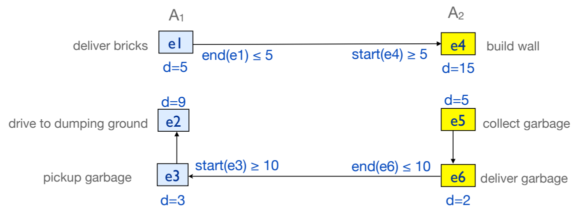
Let's look what might happen if only local constraints+ durations are given



autonomous scheduling example

To ensure a feasible joint schedule, we have to add a (weakest) set of additional constraints

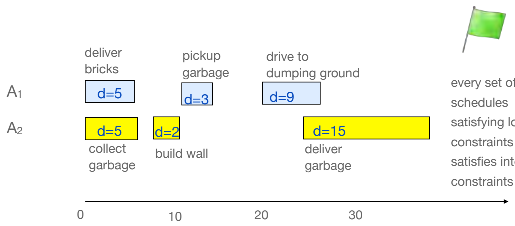
Let's look what might happen if only we add the following constraints to the given set of local constraints+ durations



autonomous scheduling example

We would like to enable the agents to make their own schedule, starting from a given point in time *z=0.*

Let's look what might happen if only local constraints+ durations are given

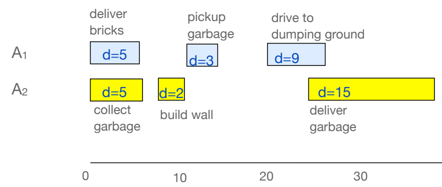


every set of schedules satisfying local constraints satisfies interagent constraints

autonomous scheduling example

We would like to enable the agents to make their own schedule, starting from a given point in time *z=0.*

Let's look what might happen if only local constraints+ durations are given



how to find a minimal set of additional constraints?
how to find a minimal set of additional constraints preserving makespan?

autonomous scheduling

First result

Given a set of events with precedence constraints and fixed durations, finding an arbitrary set of additional constraints ensuring conflict-free autonomous scheduling is easy.

Solution idea

First, take all tasks *e* such that there is no *e'* preceding *e*. Let E_1 the set of tasks obtained and d_1 the maximum duration of these tasks. Then, for every *e'* such that (i) there is an *e* in E_1, (ii) $e \ll e'$ and (iii) e and e' belong to different agents, add the constraints: end(*e*) ≤ d_1 and start(*e'*) ≥ d_1. Continue by obtaining E_2 and so on.

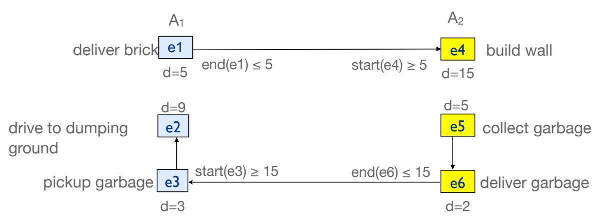
Exercise

Prove that using this idea needs to be refined to constitute a correct solution to the autonomous scheduling problem

autonomous scheduling

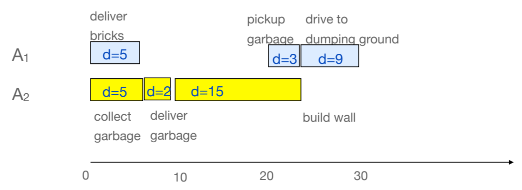
Example application

The makespan of the optimal schedule is max{5+15,5+2+3} = 20 = *M*
Adding the required constraints gives:



autonomous scheduling example

Resulting schedule



autonomous scheduling: variants

- basic variant
  Given a set E of events e with fixed durations d(e) and a precedence ordering ≼ on E
- extended variant
  Given a set E of events with *simple linear constraints* for start, end times and durations of events e
- full temporal constraint variant
  Given a set E of events with *arbitrary constraints* on start, end times and durations of events e.

organisation of the course

- dealing with the general case: the full temporal constraint variant
  We will start with an investigation of autonomous scheduling for problems represented by *general constraints*. This will allow us to characterize the complexity and to point out where the feasible cases are.
- extended variant; dealing with Simple Temporal Problems
  Then we discuss STPs and autonomous scheduling using the STP formalism. We will discuss a very general technique to solve autonomous scheduling problems here: *Temporal Decoupling* (TD)
- basic variant: specializing TD to the simple case
  We specialize the TD technique for STPs to our simple case showing a particularly simple algorithm for solving the autonomous scheduling problem.

Part II

Autonomous Scheduling

using

Constraint Systems

Dealing with the general case

We consider constraint systems with temporal variables, having values in a time domain. A solution to such a constraint system is an assignment of variables to time points, i.e., a schedule, satisfying all constraints.

The autonomous scheduling problem in its most general case then is a *constraint decomposition* problem:

Given a set of variables and constraints and a partitioning of the variables, how to ensure that solutions found for the partitioning induced subsystems constitute a joint solution to the complete system.

problem specification

distributed problems:

some problems require more than one party to solve them

each party will have to solve a separate part of the total problem;
parts might be interdependent

approach (decomposition + minimal change)

we are looking for methods to minimally change the problem specification in such a way that

(i) solutions are preserved

(ii) each party is able to solve its part independently from the others

(ii) individual solutions can be easily assembled to obtain a total solution.

decomposition: relevance

more efficient problem solving

concurrent decomposition would allow for *independent and concurrent solving* of *smaller* subproblems of a given problem.

autonomous computing

complete solutions can be obtained by autonomous 'private/local" computations without communicating partial results.

mechanism design

if a feasible total solution has to be obtained, decomposition guarantees that no solution strategy of an individual agent can affect the feasibility of the global solution.

constraints: some background

constraint systems

a constraint system is a tuple $\mathcal{S} = (X, D, C)$ where

$X = \{x_1, x_2, \dots x_n\}$ is a finite set of variables,

$D = \{D_i\}_{i=1}^n$ is a set of domains, where D_i is the domain of x_i,

C is a finite set of constraints for the variables in X.

examples

a set Φ of propositional formulae over a set of n atoms A is a constraint system $\mathcal{S} = (A, \{\{0,1\}_i\}_{i=1}^n, \Phi)$;

a system of linear equalities $Ax = b$ is a constraint system;

a schedule is a set of (temporal) constraints on a set of activities.

constraints: background (ii)

solution of a constraint system

a solution s of a constraint system $\mathcal{S} = (X, D, C)$ is an assignment of values $s(x_i) \in D_i$ to each variable $x_i \in X$ such that all constraints in C are satisfied.

the set of solutions of a constraint system $\mathcal{S}$ is denoted by $Sol(\mathcal{S})$

note that a solution s can also be represented as a *set of constraints* $\{x_i = s(x_i) : i = 1, 2, \dots, n\}$

subsystems generated by a set of vars

Let $\mathcal{S} = (X, D, C)$ be a constraint system and $X' \subseteq X$.
The subsystem generated by X' is $\mathcal{S}_{X'} = (X', D_{X'}, C_{X'})$ where

- $D_{X'}$ is the subset of domains for the variables in X'
- $C_{X'}$ is the subset of constraints mentioning only variables in X'

decomposition: current research

Decomposition in constraint systems is a technique to split a constraint system $\mathcal{S}$ into several parts $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_k$ such that

- solving each of the subsystems $\mathcal{S}_i$ is easy (or easier)
- a solution $s \in Sol(\mathcal{S})$ can be easily obtained by applying a poly-time computable function f (merger) to solutions $s_i \in Sol(\mathcal{S}_i)$

Several such techniques exist:
(hyper)tree decomposition, *hinge decomposition*, *query decomposition*, *tree clustering*

common aspects

- partitioning (covering) of variables is *byproduct* of the application of a decomposition technique
- result is set of *interrelated* acyclic subproblems, each of which can be solved efficiently
- decomposition does not result in splitting into *independently* solvable subproblems

decomposition: our approach

1. we take a *distributed* constraint system $\mathcal{S} = (\{X_i\}_{i=1}^n, D, C)$ where $\{X_i\}_{i=1}^n$ is a *given partitioning* of X.

