
Matteo Fischetti
@MFischetti
Trying to convince @mmonaci74 to use this title & abstract for our next paper. Please favorite if you are with me.
Visualizza traduzione
Can a general-purpose MILP solver be effective for real-time train rescheduling?
Matteo Fischetti, Michele Monaci
DEI, Università di Padova, Via Gradenigo 6/A, I-35131 Padova, Italy
{matteo.fischetti, michele.monaci}@unipd.it
October 3, 2015
Abstract
Yes.
Key words: Train Rescheduling, Railways optimization, Mixed-Integer Linear Programming, Real-time optimization, Computational Experiments.
RETWEET
2
MI PIACE
44
09:23 - 3 ott 2015

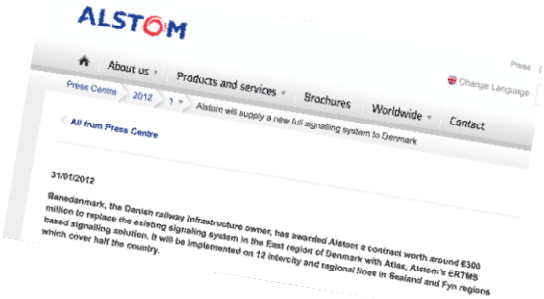
Real-time train rescheduling

Train scheduling: routing and scheduling a set of trains on a railway network (including stations) at planning level → nominal timetable

- **Train rescheduling**: nominal train timetable is fixed but unexpected events (e.g., delays and/or disruptions) occur on the network, introducing possible conflicts

- **Recovery actions** must be decided in a very short time (typically, within 2 to 10 seconds, depending on the time-window size and of the number of affected trains)

The challenge



Alstom is a global leader in the world for railways infrastructures

In 2012, Banedanmark (the Danish railway infrastructure owner) awarded Alstom a **€ 300M contract** to replace the existing signaling system in the East region of Denmark

Alstom soon activated **international collaborations** with optimization experts, with the aim of improving its dispatching system (ICONIS). We have been involved as **MIP experts**

A set of small/medium/large **real instances** has been provided to all collaborators, representing simulated disruption for a line nearby the city of London

“Can today’s MIP technology heuristically solve each of them in 2/5/10 sec.s on a reasonable hardware?”

Mathematical Programming models

- Alternative-graph representation (event-based)

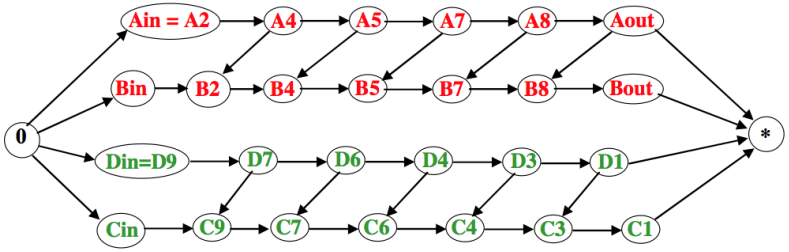


Figure 2: Event graph

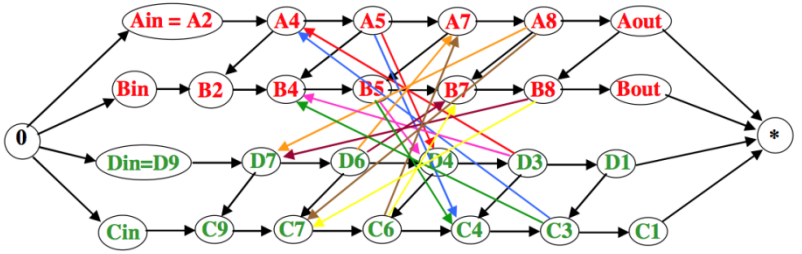


Figure 3: Alternative graph

- MIP model (with BIGM's)

$$\min \; t_* - t_0$$

s.t.

