

# Evaluation of recoverable-robust timetables on tree networks

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

**Gianlorenzo D'Angelo**<sup>1</sup> Gabriele Di Stefano<sup>1</sup>  
Alfredo Navarra<sup>2</sup>

<sup>1</sup>Dept. of Electrical and Information Engineering, University of L'Aquila, Italy  
{gianlorenzo.dangelo,gabriele.distefano}@univaq.it

<sup>2</sup>Dept. of Mathematics and Informatics, University of Perugia, Italy  
{navarra, pinotti}@dmi.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

## Recoverable Robustness

Design the timetable in order to easily recover when delays occur  
[Liebchen et al. 2007, Cicerone et al. 2007]

## This work

Keep on study the Recoverable Robustness approach for timetabling

## Results of the paper

- ▶ We study the Recoverable Robustness Timetabling problem in restricted topologies (Tree)
- ▶ We prove that the Problem is *NP*-hard for trees
- ▶ We give a pseudo-polynomial time algorithm
- ▶ We experimentally evaluate the algorithm on both real data (Italian railways) and randomly generated data

# Outline

Recoverable Robust Timetabling problem

Complexity

Algorithm

Data description

Experimental results

Conclusions

# Outline

Recoverable Robust Timetabling problem

Complexity

Algorithm

Data description

Experimental results

Conclusions

## The timetabling problem [Schöbel, 2007]

Scheduling the departure and arrival time of trains

### Instances

- ▶ A *event activity network*  $\mathcal{N} = (\mathcal{E}, \mathcal{A})$  made of departure and arrival events  $\mathcal{E}$  and activities  $\mathcal{A}$
- ▶ The *minimum time*  $L(a)$  needed for each activity  $a \in \mathcal{A}$
- ▶ The *number of passengers*  $w(v)$  involved in each event  $v \in \mathcal{E}$

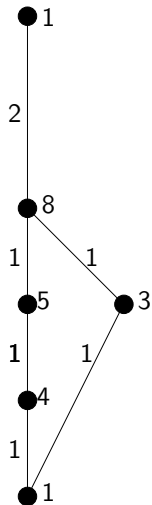
### Solutions

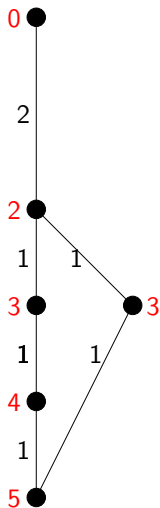
A scheduled time  $\Pi(v)$  for each event  $v \in \mathcal{E}$  in the network, such that  $\Pi$  satisfies the minimum duration time of each activity

### Objective

Minimizing the overall traveling time for passengers

$$\min f = \sum_{v \in \mathcal{E}} w(v) \Pi(v)$$







## Timetabling problem $TT$

$$\min f = \sum_{v \in \mathcal{E}} w(v) \Pi(v)$$

subject to

$$\begin{aligned} \Pi(v) - \Pi(u) &\geq L(a), & \text{for each } a = (u, v) \in \mathcal{A} \\ \Pi(v) &\in \mathbb{N}, & \text{for each } v \in \mathcal{E} \end{aligned}$$

- ▶  $I$ : set of instances  $i = (\mathcal{N}, L, w)$
- ▶  $F(i)$ : set of feasible solutions  $\Pi$  for  $i \in I$