2. we take the idea of decomposability to its extreme i.e., we require the subsystems $\mathcal{S}_i = (X_i, D_{X_i}, C_{X_i})$ to be *concurrently* and *independently* solvable such that arbitrary solutions of subsystems can always be joined to constitute a solution of $\mathcal{S}$.
We call such a distributed system *concurrently decomposable.*

3. if $\mathcal{S}$ is not concurrently decomposable, we would like to find constraint systems $\mathcal{S}'$ *closely related* to $\mathcal{S}$ such that
   - $\mathcal{S}'$ is concurrently decomposable,
   - there exists some polynomial p such that $|\mathcal{S}'| \leq p(|\mathcal{S}|)$
   - $Sol(\mathcal{S}') \subseteq Sol(\mathcal{S})$

concurrently decomposable systems

- concurrently decomposable systems

  $\mathcal{S} = (\{X_i\}_{i=1}^n, D, C)$ is concurrently decomposable if,

  - for all *i=1,2,...,n,* $\mathcal{S}_i = (X_i, D_{X_i}, C_{X_i})$
  - $Sol(\mathcal{S}_1) \times Sol(\mathcal{S}_2) \times \dots \times Sol(\mathcal{S}_n) \subseteq Sol(\mathcal{S})$

- simple consequence

  a constraint system $\mathcal{S}_i = (X_i, D_{X_i}, C_{X_i})$ is concurrently decomposable iff $\cup_{i=1}^n C_{X_i} \models C$.

complexity (i)

Deciding whether a distributed constraint system
$\mathcal{S} = (\{X_i\}_{i=1}^n, D, C)$ is concurrently decomposable, is coNP-complete.

proof
i) no-instances are easily verified.
ii) LOGICAL CONSEQUENCE can be easily reduced to the concurrent decomposability problem.
Take an instance *(U,C,c)* of LOGICAL CONSEQUENCE and consider a system *S* where the constraints are *C* ∪ *{c v x}* ∪ *{¬x}* and the partitioning of variables is *{U, {x}}*.

It follows that coNP-completeness already holds for distributed constraint systems where the partition contains only two blocks.

changing to a decomposable *S*

idea
if a distributed system $\mathcal{S}$ is not concurrently decomposable, change it to a new system $\mathcal{S}'$ such that $Sol(\mathcal{S}') \subseteq Sol(\mathcal{S})$.

example
Agent *A* has to choose between x, y and z (exclusive), while Agent *B* has to choose between y and z, and between u and v (also exclusive). Together only ≤ 2 objects can be chosen.

Independent choices cannot be made, as e.g. x, z and u might be chosen. An obvious solution is to restrict the choices of both *A* and *B* to choosing between y and z.

changing to a decomposable *S*

idea
if a distributed system $\mathcal{S}$ is not concurrently decomposable, change it to a new system $\mathcal{S}'$ such that $Sol(\mathcal{S}') \subseteq Sol(\mathcal{S})$

questions
How difficult is it to find a feasible alternative $\mathcal{S}'$?

How difficult is it to find an alternative $\mathcal{S}'$ that *comes as close* to the original system $\mathcal{S}$ as possible?

finding an alternative *S'*

arbitrary solution guaranteed (if system is consistent)

Consider a consistent $\mathcal{S} = (\{X_i\}_{i=1}^n, D, C)$. Then there exists a solution $s \in Sol(\mathcal{S})$;
Add the equality constraints $x_i = s(x_i)$ to C .
The resulting system will have s as its unique solution and every agent is forced to choose s_{Xi} as its solution

finding an alternative *S'*

finding a concurrently decomposable alternative is polynomially related to finding a solution for the original system:

Let $\mathbb{C}$ be a class of constraint systems.
Then there exists a polynomial algorithm to find a solution for every *S* in $\mathbb{C}$ iff
there exists a polynomial algorithm that, given *S* in $\mathbb{C}$ and an arbitrary partitioning of *X*, finds a decomposable alternative for *S*.

proof (sketch):
(=>) add solutions fund as equalities to constraints of system
(<=) take the finest partitioning of X; solutions to polynomially bounded sets of unary constraints can be found in polynomial time. So solve each of the parts of the system; decomposability guarantees composition.

finding a *minimal* solution

- semantical approach
  maximize the set of solutions $Sol(\mathcal{S}')$ subject to $Sol(\mathcal{S}') \subseteq Sol(\mathcal{S})$

  idea: find additional constraints to C such that $|Sol(\mathcal{S}) - Sol(\mathcal{S}')|$ is minimized.

- syntactical approach
  minimize the amount of change necessary to obtain the resulting system $\mathcal{S}'$

  idea: measure the difference in the size of the constraint sets of $\mathcal{S}$ and of $\mathcal{S}'$.

finding a minimal solution

- semantical approach
  maximize the set of solutions $Sol(\mathcal{S}')$ subject to $Sol(\mathcal{S}') \subseteq Sol(\mathcal{S})$

  idea: find additional constraints to C such that $|Sol(\mathcal{S}) - Sol(\mathcal{S}')|$ is minimized.

  NP-hard even if the cardinality of the set of solutions is poly-sized

- syntactical approach
  minimize the amount of change necessary to obtain the resulting system $\mathcal{S}'$

  idea: measure the difference in the size of the constraint sets of $\mathcal{S}$ and of $\mathcal{S}'$.

  Σ_2^p-complete even for the most simple distributed constraint systems

Part III

Autonomous scheduling using Simple Temporal Networks

We discuss the autonomous scheduling problem using a constraint system that allows for efficient solution finding.

Hence, using such a constraint system, the autonomous scheduling problem can also be solved efficiently.

The constraints allowed are differences t' - t ≤ d between time points bounded above by a constant.

simple temporal plans

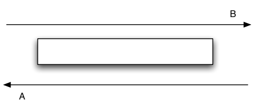
e — temporal event e

t (start of e) — t' (end of e) — time points

	direct	*binary*	*standard*
e starts at or after time 2	t ≥ 2	t - z_0 ≥ 2	z_0 - t ≤ -2
e ends at or before time 8	t' ≤ 8	t' - z_0 ≤ 8	t' - z_0 ≤ 8
e's duration is between 3 and 4	3 ≤ t' - t ≤ 4	t' - t ≤ 4	t' - t ≤ 4
		t' - t ≥ 3	t - t' ≤ -3
fixed time reference point	z_0 = 0		

Small example

Two trains A and B meet each other at a platform.
Train A arrives at 12.00 hrs and stops at most 5 minutes.
Train B must arrive at least 2 minutes later than A, and must leave within 3 minutes after A leaves.
B stays at most 7 minutes and at least 2 minutes.
When does A leave, B arrive and B leave?



Small example

Two trains A and B meet each other at a platform.
Train A arrives at 12.00 hrs and stops at most 5 minutes.
Train B must arrive at least 2 minutes later than A, and must leave within 3 minutes after A leaves.
B stays at most 7 minutes and at least 2 minutes.
When does A leave, B arrive and B leave?

time points

t_A, t_B : arrival of A, B — z_0 : time reference

t'_A, t'_B : departure of A, B

constraints

z_0 = 12.00 — $t_B - t_A \geq 2$

$0 \leq t_A - z_0 \leq 0$ — $t'_B - t'_A \leq 3$

$t'_A - t_A \leq 5$ — $2 \leq t'_B - t_B \leq 7$

transforming constraints to standard form
$t_j - t_i \leq a_{ij}$

Small example

Two trains A and B meet each other at a platform.
Train A arrives at 12.00 hrs and stops at most 5 minutes.
Train B must arrive at least 2 minutes later than A, and must leave within 3 minutes after A leaves.
B stays at most 7 minutes and at least 2 minutes.
When does A leave, B arrive and B leave?

time points

t_A, t_B : arrival of A, B z_0 : time reference

t'_A, t'_B : departure of A, B

constraints

$z_0 = 12.00$ $t_A - t_B \le -2$

$0 \le t_A - z_0 \le 0$ $t'_B - t'_A \le 3$

$t'_A - t_A \le 5$ $t'_B - t_B \le 7$ $t_B - t'_B \le -2$

This is the standard representation in
Simple
Temporal
Plans

Small example

Two trains A and B meet each other at a platform.
Train A arrives at 12.00 hrs and stops at most 5 minutes.
Train B must arrive at least 2 minutes later than A, and must leave within 3 minutes after A leaves.
B stays at most 7 minutes and at least 2 minutes.
When does A leave, B arrive and B leave?

time points

t_A, t_B : arrival of A, B z_0 : time reference

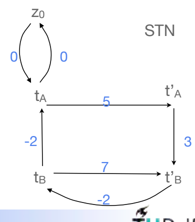
t'_A, t'_B : departure of A, B

constraints

$z_0 = 12.00$ $t_A - t_B \le -2$

$0 \le t_A - z_0 \le 0$ $t'_B - t'_A \le 3$

$t'_A - t_A \le 5$ $t'_B - t_B \le 7$ $t_B - t'_B \le -2$



Simple Temporal Plans

A Simple Temporal Plan (STP) is a tuple S = (T, C)
where
- T is a set of time-points and
- C is a set of binary constraints of the form $t' - t \le \delta$

