Delay Management Problem: Complexity Results and Robust Algorithms

Work partially supported by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2 (project ARRIVAL)

Serafino Cicerone[1] **Gianlorenzo D'Angelo**[1]
Gabriele Di Stefano[1] Daniele Frigioni[1] Alfredo Navarra[2]

[1]Dept. of Electrical and Information Engineering, University of L'Aquila, Italy
{cicerone,gdangelo,gabriele,frigioni}@ing.univaq.it

[2]Dept. of Mathematics and Informatics, University of Perugia, Italy
navarra@dipmat.unipg.it

Timetabling

Schedule the departure and arrival time of trains in order to reduce the traveling time for passengers

Delay Management

Modify the timetable when unpredictable events cause delays

Approaches

- Apply recovery strategies or rescheduling
- Design the timetable in order to easily recover when delays occur. This approach is called *Recoverable Robustness* [Liebchen et al. 2007, Cicerone et al. 2007]

This work

This work studies the *Recoverable Robustness* approach

Results of the paper

- We define the problem of finding timetables that are *recoverable robust* w. r. t. some kind of delays ($\mathcal{RDM}$ Problem)
- We show that finding an optimal solution for $\mathcal{RDM}$ is NP-hard
- We give an approximation algorithm, compute its price of robustness and we show that it is optimal for particular cases

Outline

Outline

Outline

Idea

Given some *recovery capabilities*, a timetable is *robust against delays* if it can be turned into a *new* feasible timetable by applying the recovery capabilities when a delay occurs

Mimimization Problem

$P = (I, F, f)$ where

- I, set of instances of P
- F, for each $i \in I$, $F(i)$ is the set of feasible solutions for i
- $f : S \to \mathbb{R}$, objective function of P, where $S = \bigcup_{i \in I} F(i)$

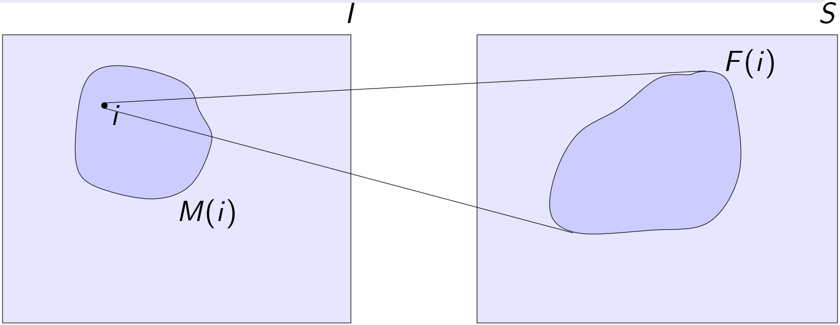
Recoverable Robustness Problem

$\mathcal{P} = (P, M, \mathbb{A})$

- $M : I \to 2^I$, *modification* function for instances of P
- $\mathbb{A}$, class of *recovery algorithms*

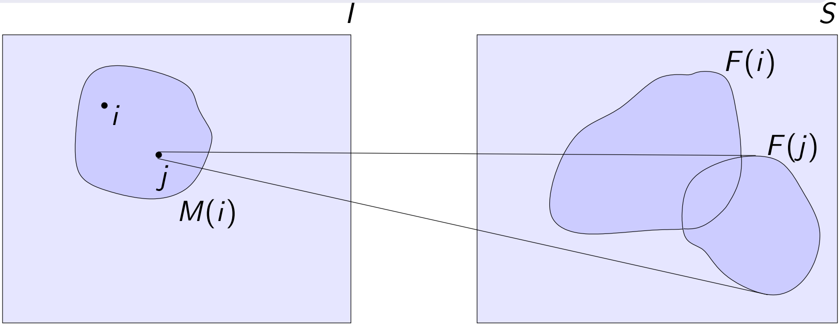
Modification function

- $i \in I$, an instance of P
- a disruption is a modification to i that can be seen as another instance $j \in M(i)$
- $M(i)$, set of instances of that can be obtained by applying all possible modifications to i