## Modification function $M$

We allow only one delay on an activity  $a$  of at most  $\alpha$  time

We can model it as an increase of the minimal duration time  $a$

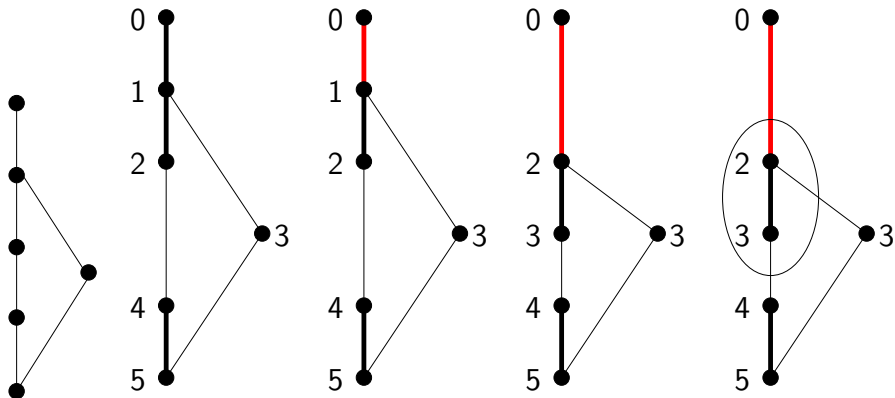
Given  $i = (\mathcal{N}, L, w)$ ,

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

## Recovery capabilities $\Delta$

A recovery algorithm can change the time of at most  $\Delta$  *affected events*

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



## The Recoverable Robust Timetabling Problem $\mathcal{RTT}$

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

## Robust Algorithm

A *robust algorithm* for  $TT$  is any algorithm  $A_r$  which solves  $\mathcal{RTT}$ .

## Price of robustness

The worst case ratio between the cost of the solution computed by  $A_r$  and the optimal one is called *price of robustness of  $A_r$* .

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

The minimum price of robustness among all the robust algorithms is called *price of robustness of problem*

## Previous Works

- ▶  $RTT$  is  $NP$ -hard for general event activity networks and directed acyclic graphs
- ▶ Linear time approximation algorithm
- ▶ Polynomially solvable for linear graphs

## In this paper

- ▶ We restrict the topology of event activity networks to trees

# Outline

Recoverable Robust Timetabling problem

Complexity

Algorithm

Data description

Experimental results

Conclusions

## Theorem

*Finding a solution for  $\mathcal{RTT}$  that minimizes  $f$  is NP-hard even on tree topologies.*

## Corollary

*Computing  $P_{rob}(\mathcal{RTT})$  is NP-hard even on tree topologies.*



# Outline

Recoverable Robust Timetabling problem

Complexity

**Algorithm**

Data description

Experimental results

Conclusions

## Slack times

According to a timetable  $\Pi$ , the slack time  $s(a)$  of an activity  $a = (u, v)$  is the amount of time assigned to an activity in addition to its minimum time needed  $L(a)$

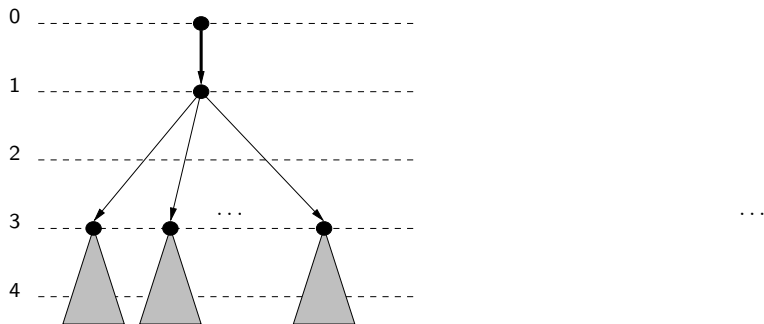
$$s(a) = \Pi(v) - \Pi(u) - L(a)$$

A robust timetable assigns at one least slack time of  $\alpha$  every  $\Delta$  subsequent events