Simple Temporal Plan

S = (T, C)

$T = \{z_0, t_A, t_B, t'_A, t'_B\}$

$C = \{ z_0 = 12.00,\ t_A - t_B \le -2,\ 0 \le t_A - z_0 \le 0,\ t'_A - t_A \le 5,\ t'_B - t'_A \le 3,\ t'_B - t_B \le 7,\ t_B - t'_B \le -2 \}$

Simple Temporal Plans

An STN S =(T,C) can also be represented by a labeled directed graph
$G_S = (T, E, l)$
called a Simple Temporal Network (STN)
Nodes are time points, labeled edges are derived from constraints: $t' - t \le \delta \leftrightarrow t \xrightarrow{\delta} t'$

Simple Temporal Plan

S = (T, C)

$T = \{z_0, t_A, t_B, t'_A, t'_B\}$

$C = \{ z_0 = 12.00,\ t_A - t_B \le -2,\ 0 \le t_A - z_0 \le 0,\ t'_A - t_A \le 5,\ t'_B - t'_A \le 3,\ t'_B - t_B \le 7,\ t_B - t'_B \le -2 \}$

Simple Temporal Plans

An STN S =(T,C) can also be represented by a labeled directed graph
$G_S = (T, E, l)$
called a Simple Temporal Network (STN) or Distance Graph
Nodes are time points, labeled edges are derived from constraints: $t' - t \le \delta \leftrightarrow t \xrightarrow{\delta} t'$

Simple Temporal Plan

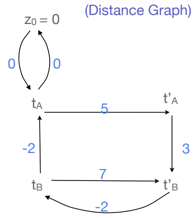
S = (T, C)

$T = \{z_0, t_A, t_B, t'_A, t'_B\}$

$C = \{ z_0 = 12.00,\ t_A - t_B \le -2,\ 0 \le t_A - z_0 \le 0,\ t'_A - t_A \le 5,\ t'_B - t'_A \le 3,\ t'_B - t_B \le 7,\ t_B - t'_B \le -2 \}$

Simple Temporal Network

(Distance Graph)

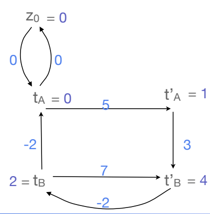


STN: solution & consistency

A *solution* to an STN $S = (T, C)$ is an assignment $\{ t_i = v_i : t_i \in T \}$ of values to variables that satisfies all constraints.

The set of solutions of S is denoted as $Sol(S)$

S is consistent iff $Sol(S) \ne \emptyset$ S equivalent to S' iff $Sol(S) = Sol(S')$



Solution 1:

$z_0 = 0$
$t_A = 0$
$t'_A = 1$
$t_B = 2$
$t'_B = 4$

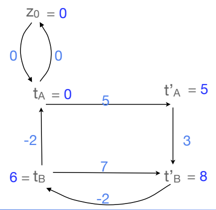
how to determine consistency....

STN: solution & consistency

A *solution* to an STN $S = (T, C)$ is an assignment $\{ t_i = v_i : t_i \in T \}$ of values to variables that satisfies all constraints.

The set of solutions of S is denoted as $Sol(S)$

S is consistent iff $Sol(S) \neq \emptyset$ S equivalent to S' iff $Sol(S) = Sol(S')$



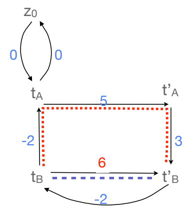
	Solution 1:	Solution 2:
z_0	= 0	= 0
t_A	= 0	= 0
t'_A	= 1	= 5
t_B	= 2	= 6
t'_B	= 4	= 8

how to find solutions….

Distance Graph

explicit constraints can be combined to imply other constraints:

constraints	some implied constraints
$t'_A - t_A \leq 5$ $t_A - t_B \leq -2$ $t_B - t'_B \leq -2$ $t'_B - t'_A \leq 3$ $t'_B - t_B \leq 7$	$t'_B - t_B \leq (t'_B - t'_A) + (t'_A - t_A) + (t_A - t_B)$ $\leq 3 + 5 + -2$ $= 6$



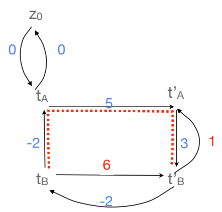
Look at the labels of the edges as *distances* between the corresponding nodes.

Finding a tighter constraint comes down to finding a shorter path.

Distance Graph

explicit constraints can be combined to imply other constraints:

constraints	some implied constraints
$t'_A - t_A \leq 5$ $t_A - t_B \leq -2$ $t_B - t'_B \leq -2$ $t'_B - t'_A \leq 3$ $t'_B - t_B \leq 7$	$t'_B - t_B \leq (t'_B - t'_A) + (t'_A - t_A) + (t_A - t_B)$ $\leq 3 + 5 + -2$ $= 6$ $t'_A - t'_B \leq (t'_A - t_A) + (t_A - t_B) + (t_B - t'_B)$ $\leq 5 + -2 + -2$ $= 1$



	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	6	8
t_A	0	0	5	6	8
t'_A	-1	-1	0	1	3
t_B	-2	-2	3	0	6
t'_B	-4	-4	1	-2	0

strongest constraints in STP correspond to *shortest* paths in STN

table of all (strongest) implied constraints

Distance Matrix

Strongest constraints in STP correspond to *shortest* paths in STN.

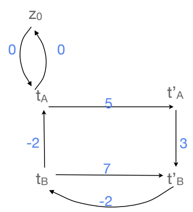
The distance matrix D_S for an STN $S = (T, C)$ is defined as:

$D(t_i, t_j)$ = length of shortest path from t_i to t_j in the distance graph of S or ∞ if no such path exists

Given the distance graph G_S there are well-known efficient algorithms to compute the distance matrix.

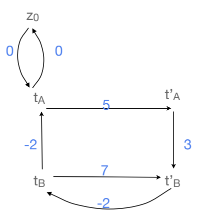
Example: all-pairs shortest path algorithm Floyd-Warshall: $O(n^3)$

G_S :



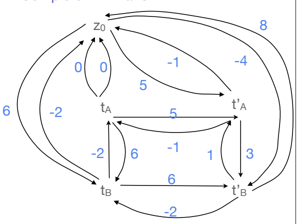
D( ) :

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	6	8
t_A	0	0	5	6	8
t'_A	-1	-1	0	1	3
t_B	-2	-2	3	0	6
t'_B	-4	-4	1	-2	0



equivalent systems

Complete minimal STN



	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	∞	∞
t_A	0	0	5	6	8
t'_A	∞	∞	0	∞	3
t_B	∞	-2	∞	0	7
t'_B	∞	∞	∞	-2	0

all-pairs shortest path computation

D:

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	6	8
t_A	0	0	5	6	8
t'_A	-1	-1	0	1	3
t_B	-2	-2	3	0	6
t'_B	-4	-4	1	-2	0

checking consistency

Given an STN S with its graph G_s and distance graph D, the following statements are equivalent:

- S is consistent
- D has only 0's on its diagonal
- G has no negative cycles

this property can be checked in $O(n^3)$

original constraint matrix

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	∞	∞
t_A	0	0	5	6	8
t'_A	∞	∞	0	∞	3
t_B	∞	-2	∞	0	7
t'_B	∞	∞	∞	-2	0

all-pairs shortest path computation $O(n^3)$

S is consistent

distance matrix D = [d(t,t')]

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	6	8
t_A	0	0	5	6	8
t'_A	-1	-1	0	1	3
t_B	-2	-2	3	0	6
t'_B	-4	-4	1	-2	0

finding solutions

Given an STN S with its distance graph D, if D does not contain negative diagonal elements,

- the set { $t_i = D(z_0, t_i)$: i=1,2, … n} ∪ {$z_0 = 0$} is a solution
- the set { $t_i = -D(t_i, z_0)$: i=1,2, … n} ∪ {$z_0 = 0$} is a solution

Why

original constraint matrix

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	∞	∞
t_A	0	0	5	6	8
t'_A	∞	∞	0	∞	3
t_B	∞	-2	∞	0	7
t'_B	∞	∞	∞	-2	0

all-pairs shortest path computation $O(n^3)$

set of solutions

distance matrix D = [d(t,t')]

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	6	8
t_A	0	0	5	6	8
t'_A	-1	-1	0	1	3
t_B	-2	-2	3	0	6
t'_B	-4	-4	1	-2	0

finding solutions

Given an STN S with its distance graph D, if D does not contain negative diagonal elements,