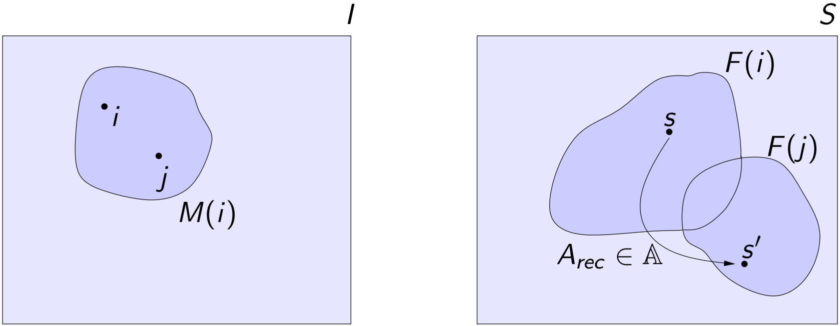
Modification function

- $i \in I$, an instance of P
- a disruption is a modification to i that can be seen as another instance $j \in M(i)$
- $M(i)$, set of instances of that can be obtained by applying all possible modifications to i



Recovery algorithms

- Represent recovery capabilities against disruptions
- Given $(i, s) \in I \times S$, $j \in M(i)$ and $A_{rec} \in \mathbb{A}$, $A_{rec}(i, s, j) = s' \in F(j)$



Recovery algorithms

$\mathbb{A}$ can be defined in terms of some restrictions, for example:

computational complexity: $\forall i \in I, \forall s \in S, \forall j \in M(i)$, $A_{rec}(i,s,j)$ must be computed in $O(\cdot)$ time

feasibility constraints: Given a distance function $d : S \times S \to \mathbb{R}$ and $\Delta \in \mathbb{R}$, then $\forall i \in I, \forall s \in S, \forall j \in M(i)$, $d(s, A_{rec}(i,s,j)) \leq \Delta$

Outline

A robust solution is a feasible solution that can be easily turned into a new feasible solution in case of disruptions

Definition (Robust Solution)

Given an instance $i \in I$, $s \in F(i)$ is a *robust solution* for i if and only if:

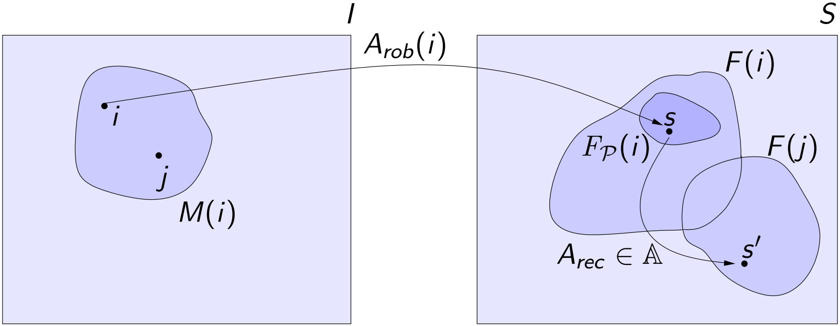
$$\forall j \in M(i),\ \exists A_{rec} \in \mathbb{A} :\ A_{rec}(i, s, j) \in F(j)$$

Set of Robust solutions:

$$F_{\mathcal{P}}(i) = \{s \in F(i) : s \text{ is a robust solution for } i\}$$

Definition (Robust Algorithm)

A *robust algorithm* is any algorithm A_{rob} such that, for each $i \in I$, $A_{rob}(i)$ is a robust solution for i with respect to $\mathcal{P}$.

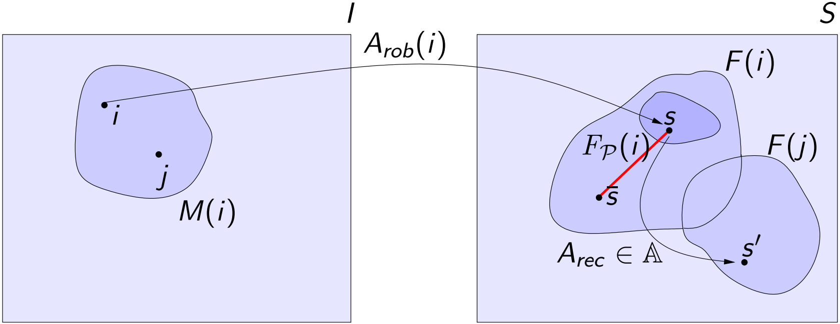


Strict Robustness

If we do not consider recoverability, that is, algorithms in $\mathbb{A}$ do not change the solution s, $\forall (i, s) \in I \times S$, $\forall j \in M(i)$, $A_{rec}(i, s, j) = s$, then $\mathcal{P} = (P, M, \mathbb{A})$ is called *strict robustness problem*

Outline

A robust solution can be far from the optimal one



Price of robustness

The worst case ratio between the cost of the solution computed by A_{rob} and the optimal one is called *price of robustness of A_{rob}*.
The minimum price of robustness among all the robust algorithms is called *price of robustness of problem*