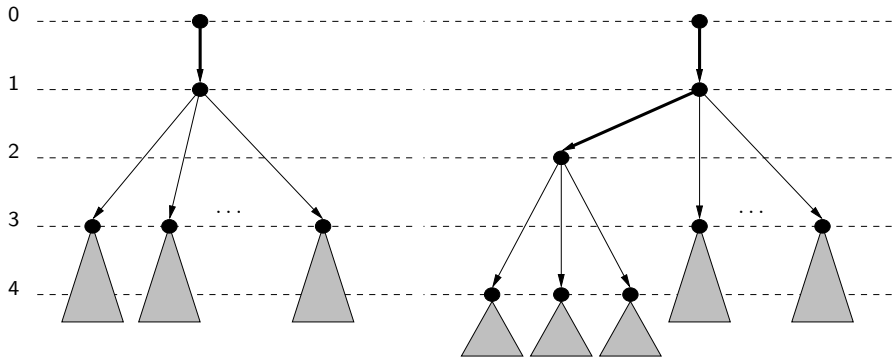
## Idea

Assign the slack times as late as possible

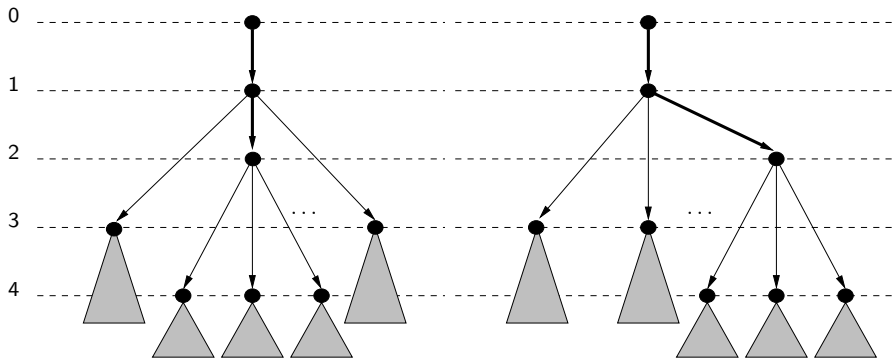
Example for  $\Delta = 2$  and  $L(a) = 1$  for each  $a \in \mathcal{A}$



Example for  $\Delta = 2$  and  $L(a) = 1$  for each  $a \in \mathcal{A}$



Example for  $\Delta = 2$  and  $L(a) = 1$  for each  $a \in \mathcal{A}$



## Computational complexity

This algorithm optimally solves *RTT* in  $O(n^{\Delta+1})$  time. It is pseudo-polynomial for fixed  $\Delta$

# Outline

Recoverable Robust Timetabling problem

Complexity

Algorithm

**Data description**

Experimental results

Conclusions

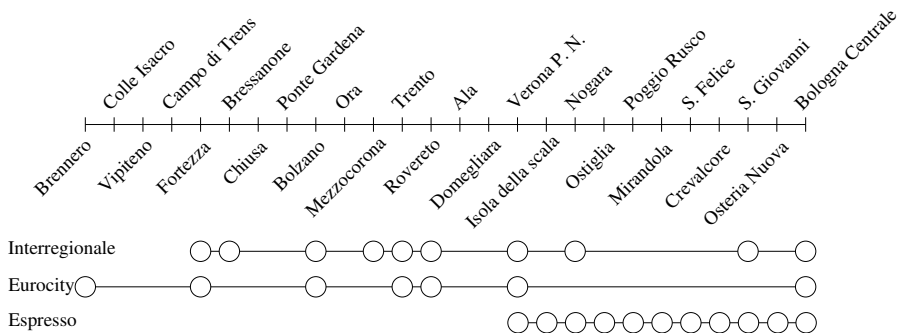
We evaluate the algorithms on two data sets:

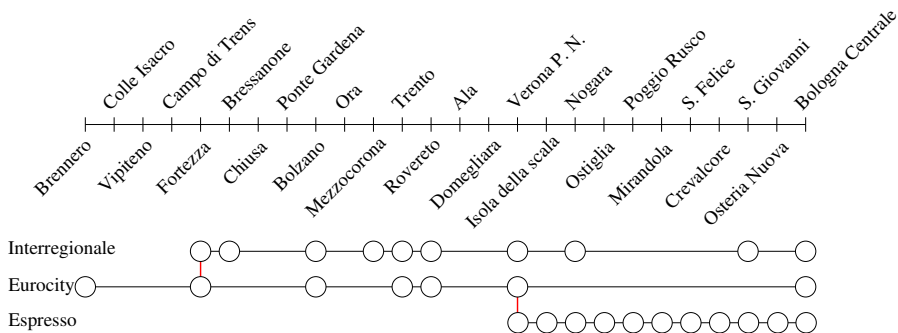
- ▶ Italian Single line corridors
- ▶ Randomly generated data



## Single line corridors

- ▶ A corridor is a set of subsequent stations served by many trains of different type
- ▶ In practice (and intuitively) slow trains wait for faster trains to allow passengers to change from one train to another
- ▶ We require that changes between trains are only those connecting a fast train to the starting event of a slow train





## Randomly generated data

- ▶ Each tree contains 1000 nodes
- ▶ Each tree is generated starting from a single node and then by linking a new generated node to an existing one extracted uniformly at random.