- the set { $t_i = D(z_0, t_i)$: i=1,2, … n} ∪ {$z_0 = 0$} is a solution
- the set { $t_i = -D(t_i, z_0)$: i=1,2, … n} ∪ {$z_0 = 0$} is a solution

Why

Take an arbitrary constraint $t' - t \leq \delta$.
We have $D(z_0,t') \leq D(z_0,t)+D(t,t') \leq D(z_0,t) + \delta$
Hence, $D(z_0,t') - D(z_0,t) \leq \delta$.
So the first set of solutions satisfies every constraint. Likewise,
$D(t,z_0) \leq D(t,t') + D(t',z_0) \leq \delta + D(t',z_0)$
Hence, $-D(t',z_0) + -D(t,z_0) \leq \delta$
So the second set of solutions satisfies every constraint.

distance matrix D = [d(t,t')]

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	6	8
t_A	0	0	5	6	8
t'_A	-1	-1	0	1	3
t_B	-2	-2	3	0	6
t'_B	-4	-4	1	-2	0

general recipe for solutions

sol := ∅
while *T* ≠ ∅
  select some *v* ∈ *[-D(t,z₀), D(z₀,t)]*
  add constraint *{t = v}* to STN and update the distance matrix *D*;
  remove row and column *t* from *D;*
  sol := sol ∪ {t=v}

start with $t_A \in [0,0]$; select $t_A = 0$

distance matrix D = [D(t,t')]

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	~~0~~	5	6	8
t_A	~~0~~	~~0~~	~~5~~	~~6~~	~~8~~
t'_A	-1	~~-1~~	0	1	3
t_B	-2	~~-2~~	3	0	6
t'_B	-4	~~-4~~	1	-2	0

general recipe for solutions

sol := ∅
while *T* ≠ ∅
  select some *v* ∈ *[-D(t,z₀), D(z₀,t)]*
  add constraint *{t = v}* to STN and update the distance matrix *D*;
  remove row and column *t* from *D;*
  sol := sol ∪ {t=v}

start with $t_A \in [0,0]$; select $t_A = 0$
take $t_B \in [2, 6]$; select $t_B = 4$; update

distance matrix D = [D(t,t')]

	z_0	t_A	t'_A	t_B	t'_B
z_0	0		5	4	8
t_A					
t'_A	-1		0	1	3
t_B	-4		3	0	6
t'_B	-4		1	-2	0

general recipe for solutions

sol := ∅
while *T* ≠ ∅
  select some *v* ∈ *[-D(t,z₀), D(z₀,t)]*
  add constraint *{t = v}* to STN and update the distance matrix *D*;
  remove row and column *t* from *D;*
  sol := sol ∪ {t=v}

start with $t_A \in [0,0]$; select $t_A = 0$
take $t_B \in [2, 6]$; select $t_B = 4$; update

distance matrix D = [D(t,t')]

	z_0	t_A	t'_A	t_B	t'_B
z_0	0		5		8
t_A					
t'_A	-3		0		3
t_B					
t'_B	-6		-1		0

general recipe for solutions

sol := ∅
while *T* ≠ ∅
  select some *v* ∈ *[-D(t,z₀), D(z₀,t)]*
  add constraint *{t = v}* to STN and update the distance matrix *D*;
  remove row and column *t* from *D;*
  sol := sol ∪ {t=v}

start with $t_A \in [0,0]$; select $t_A = 0$
take $t_B \in [2, 6]$; select $t_B = 4$; update
take $t'_A \in [3, 5]$; select $t'_A = 3$; update

distance matrix D = [D(t,t')]

	z_0	t_A	t'_A	t_B	t'_B
z_0	0		3		8
t_A					
t'_A	-3		0		3
t_B					
t'_B	-6		-1		0

general recipe for solutions

sol := ∅
while $T \neq \emptyset$
  select some $v \in [-D(t,z_0), D(z_0,t)]$
  add constraint $\{t = v\}$ to STN and update the distance matrix D;
  remove row and column t from D;
  sol := sol ∪ {t=v}

take $t_A \in [0, 0]$; select $t_A = 0$; update
take $t_B \in [2, 6]$; select $t_B = 4$; update
take $t'_A \in [3, 5]$; select $t'_A = 3$; update
take $t'_B \in [6, 6]$; select $t'_B = 6$

distance matrix D = [D(t,t')]

	z_0	t_A	t'_A	t_B	t'_B
z_0	0				6
t_A					
t'_A					
t_B					
t'_B	-6				0

adding constraints to S

Let D be a distance matrix of a consistent STN S.
Let c: t - t' ≤ δ be a new constraint and S' = S + c.

Then
- S' is consistent if δ ∈ [-D(t',t), D(t,t')]
- S' is inconsistent if δ < -D[t',t]

D

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	6	8
t_A	0	0	5	6	8
t'_A	-1	-1	0	1	3
t_B	-2	-2	3	0	6
t'_B	-4	-4	1	-2	0

add $t'_A - t_A \leq 0$

S' is inconsistent:
$0 \leq 1$

	z_0	t_A	t'_A	t_B	t'_B
z_0	-∞	-∞	-∞	-∞	-∞
t_A	-∞	-∞	-∞	-∞	-∞
t'_A	-∞	-∞	-∞	-∞	-∞
t_B	-∞	-∞	-∞	-∞	-∞
t'_B	-∞	-∞	-∞	-∞	-∞

adding constraints to S

Let D be a distance matrix of a consistent STN S.
Let c: t - t' ≤ δ be a new constraint and S' = S + c.

Then
- S' is consistent if δ ∈ [-D(t',t), D(t,t')]
- S' is inconsistent if δ < -D[t',t]

D

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	6	8
t_A	0	0	5	6	8
t'_A	-1	-1	0	1	3
t_B	-2	-2	3	0	6
t'_B	-4	-4	1	-2	0

add $t'_A - t_A \leq 2$

S' is inconsistent:
$0 \leq 1$

consistent

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	2	3	5
t_A	0	0	2	3	5
t'_A	-1	-1	0	1	3
t_B	-2	-2	0	0	3
t'_B	-4	-4	-2	-2	0

temporal decoupling

In finding solutions, we assumed that one agent controls the assignment of values to all the variables.

What happens if there is more than one agent and every agent A_i controls a disjoint subset T_i of time-point variables?

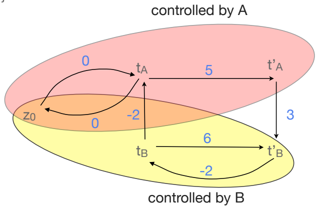
Each of these agents A_i wants to determine a local solution to the sub-STN S_i generated by its set T_i.

We are facing a major problem here

temporal decoupling: example

Consider the train example discussed before.
Train A and Train B would like to determine their schedule independently from each other.

Suppose:
A chooses $Sol(S_A) \ni \{ t_A = 0, t'_A = 2 \}$
B chooses $Sol(S_B) \ni \{ t_B = 3, t'_B = 6 \}$



temporal decoupling: example

Consider the train example discussed before.
Train A and Train B would like to determine their schedule independently from each other.

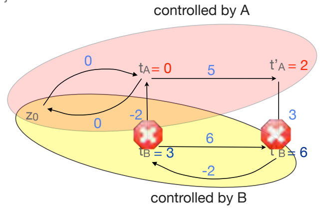
Suppose:
A chooses $Sol(S_A) \ni \{ t_A = 0, t'_A = 2 \}$
B chooses $Sol(S_B) \ni \{ t_B = 3, t'_B = 6 \}$

then

$Sol(S_A) \cup Sol(S_B) \ni$
$\{ t_A = 0, t'_A = 2 , t_B = 3, t'_B = 6 \}$

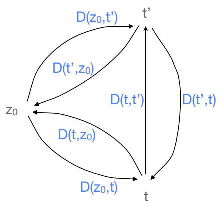
is not a solution of S !

intuitive reason for failure:
inter-agent constraints are not implied by local constraints



temporal decoupling: method

Take an arbitrary constraint $t' - t \leq D(t,t')$ such that t and t' belong to different blocks.
Consider the intra-agent constraints $t' - z_0 \leq D(z_0,t')$, $z_0 - t \leq D(t,z_0)$.