Given a Recoverable Robustness Problem $\mathcal{P}$,

Definition (Price of robustness of a robust algorithm A_{rob})

$$P_{rob}(\mathcal{P}, A_{rob}) = \max_{i \in I} \left\{ \frac{f(A_{rob}(i))}{\min\{f(x) : x \in F(i)\}} \right\}$$

Definition (Price of robustness of $\mathcal{P}$)

$P_{rob}(\mathcal{P}) = \min\{P_{rob}(\mathcal{P}, A_{rob}) : A_{rob}$ is a robust algorithm for $\mathcal{P}\}$

Definition (exact and optimal algorithms)

- A_{rob} is *exact* if $P_{rob}(\mathcal{P}, A_{rob}) = 1$
- A_{rob} is $\mathcal{P}$*-optimal* if $P_{rob}(\mathcal{P}, A_{rob}) = P_{rob}(\mathcal{P})$

Outline

Outline

The problem [Schöbel, 2007]

Scheduling the departure and arrival time of trains

Instances

- A *event activity network* made of departure and arrival events and activities
- The *minimum time* needed for each activity
- The *number of passengers* involved in each activity

Solutions

A scheduled time for each event in the network that satisfies the minimum duration time of each activity

Objective

Minimizing the overall traveling time for passengers

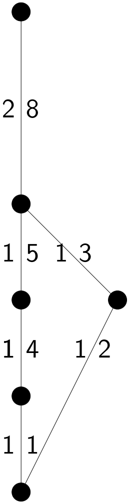
Instances

$\mathcal{N} = (\mathcal{E}, \mathcal{A})$ where

- $\mathcal{E}$: *departure* and *arrival events*
- $\mathcal{A}$: *driving*, *waiting* and *changing activities*

$L : \mathcal{A} \rightarrow \mathbb{N}$: for each activity $a \in \mathcal{A}$, $L(a)$ is the minimum duration time for a

$w : \mathcal{A} \rightarrow \mathbb{N}$: for each activity $a \in \mathcal{A}$, $w(a)$ is the number of passengers involved in a


2 8
1 5
1 3
1 4
1 2
1 1

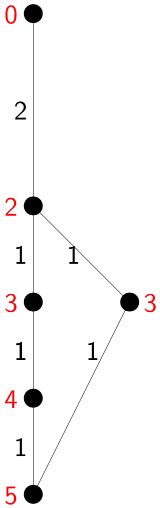
Feasible solutions

$\Pi \in \mathbb{N}^{|\mathcal{E}|}$ such that:

- $\Pi(v) - \Pi(u) \geq L(a)$, for each $a = (u, v) \in \mathcal{A}$

Objective

$\min f = \sum_{a=(u,v)\in\mathcal{A}} w(a)\,(\Pi(v) - \Pi(u))$


0
2
2
1
1
3
3
1
1
4
1
5

Outline

To define a recoverable robustness problem, we need to define a modification function M and a set of recovery capabilities $\mathbb{A}$

Modification function

We allow only one delay on an activity a of at most α time
We can model it as an increase of the minimal duration time a
Given $i = (\mathcal{N}, L, w)$,

$$M(i) = \{(\mathcal{N}, L', w) : L' \text{ differs from } L \text{ by at most one activity } \}$$

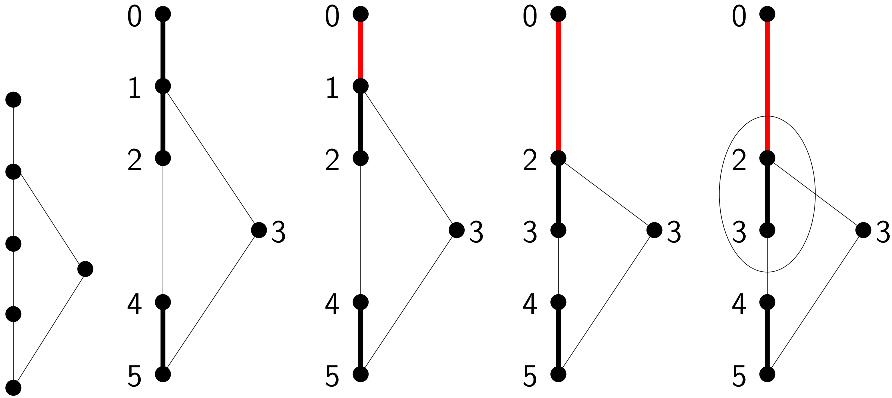
Recovery capabilities

A recovery algorithm can change the time of at most Δ *affected events*

Affected events

- The slack time of an activity $a = (u, v)$ is $s(a) = \Pi(v) - \Pi(u) - L(a)$
- Given a delay of α time on the activity $a = (u, v)$, a node x is affected if there exists a path from (u, v) to x whose total slack time is less then α

$$L(a) = 1 \; \forall a \in \mathcal{A}$$



Problem $\mathcal{RDM}$

$\mathcal{RDM} = (TT, M, \mathbb{A})$ is the problem of finding a timetable that can be recovered by changing the time of at most Δ events when a delay of at most α time occurs

Outline

Theorem

Finding a solution for $\mathcal{RDM}$ that minimizes f is NP-hard for $\Delta \geq 5$.