# Outline

Recoverable Robust Timetabling problem

Complexity

Algorithm

Data description

**Experimental results**

Conclusions

<b>Corridor</b>	<b>Line</b>	<b>N. of Stations</b>	<b>N. of Trains</b>
BrBo	Brennero–Bologna	48	68
MdMi	Modane–Milano	54	291
BzVr	Bolzano–Verona	27	65
PzBo	Piacenza–Bologna	17	25

**Table:** Real world data used in the experiments.

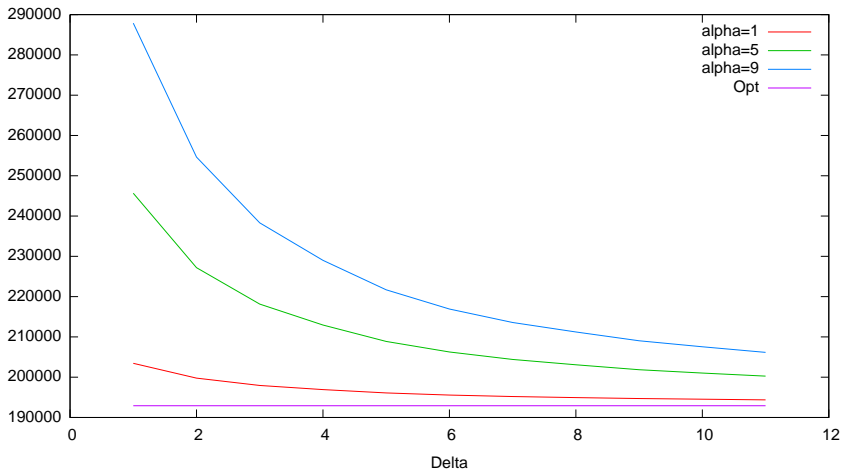
Corridor	N. of Nodes	Avg Activity Time
BrBo	1103	9
MdMi	4358	8
BzVr	648	5
PzBo	163	10
Random	1000	9

Table: Sizes of the trees.