If $D(t,z_0) + D(z_0,t') > D(t,t')$ then $t' - t \leq D(t,t')$ is *not implied* by $t' - z_0 \leq D(z_0,t')$, $z_0 - t \leq D(t,z_0)$

We can ensure implication of $t' - t \leq D(t,t')$ by intra-agent constraints by *tightening* $t' - z_0 \leq D(z_0,t')$ and $z_0 - t \leq D(t,z_0)$

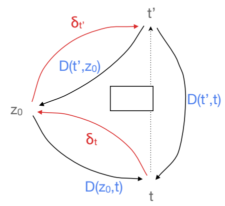
Method: Choose δ_t and $\delta_{t'}$ such that

1. $-D(z_0,t) \leq \delta_t < D(t,z_0)$
2. $-D(t,z_0) \leq \delta_{t'} < D(z_0,t')$
3. $\delta_t + \delta_{t'} \leq D(t,t')$

Add constraints $t' - z_0 \leq \delta_t$ and $z_0 - t \leq \delta_t$ to S compute new distance matrix.

temporal decoupling: method

Take an arbitrary constraint $t' - t \leq D(t,t')$ such that t and t' belong to different blocks.
Consider the intra-agent constraints $t' - z_0 \leq D(z_0,t')$, $z_0 - t \leq D(t,z_0)$.



If $D(t,z_0) + D(z_0,t') > D(t,t')$ then $t' - t \leq D(t,t')$ is *not implied* by $t' - z_0 \leq D(z_0,t')$, $z_0 - t \leq D(t,z_0)$

We can ensure implication of $t' - t \leq D(t,t')$ by intra-agent constraints by *tightening* $t' - z_0 \leq D(z_0,t')$ and $z_0 - t \leq D(t,z_0)$

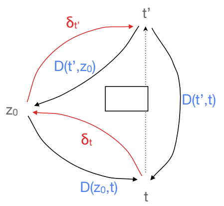
Method: Choose δ_t and $\delta_{t'}$ such that

1. $-D(z_0,t) \leq \delta_t < D(t,z_0)$
2. $-D(t,z_0) \leq \delta_{t'} < D(z_0,t')$
3. $\delta_t + \delta_{t'} \leq D(t,t')$

Add constraints $t' - z_0 \leq \delta_{t'}$ and $z_0 - t \leq \delta_t$ to S compute new distance matrix.

temporal decoupling: method

Take an arbitrary constraint $t' - t \leq D(t,t')$ such that t and t' belong to different blocks.
Consider the intra-agent constraints $t' - z_0 \leq D(z_0,t')$, $z_0 - t \leq D(t,z_0)$.

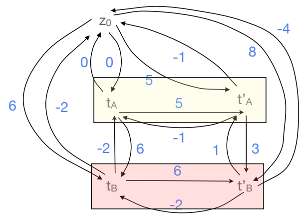


If $D(t,z_0) + D(z_0,t') > D(t,t')$ is *implied* by $t' - z_0 \leq \delta_{t'}$ and $z_0 - t \leq \delta_t$ it can be removed from S without any consequence.

This procedure can be repeated for every inter agent constraint not implied by intra-agent constraints.

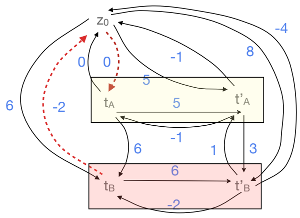
The resulting system is a decoupled STN

Temporal Decoupling in action



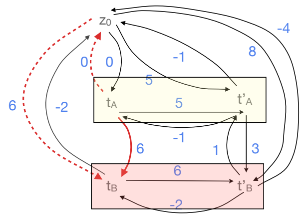
Step 1: ensure that $t_A - t_B \leq -2$

Temporal Decoupling in action



Step 1: ensure that $t_A - t_B \leq -2$.
This constraint is implied.
remove it.

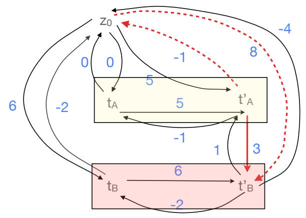
Temporal Decoupling in action



Step 1: take $t_A - t_B \leq -2$.
This constraint is implied.
remove it.

Step 2: take $t_B - t_A \leq 6$.
This constraint is implied.
remove it.

Temporal Decoupling in action

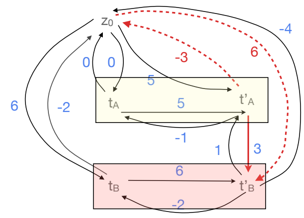


Step 1: take $t_A - t_B \leq -2$.
This constraint is implied.
remove it.

Step 2: take $t_B - t_A \leq 6$.
This constraint is implied.
remove it.

Step 3: take $t'_B - t'_A \leq 3$.
This constraint is not implied.
Take $z_0 - t'_A \leq -3$ and $t'_B - z_0 \leq 6$.
Then $t'_B - t'_A \leq 3$ is implied.
Update D.

Temporal Decoupling in action

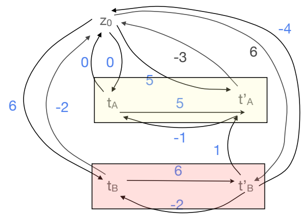


Step 1: take $t_A - t_B \leq -2$.
This constraint is implied.
remove it.

Step 2: take $t_B - t_A \leq 6$.
This constraint is implied.
remove it.

Step 3: take $t'_B - t'_A \leq 3$.
This constraint is not implied.
Take $z_0 - t'_A \leq -3$ and $t'_B - z_0 \leq 6$.
Then $t'_B - t'_A \leq 3$ is implied.
Update D and remove $t'_B - t'_A \leq 3$

Temporal Decoupling in action

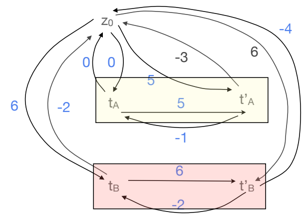


Step 1: take $t_A - t_B \leq -2$.
This constraint is implied.
remove it.

Step 2: take $t_B - t_A \leq 6$.
This constraint is implied.
remove it.

Step 4: take $t'_A - t_B \leq 1$.
This constraint is implied.
remove it.

Temporal Decoupling in action



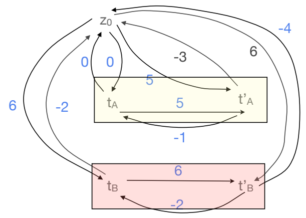
Step 1: take $t_A - t_B \leq -2$.
This constraint is implied.
remove it.

Step 2: take $t_B - t_A \leq 6$.
This constraint is implied.
remove it.

Step 4: take $t'_A - t_B \leq 1$.
This constraint is implied.
remove it.

The constraints between t_A and t'_B and t'_A and t_B are treated analogously

Temporal Decoupling in action



For decoupled systems, local solutions always imply a global solution: every combination of local solutions is a global solution.

(every inter-agent constraint is satisfied by any combination of local solutions)

Example:
$\{t_A = 0, t'_A = 4\}$ and $\{t_B = 3, t'_B = 5\}$ satisfies all constraints.

Temporal decoupling in STNs

a temporal decoupling *S* is said to be *locally optimal* if there does not exists an alternative decoupling *S'* where

- all the constraints are at most as *tight* as the corresponding constraints in S and
- at least one constraint is less *tight* than the corresponding constraint in *S*

Hunsberger presented an algorithm for obtaining *locally optimal* decouplings.

We will present an algorithm for obtaining a *globally optimal* temporal decoupling minimizing the *total tightness* of the constraints.

Finding an optimal TD

Idea:

given a distributed STN $\mathcal{S} = (\{X_i\}_{i=1}^{n}, D, C)$ we associate to it an LP problem having the variables p_{xy} for every $x,y \in X$.

These variables encode the upper bounds of the constraints in the resulting decoupled STN: $x - y \leq p_{xy}$ and therefore completely specify the decomposable alternative.

The linear (in)equalities encode the conditions each of the p_{xy} and have to satisfy:

- the minimal STN conditions
- the consistency conditions
- the temporal decoupling conditions

Finding an optimal TD

Consider the following linear constraints for the variables p_{xy}

$\forall u,w,v \in X$: $p_{vx} \leq p_{vw} + p_{wx}$ *(minimality conditions)*

$\forall x \in X$: $p_{xx} = 0$ *(consistency)*

$\forall\ x - y \leq w_{yx} \in C$: $p_{yx} \leq w_{yx}$ *(preserving solutions)*

$\forall\ x - y \leq w_{yx} \in C$ s.t.
x and y in different partitions:
$p_{yz} + p_{zx} \leq w_{yx}$ *(temporal decoupling)*

Consider the following objective function:

maximize $\Sigma_{X_i} \Sigma_{x,y \in X_i}\ p_{xy}$

The solution of this LP will return an optimal decoupled STN

Finding an optimal TD (ii)

Whenever the objective function is *linear*, an optimal solution for the optimal temporal decoupling problem can be obtained in polynomial time.