$$t_v - t_u + M(1-y_{eb}) \geq f_{uv} \qquad (i,j) \in F$$

$$t_j - t_k + M(1-y_{eb}) + M(1-y_{cd}) + Mx_{kjhi} \geq a_{kj}$$

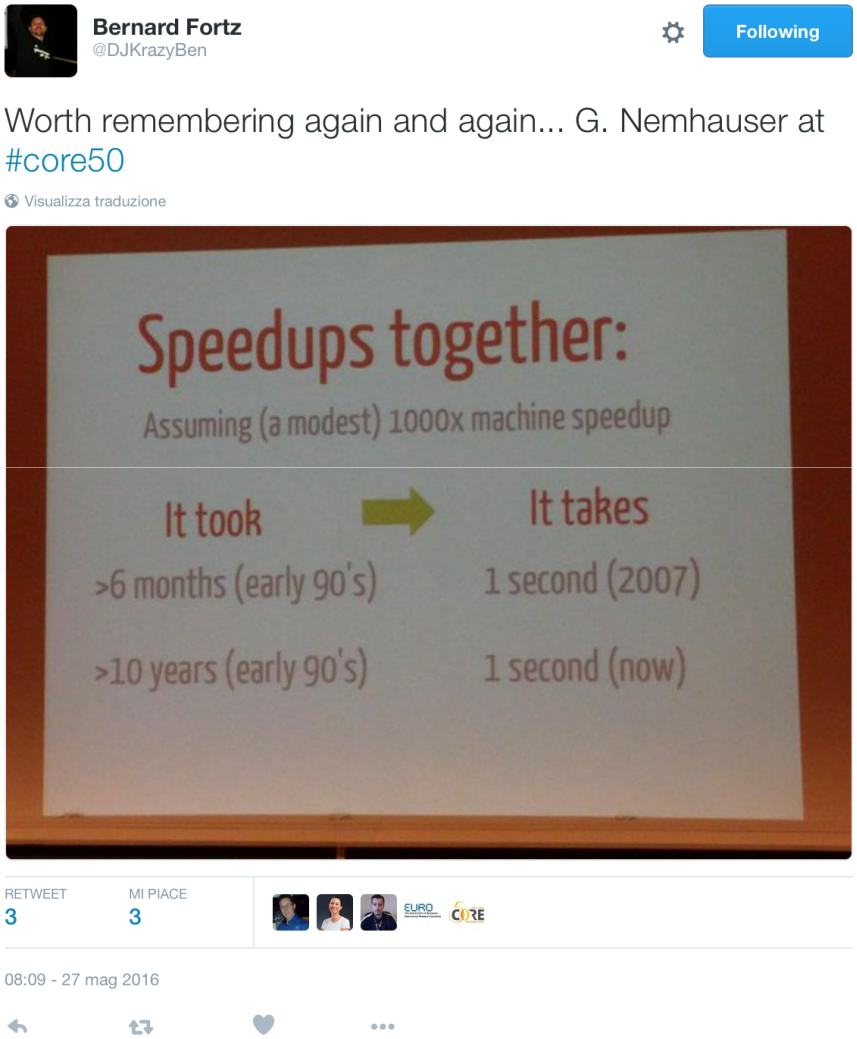
$$t_i - t_h + M(1-y_{eb}) + M(1-y_{cd}) + M(1-x_{kjhi}) \geq a_{hi} \qquad ((k,j),(h,i)) \in A$$

$$\Sigma_{e=1,..,\,rb}\, y_{eb} = 1 \qquad b = 1,\ldots, n_T$$

$$x_{kjhi} \in \{0,1\}$$

- Wide body of literature:
  - Caprara, A., Kroon, L., Monaci, M., Peeters, M., and Toth, P. (2007). Passenger railway optimization. In Bernhart, C. and Laporte, G., editors, Handbook in OR & MS, Vol. 14, chapter 3, pages 129-197. Elsevier, North-Holland.
  - **D'Ariano, A., D'Ariano, P., Sama, S., and Pacciarelli, D. (2013). Real-time train scheduling: from theory to practice. Technical report, RT-DIA-207- 2013, Università degli Studi Roma Tre**.
  - Mannino, C. and Mascis, A. (2009). Optimal real-time traffic control in metro stations. Operations Research, 57(4):1026-1039.
  - …

Can we just use a generic MIP solver?