Corollary

Computing $P_{rob}(\mathcal{RDM})$ is NP-hard.

Outline

We give an approximation algorithm, compute its price of robustness and we show that it is optimal for particular cases

Critical Path Method

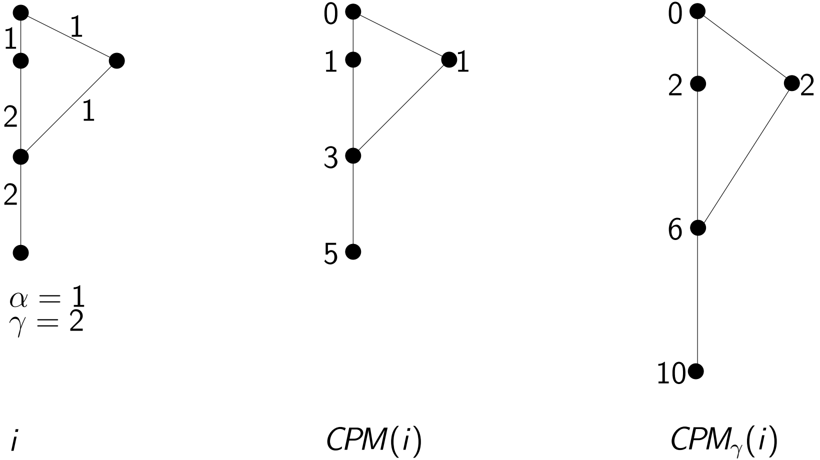
Assume that we have a single source in $\mathcal{N}$, v_0
Given $i = (\mathcal{N}, L, w)$,

$$CPM(i) = \begin{cases} \Pi(v) = 0 & \text{if } v = v_0 \\ \Pi(v) = \max\{\Pi(u) + L(a) : a = (u,v) \in \mathcal{A}\} & \text{otherwise} \end{cases}$$

- w_{min}, w_{max}, L_{min} and L_{max}: minimum and maximum values assigned by the functions w and L
- $\gamma = (1 + \frac{\alpha}{L_{min}})$
- $i_\gamma = (\mathcal{N}, \gamma L, w)$

Robust algorithm

$$CPM_\gamma(i) = CPM(i_\gamma)$$


1
1
2
1
2
α = 1
γ = 2
0
1
1
3
5
0
2
2
6
10
i
CPM(i)
CPMγ(i)

Robustness of CPM_γ

It is easy to prove that CPM_γ is robust against delays:
For each $a = (u, v) \in \mathcal{A}$,

$$\Pi(v) - \Pi(u) \geq \left(1 + \frac{\alpha}{L_{min}}\right) L(a) \geq L(a) + \alpha$$

Theorem (Price of robustness of CPM_γ)

For any $\Delta \geq 0$, the price of robustness of CPM_γ for $\mathcal{RDM} = (TT, M, \mathbb{A}_\Delta)$ is bounded by:

$$P_{rob}(\mathcal{RDM}, CPM_\gamma) \leq \gamma \frac{w_{max}}{w_{min}}$$

Theorem (Price of robustness of $\mathcal{RDM}$)

Let $\Delta = 0$ and $L(a)$ be constant for each $a \in A$, then

$$P_{rob}(\mathcal{RDM}) \geq \gamma \frac{w_{min}}{w_{max}}$$

Corollary

Let $\Delta = 0$, and w and L be constant, then $P_{rob}(\mathcal{RDM}, CPM_\gamma) = \gamma$ and CPM_γ is $\mathcal{RDM}$-optimal.

Outline

Results of the paper

- We have defined a recoverable robustness problem: the delay management problem with one single bounded delay and limited recovery capabilities
- We have shown that finding an optimal solution for this problem is *NP*-hard
- We have given an approximation algorithm which is optimal for particular cases

Future Works

- Considering more than one delays, other kind of disruptions or different recovery capabilities
- Taking into account the number of affected nodes
- Considering particular graph classes (e.g. Trees)
- Experimental evaluation