For some quadratic objective functions, the solution cannot be obtained in polynomial time unless P = NP.

(Vertex cover can be encoded in LP with a quadratic objective function)

Part IV

Applications of Temporal Decoupling

The Turnaround Process

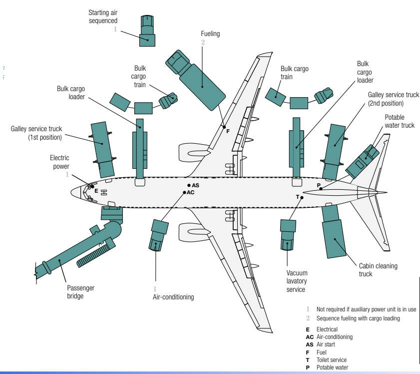
At airports, planes have to be serviced at the gates.

Typically, multiple services have to be applied (baggage handling, container loading, catering, fueling, lavatory services, cleaning) by different independent service providers.

For each aircraft unique durations and dependencies of these services are known.

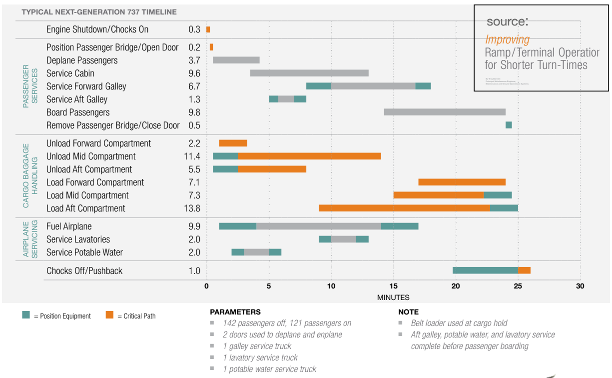
There is a daily specification of the schedule for this turnaround process, on a mid-size to large airport requiring to service a few *hundred* to a few *thousand* planes.

Turnaround servicing arrangement



source:
Improving Ramp/Terminal Operatior for Shorter Turn-Times

Turnaround timeline (Boeing 737)



source:
Improving Ramp/Terminal Operation for Shorter Turn-Times

The problem

Given a daily turnaround plan and a set of autonomous service providers

- *How to obtain for each ground handling agent a temporal plan such that it can schedule its activities independently of the others, whilst the feasibility of the overall solution is ensured.*
- *How to provide each agent with an estimation of the number of resources it minimally needs to carry out the activities listed in its schedule.*

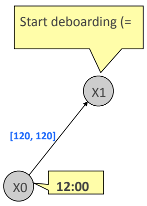
The problem

Given a daily turnaround plan and a set of autonomous service providers

- *How to obtain for each ground handling agent a temporal plan such that it can schedule its activities independently of the others, whilst the feasibility of the overall solution is ensured.* TEMPORAL DECOUPLING
- *How to provide each agent with an estimation of the number of resources it minimally needs to carry out the activities listed in its schedule.* SPECIALISED ALGORITHM