Unfortunately not ...

Instance						Cplex				Cplex heu			
ID	NT	NR	#cols	#rows	z_{BST}	t_{TOT}	z_{FIN}	t_1	z_1	t_{TOT}	z_{FIN}	t_1	z_1
1	33	55	4508	8528	0.90	1.57	0.90	1.52	**13.46**	1.37	0.90	1.16	**49.55**
2	37	46	3499	6289	15.76	0.89	15.76	0.51	**68.57**	0.36	15.76	0.21	**30.49**
3	41	62	5331	10127	151.02	8.86	151.02	2.41	290.28	9.28	151.02	1.61	**159.23**
4	38	58	10056	19653	8.52	27.22	8.52	4.41	9.08	42.99	8.52	2.91	14.01
5	40	43	2867	4548	15.51	0.50	15.51	0.47	**15.51**	0.55	15.51	0.34	**95.93**
6	34	42	5742	10606	8.82	2.55	8.82	2.55	8.82	2.03	8.82	1.99	**8.82**
7	36	44	4994	9187	13.44	1.42	13.44	1.03	**58.46**	1.51	13.44	1.46	**13.67**
8	32	40	4462	8216	3.86	0.71	3.86	0.62	**3.86**	1.00	3.86	0.97	**3.86**
9	33	45	9363	17827	40.50	51.44	40.50	51.39	40.50	18.92	40.50	16.14	52.85
10	37	48	6994	13226	90.78	17.69	90.78	12.80	128.78	9.96	90.78	3.30	90.78
11	40	73	14506	28139	3.41	31.88	3.41	6.11	29.26	9.00	3.41	4.11	**34.93**
12	48	79	15182	197594	44.11	490.36	44.11	20.17	223.53	485.57	44.11	9.46	241.85
13	54	72	16232	30958	25.36	1410.71	25.36	197.49	194.66	948.29	25.36	8.48	41.45
14	52	59	8529	15181	10.75	23.10	10.75	3.96	**66.17**	9.95	10.75	2.63	**20.63**
15	49	53	7318	12559	24.30	33.19	24.30	3.88	**140.66**	10.05	24.30	2.80	**24.59**
16	42	51	11996	22459	67.53	90.12	67.53	13.59	141.47	34.68	67.53	7.34	1024.36
17	44	57	12683	24020	23.84	116.57	23.84	18.21	54.42	51.62	23.84	27.92	135.25
18	43	54	9081	16778	11.91	18.61	11.91	2.26	**45.98**	10.65	11.91	2.85	**116.00**
19	41	55	15145	28876	58.81	170.77	58.81	147.36	69.41	123.07	58.81	61.85	97.15
20	59	65	16431	30025	189.35	109.98	189.35	11.64	198.49	33.23	189.35	4.52	**237.52**
21	57	83	34339	67124	7.36	265.14	7.36	28.86	19.78	94.92	7.36	17.80	10.92
22	58	87	58036	114982	14.62	2529.88	14.62	710.88	31.06	1938.17	14.62	332.58	65.21
23	64	80	26061	49277	37.14	656.77	37.14	105.01	38.64	523.92	37.14	187.67	38.92
24	58	91	41827	81587	8.11	542.45	8.11	469.67	22.49	174.25	8.11	90.17	20.82
25	66	97	34052	762030	44.75	3600.00	121.53	355.92	121.53	3600.01	45.48	103.59	299.78
26	90	128	52542	101789	69.11	3600.00	98.87	1141.91	98.87	3600.04	69.11	51.64	95.23
27	59	64	19014	34616	34.57	234.90	34.57	17.45	120.74	258.49	34.57	17.19	68.56
28	85	105	56520	108630	–	3600.00	–	–	–	3600.00	–	–	–
29	52	59	8529	15181	10.75	23.08	10.75	4.04	**66.17**	9.93	10.75	2.64	**20.63**

Table 1: Solution of our instances using `Cplex` in two different settings by using 10 quadcore PC's (1h timelimit). Boldface entries meet the target computing time of 2/5/10 sec.s

But we are MIP expert, don't we?