$$\alpha \in \{1, 5, 9, 13, 17\}$$

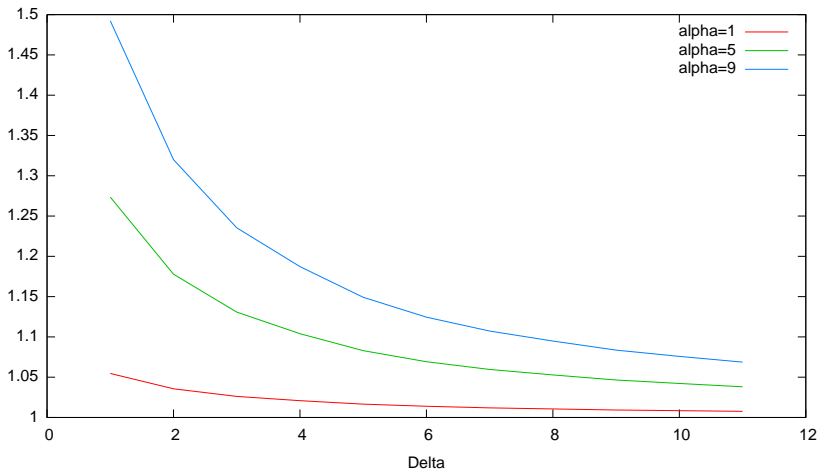
$$\Delta \in \{1, 2, \dots, 11\}$$

## BrBo Objective function

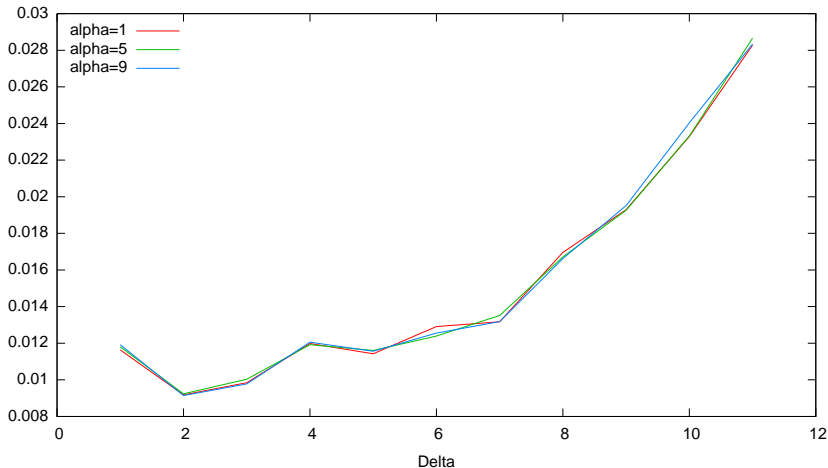




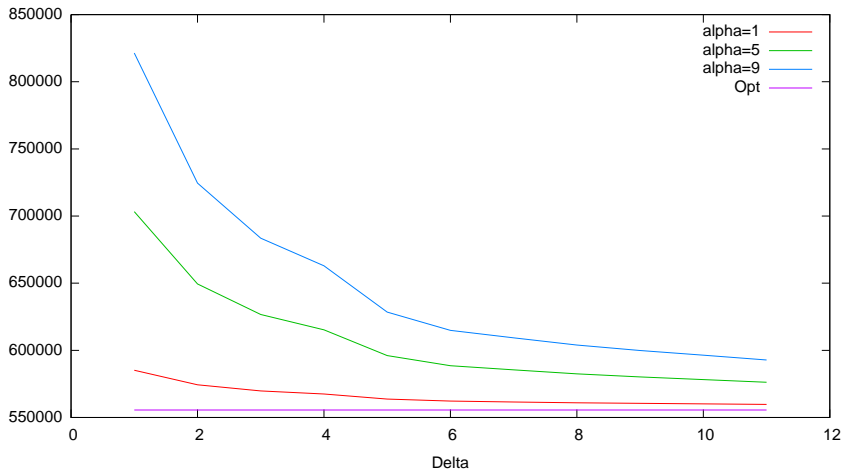
## BrBo Price of robustness



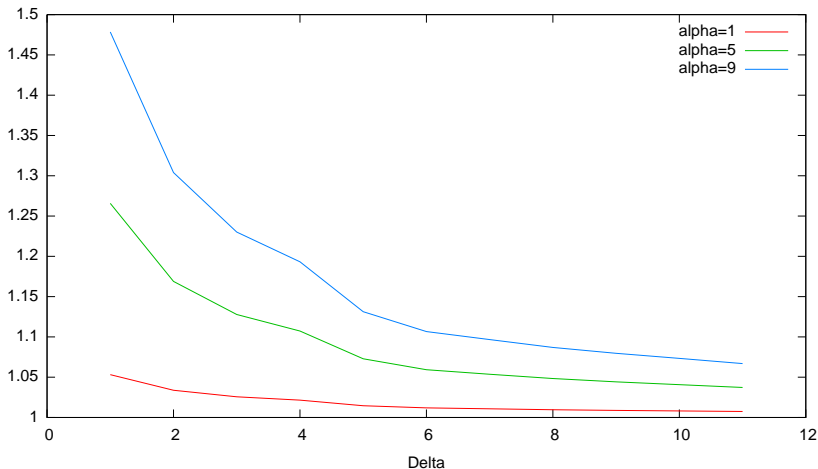
## BrBo Computational time



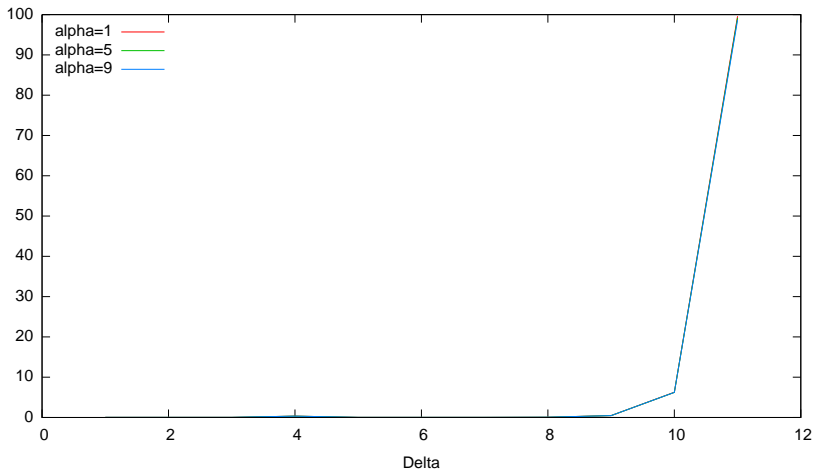