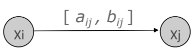
Solving the decoupling problem

modeling the turnaround process



$$a_{ij} \leq x_j - x_i \leq b_{ij}$$

is represented as



turnaround constraints

on-block+start deboarding

at 14.00 o'clock

120 ≤ X1 – X0 ≤ 120

deboarding: 5-10 min

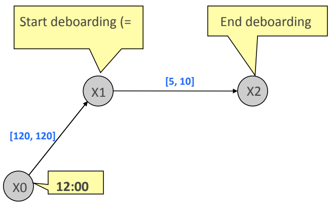
fuelling: 10-20 min

fuelling after deboarding

turnaround: 60 min

off-block ≥30 after fuelling

modeling the turnaround process



turnaround constraints

on-block+start deboarding

at 14.00 o'clock

120 ≤ X1 – X0 ≤ 120

deboarding: 5-10 min

5 ≤ X1 – X2 ≤ 10

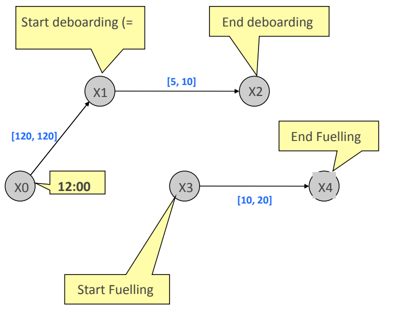
fuelling: 10-20 min

fuelling after deboarding

turnaround: 60 min

off-block ≥30 after fuelling

modeling the turnaround process



turnaround constraints

on-block+start deboarding at 14.00 o'clock

$120 \leq X1 - X0 \leq 120$

deboarding: 5-10 min

$5 \leq X1 - X2 \leq 10$

fuelling: 10-20 min

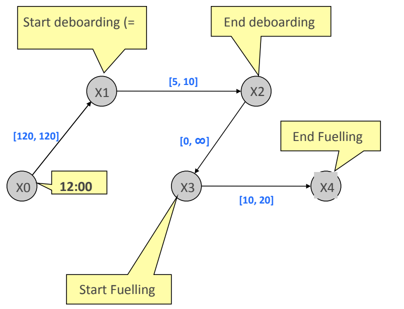
$10 \leq X4 - X3 \leq 20$

fuelling after deboarding

turnaround: 60 min

off-block ≥30 after fuelling

modeling the turnaround process



turnaround constraints

on-block+start deboarding at 14.00 o'clock

$120 \leq X1 - X0 \leq 120$

deboarding: 5-10 min

$5 \leq X1 - X2 \leq 10$

fuelling: 10-20 min

$10 \leq X4 - X3 \leq 20$

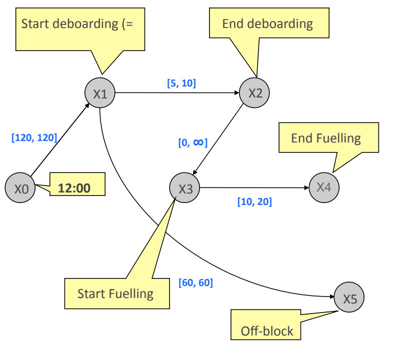
fuelling after deboarding

$0 \leq X3 - X2 \leq \infty$

turnaround: 60 min

off-block ≥30 after fuelling

modeling the turnaround process



turnaround constraints

on-block+start deboarding at 14.00 o'clock

$120 \leq X1 - X0 \leq 120$

deboarding: 5-10 min

$5 \leq X1 - X2 \leq 10$

fuelling: 10-20 min

$10 \leq X4 - X3 \leq 20$

fuelling after deboarding

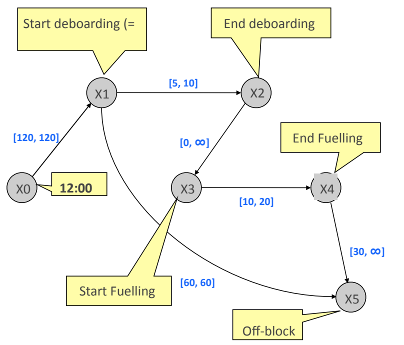
$0 \leq X3 - X2 \leq \infty$

turnaround: 60 min

$60 \leq X5 - X1 \leq 60$

off-block ≥30 after fuelling

modeling the turnaround process



turnaround constraints

on-block+start deboarding at 14.00 o'clock

$120 \leq X1 - X0 \leq 120$

deboarding: 5-10 min

$5 \leq X1 - X2 \leq 10$

fuelling: 10-20 min

$10 \leq X4 - X3 \leq 20$

fuelling after deboarding

$0 \leq X3 - X2 \leq \infty$

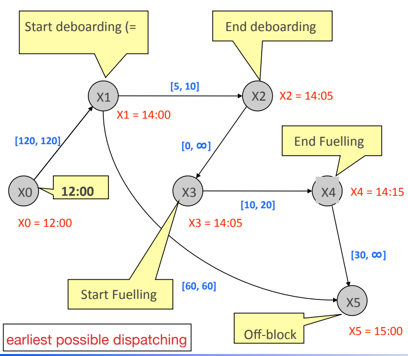
turnaround: 60 min

$60 \leq X5 - X1 \leq 60$

off-block ≥30 after fuelling

$30 \leq X5 - X4 \leq \infty$

Dispatching the process



turnaround constraints

on-block+start deboarding at 14.00 o'clock

$120 \leq X1 - X0 \leq 120$

deboarding: 5-10 min

$5 \leq X1 - X2 \leq 10$

fuelling: 10-20 min

$10 \leq X4 - X3 \leq 20$

fuelling after deboarding

$0 \leq X3 - X2 \leq \infty$

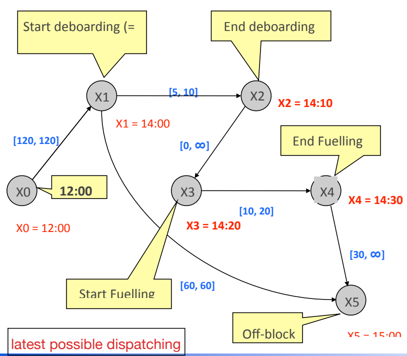
turnaround: 60 min

$60 \leq X5 - X1 \leq 60$

off-block ≥30 after fuelling

$30 \leq X5 - X4 \leq \infty$

Dispatching the process



turnaround constraints

on-block+start deboarding at 14.00 o'clock

$120 \leq X1 - X0 \leq 120$

deboarding: 5-10 min

$5 \leq X1 - X2 \leq 10$

fuelling: 10-20 min

$10 \leq X4 - X3 \leq 20$

fuelling after deboarding

$0 \leq X3 - X2 \leq \infty$

turnaround: 60 min

$60 \leq X5 - X1 \leq 60$

off-block ≥30 after fuelling

$30 \leq X5 - X4 \leq \infty$

Decoupling turnaround in 4 steps

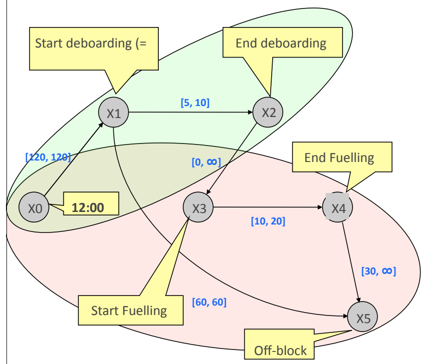
construct STN
using airport planning domain, construct overall problem representation using Simple Temporal Networks

decouple
decouple this STN into the relevant local (service-provider) domains, adding additional constraints where needed.

local solving
decoupled STN's are solved locally by each service provider, applying its own preferences, tools and methods

merging
local solutions (schedules) are merged into a total airport turnaround schedule which is guaranteed to be conflict-free

Decoupling



Two parties involved

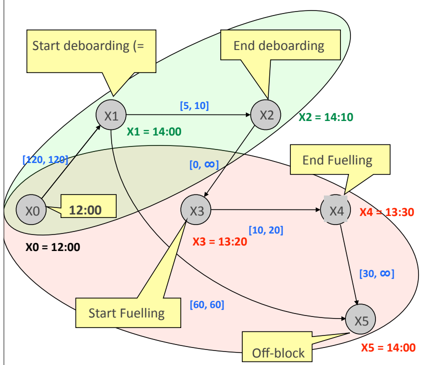
deboarding

fuelling

Each party controls a sub STN

CONFLICTING SOLUTIONS POSSIBLE

Decoupling



Two parties involved

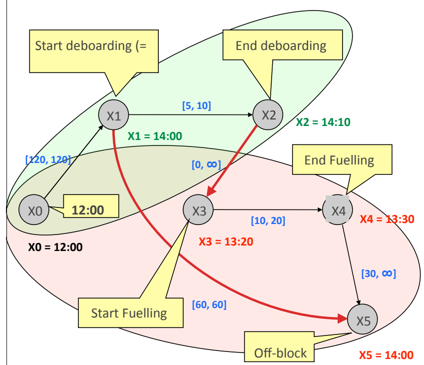
deboarding

fuelling

Each party controls a sub STN

CONFLICTING SOLUTIONS

Decoupling



Two parties involved

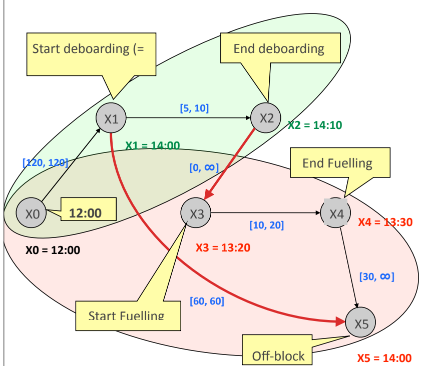
deboarding

fuelling

Each party controls a sub STN

INTER AGENT CONSTRAINTS VIOLATED

Decoupling



Temporal Decoupling Idea

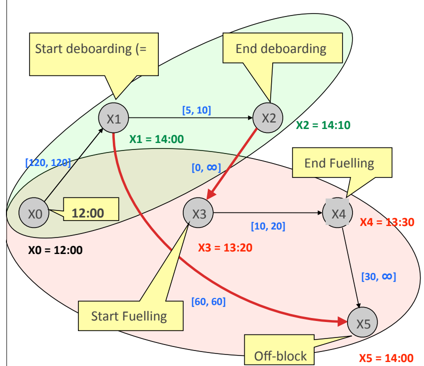
Tighten some intra-agent constraints such that all inter agent constraints are satisfied.

Result

two decoupled subsystems, independently solvable

No more conflicting solutions

Decoupling



Temporal Decoupling Idea

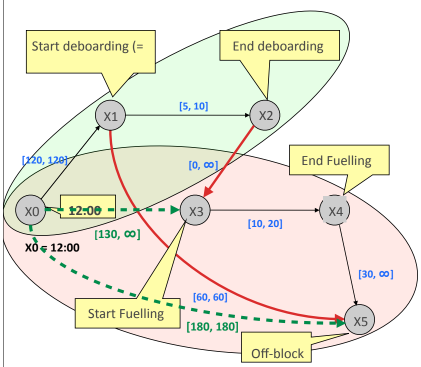
Tighten some intra-agent constraints such that all inter agent constraints are satisfied.

Result

two decoupled subsystems, independently solvable

No more conflicting solutions

Decoupling



Decoupling Process

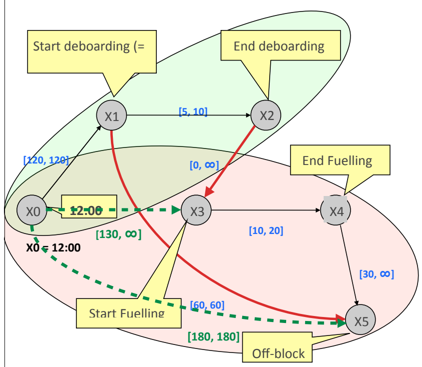
1. Add temporal constraint for (X0,X3):

   130 ≤ X3 – X0 ≤ ∞

2. Add temporal constraint for (X0,X5):

   180 ≤ X5 – X0 ≤ 180

Decoupling



Decoupling Process

1. Add temporal constraint for (X0,X3):

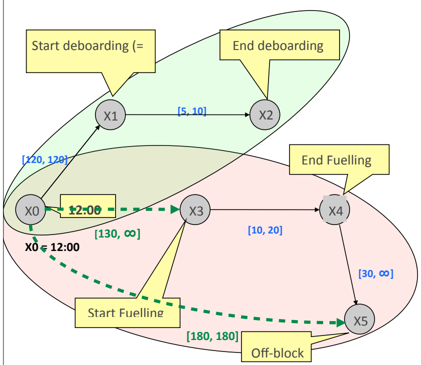
   130 ≤ X3 – X0 ≤ ∞

2. Add temporal constraint for (X0,X5):

   180 ≤ X5 – X0 ≤ 180

3. Remove superfluous inter agent constraints

Decoupling



Decoupling Process

1. Add temporal constraint for (X0,X3):

   130 ≤ X3 – X0 ≤ ∞

2. Add temporal constraint for (X0,X5):

   180 ≤ X5 – X0 ≤ 180

3. Remove superfluous inter agent constraints

status first problem

Question

How to obtain for each ground handling agent a temporal plan such that it can schedule its activities independently of the others, whilst the feasibility of the overall solution is ensured.

Result

After decoupling, each agent can decide upon its own schedule.

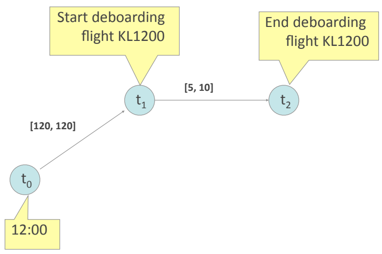
All individual schedules are mergeable into an conflict-free schedule for the overall schedule.

Conclusion

Temporal decoupling solves our autonomous scheduling problem for the turnaround process at airports

Determining Resource Needs

Composing service plans



<u>Flight KL1200 deboarding:</u>

Gate:	F12
Earliest start:	14:00
Latest start:	14:00
Earliest end:	14:05
Latest end:	14:10

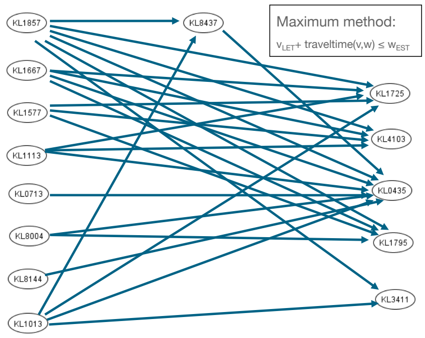
Determining Resource Needs:
Example for the Fuelling Company

Flight	EST	LST	EET	LET
KL1857	12:00:00	12:00:00	12:12:00	13:38:58
KL1013	12:10:00	12:10:00	12:22:00	13:46:15
KL1667	12:17:03	12:17:03	12:48:03	14:09:58
KL1577	12:29:30	12:29:30	13:00:30	14:17:15
KL8004	12:57:13	12:57:13	13:28:13	14:40:58
KL1113	13:00:30	13:00:30	13:27:30	15:51:49
KL0713	13:28:13	13:28:13	14:00:13	15:12:58
KL8114	13:29:15	13:29:15	13:56:15	16:18:49
KL3411	13:56:15	13:56:15	14:11:26	16:34:00
KL8437	14:00:13	14:00:13	14:31:13	15:43:58
KL4103	14:11:26	14:11:26	14:43:26	17:19:00
KL1725	14:31:13	14:31:13	14:43:13	16:34:00
KL1795	14:43:13	14:43:13	14:55:13	17:39:45
KL0435	16:48:01	16:48:01	17:19:01	18:10:45

Determining Resource Needs:
Minimum and Maximum Method

- **minimum method (optimistic)**
  - EET + traveltime(v,w) ≤ LST
  - 12:12:00 + 00:05:00 ≤ 12:17:03
  - For example: KL1577 *can be* serviced after KL1857
- **maximum method (pessimistic)**
  - LET+ traveltime (v,w) ≤ EST
  - 13:14:28 + 00:05:00 ≤ 12:17:03
  - For example: KL1577 *cannot be* serviced after KL1857

Determining Resource Needs:
Reachability Graph



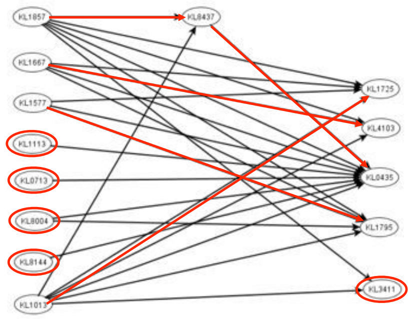
Maximum method:
v_{LET}+ traveltime(v,w) ≤ w_{EST}

Small Idea
To service KL1857, KL8437 and KL0435, we need one vehicle

Big Idea
Using the reachability graph, the minimum number of vehicles needed equals the smallest number of node-disjoint paths that cover the graph

Problem: find a smallest number of node-disjoint paths covering the reachability graph

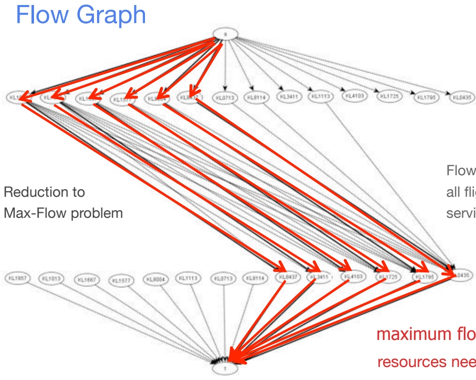
Determining Resource Needs:
Minimum Capacity based on the Maximum Method



Example of a path cover requiring 9 vehicles to do the fuelling

Can we do better?

Determining Resource Needs:
Flow Graph



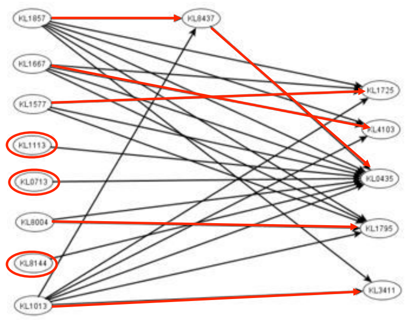
Reduction to Max-Flow problem

Flow is assigned to all flights not serviced first!

maximum flow: 6

resources needed: 14 -6 = 8

Determining Resource Needs:
Translating back



Flights: 14.

Maximum flow: 6

Paths with length 2:
{(KL1857,KL8437, KL0435)}

Paths with length 1:
{(KL1667,KL4103), (KL1577,KL1725), (KL8004,KL1795, (KL1013,KL3411)}

Paths with length 0:
{KL1113, KL0713, KL8114}

Required capacity:
|V| - f, with
- |V| = #nodes in graph
- f = maximum flow in flowgraph

Required capacity: 14-6=8

Tool Implementation

- Rather fast algorithm presented to determine resource needs: *$O(n^3)$*
- Execution time for a problem instance of 207 flights:
  - Minimum required capacity calculated in 23 sec.
  - Maximum required capacity in 17 sec.
  - Entire planning tool (both temporal planning and decoupling and determining resource needs):
    - Less than 3 minutes.
- Realistic application:
  - Execution time for a full-day scenario calculated in less than 20 minutes.

Conclusions

We presented a multi-agent solution to a real-life distributed planning problem presented solving the

- Autonomous Scheduling Problem:

  Agents can plan their activities independently while the feasibility of the overall solution is ensured.

- *Determining Resource Needs Problem:*

  Agents can obtain an estimation of the number of resources it minimally needs to carry out the activities listed in its schedule

Future Research

- More realistic resource determination by including
  - Resource availability in time
    (e.g. fuelling vehicles need re-fuelling themselves)
  - Shortest route calculation (reducing travel time)
- Re-planning in case of large disruptions:
  - Taking backup resources into account (including corresponding extra costs)
  - Adapting the level of decoupling (merge)
  - Look at swap as an operation to solve specific disruptions
- Integrating decoupling and minimisation of resource needs
- Development of validated decision support tool becoming operational at a real airport

References

Buzing, P.C. and A.W. ter Mors and J.M. Valk and C. Witteveen (2006). Coordinating Self-Interested Planning Agents. Autonomous Agents and Multi-Agent Systems 12 (2):199--218.

Brambilla,A., M Lavagna, A coordination mechanism to solve common resource contention in multi agent space systems, International Symposium on Artificial Intelligence, Robotics and Automation in Space-iSAIRAS pp. 26–29, 2008.

L. Hunsberger, Distributing the control of a temporal network among multiple agents, Proceedings of the AAMAS, pp 899-906, 2003

L. Hunsberger, Algorithms for a temporal decoupling problem in multi-agent planning, Proceedings of the AAAI, 2002.

L. Hunsberger. 2008. A Practical Temporal Constraint Management System for Real-Time Applications. In Proceedings of European Conference on Artificial Intelligence (ECAI-2008).

P. van Leeuwen, C. Witteveen, Temporal Decoupling and Determining Resource Needs of Autonomous Agents in the Airport Turnaround Process, IAT2009, pp. 185-192, 2009.

Planken, L.R. and Mathijs M. de Weerdt and Cees Witteveen (2010). Optimal Temporal Decoupling in Multiagent Systems. In Van der Hoek and Kaminka and Lesperance and Luck and Sen (Eds.). *Proceedings of the Ninth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-10)*, pp. 789-796, Toronto.

Smith, S.F., "Is Scheduling a Solved Problem?", Proceedings First Multi-Disciplinary International Conference on Scheduling: Theory and Applications (MISTA 03), Nottingham, UK, 2003

Yadati, C. and Cees Witteveen and Yingqian Zhang and Mengxiao Wu and Han La Poutre (2008). Autonomous Scheduling with unbounded and bounded agents. In Ralph Bergmann and Gabriela Lindemann and Stefan Kirn and Michal Pechoucek (Eds.). Multiagent System Technologies. 6th German Conference, MATES 2008, pp. 195-206, Kaiserlautern, Germany. Lecture Notes In Computer Science Springer -Verlag.

Witteveen, C., W. van der Hoek and N. Roos (2009). Concurrently Decomposable Constraint Systems. In L.Braubach et al. (Eds.). *Multiagent System Technologies, 7th German Conference, MATES Multiagent System Technologies MATES 2009*, pp. 153-164.