## MdMi Objective function



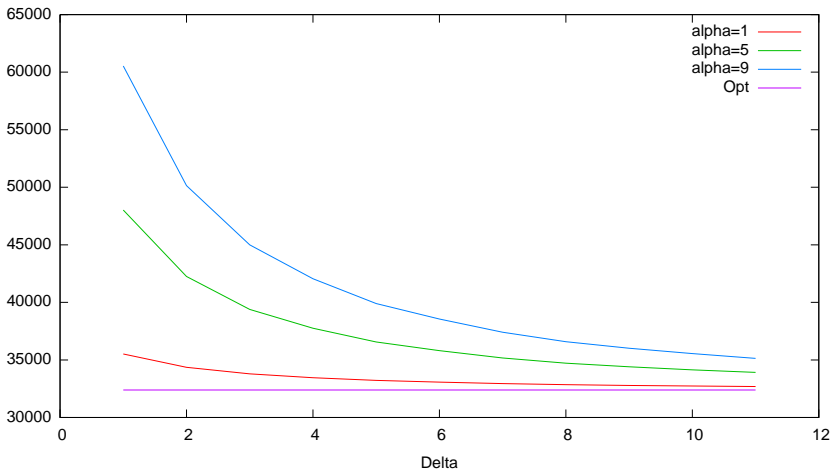
## MdMi Price of robustness



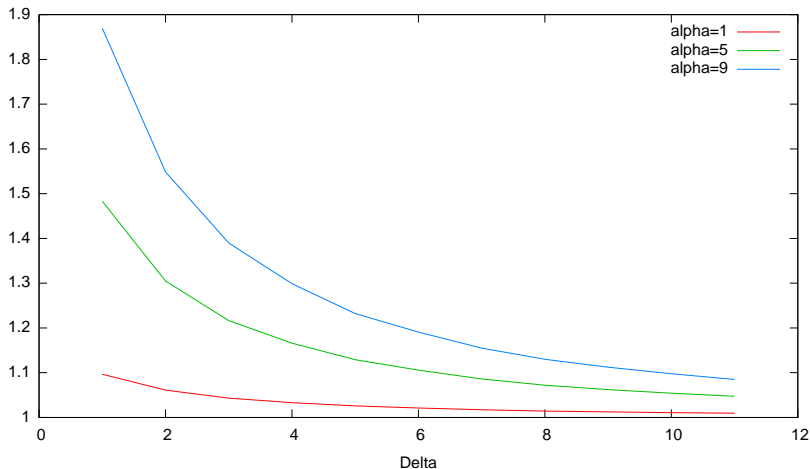
## MdMi Computational time



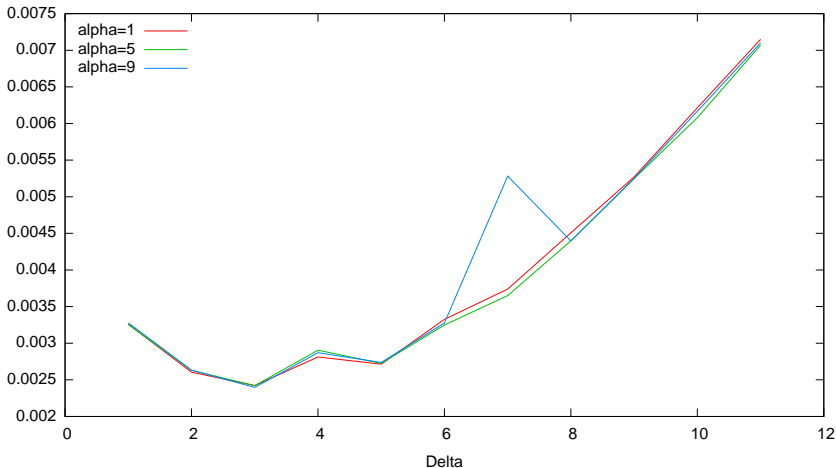
## BzVr Objective function



## BzVr Price of robustness

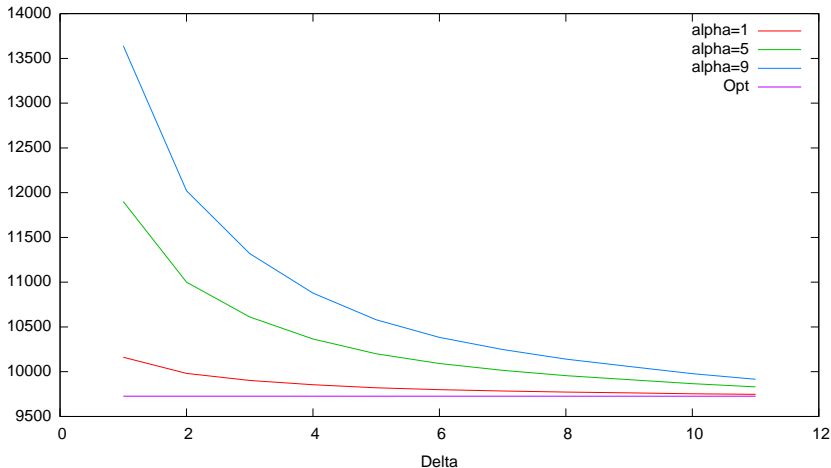


## BzVr Computational time

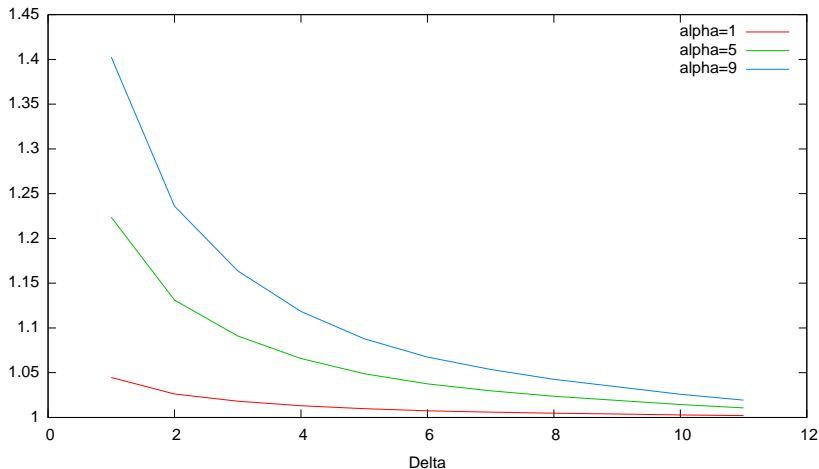




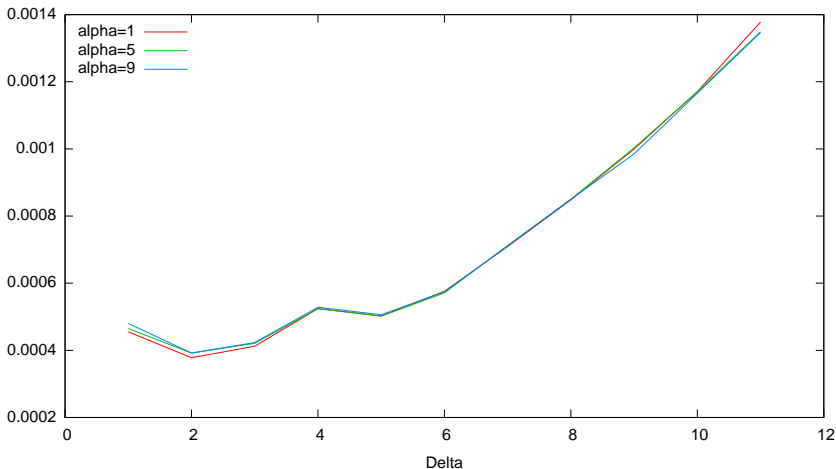
## PzBo Objective function



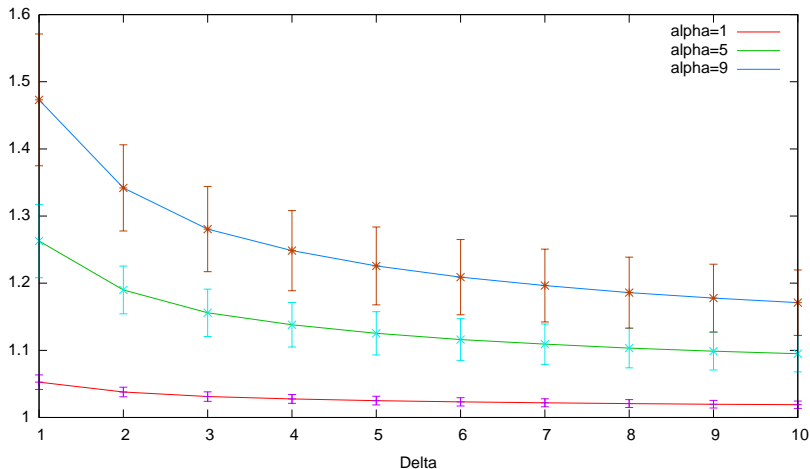
## PzBo Price of robustness



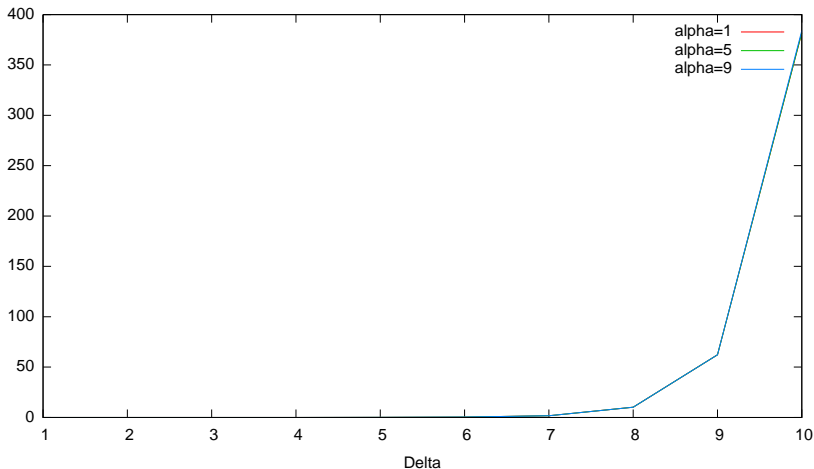
## PzBo Computational time



## Random Graphs with 1000 nodes – Price of robustness



## Random Graphs with 1000 nodes – Computational time



# Outline

Recoverable Robust Timetabling problem

Complexity

Algorithm

Data description

Experimental results

**Conclusions**

- ▶ We shown that the  $RTT$  problem is  $NP$ -hard even on tree networks
- ▶ We proposed a pseudo-polynomial time algorithm
- ▶ We experimentally shown the performances of the algorithm applied on both real data and randomly generated data
- ▶ Although the problem is proved to be  $NP$ -hard, the obtained results show the applicability of the algorithm