- We worked hardly on a number of exciting ideas

  - **BIGM tightening** (exact and heuristic)
  - Ad hoc **branching** rules exploiting erraticity (bet-and-run)
  - Heuristic **Benders' decomposition** (*x* var.s in the master, *t* vars. subprob.)
  - Heuristic **McCormick linearization** of disjunctions represented by bilinear terms instead of BIGMs

    $x = 1 \rightarrow y := a^T t - b \geq 0$ with *x* binary and *y* continuous (scalar) var.s

    $z = xy,\ z \geq 0$ bilinear reformulation of the disjunction
  - Ad-hoc **preprocessing** based on fast shortest path computations
  - Independent **parallel** runs with different **random seeds/parameters**
  - Heuristic variable **fixing** and **local branching**
  - Improved dual bounds by **aggressive cutting planes**
  - **…**

Eventually we did it!

Instance			MIP heuristic					
NPC			1	4	8	1	4	8
#threads			1	1	1	4	4	4
ID	z_{BST}	tl	best solution value at time limit tl					
1	0.90	2	0.90	0.90	0.90	0.90	0.90	0.90
2	15.76	2	15.76	15.76	15.76	15.76	15.76	15.76
3	151.02	2	158.67	158.67	152.34	151.02	151.02	151.02
4	8.52	2	–	9.93	8.67	8.52	8.67	8.67
5	15.51	2	15.51	15.51	15.51	15.51	15.51	15.51
6	8.82	2	8.82	8.82	8.82	8.82	8.82	8.82
7	13.44	2	13.44	13.44	13.44	13.44	13.44	13.44
8	3.86	2	3.86	3.86	3.86	3.86	3.86	3.86
9	40.50	2	–	107.28	79.80	–	107.28	79.80
10	90.78	2	–	180.86	98.42	–	180.86	98.42
11	3.41	5	3.41	3.41	3.41	3.41	3.41	3.41
12	44.11	5	55.45	55.45	55.45	44.58	44.58	44.58
13	25.36	5	–	55.85	55.85	–	35.11	34.88
14	10.75	5	11.94	11.94	11.94	11.94	11.94	11.94
15	24.30	5	24.46	24.30	24.30	24.30	24.30	24.30
16	67.53	5	–	130.58	130.58	–	146.91	154.09
17	23.84	5	23.84	23.84	23.84	23.84	23.84	23.84
18	11.91	5	11.91	11.91	11.91	11.91	11.91	11.91
19	58.81	5	–	111.94	104.16	–	111.94	111.94
20	189.35	10	–	234.46	190.13	–	235.81	189.35
21	7.36	10	–	13.19	7.37	–	13.19	7.37
22	14.62	10	16.41	16.41	15.86	14.99	14.65	14.66
23	37.14	10	–	134.01	134.01	–	98.29	98.26
24	8.11	10	8.11	8.11	8.11	8.11	8.11	8.11
25	44.75	10	83.00	83.00	78.14	45.12	68.24	49.26
26	69.11	10	–	81.90	81.90	–	72.15	71.62
27	34.57	10	–	48.11	48.11	–	47.50	47.50
29	10.75	10	11.94	11.94	11.94	11.94	11.94	11.94

Hardware: independent runs on 1 or 4 or 8 independent quadcore PC's with 16GB ram each (last-generation Intel Core i7 @ 4Ghz)

Threads per PC:

- 1 (faster root node)
- 4 (faster enumeration)

For all instances(*), almost optimal solutions found within the imposed time limit of 2/5/10 wall-clock sec.s!

(*) but instance 28, for which no solution was found even by 10 runs of best-tuned Cplex with 1-hour time limit → infeasible?

VERY GOOD! JUST ADD SOME THEOREMS, WRITE A PAPER, AND SUBMIT IT!

Science or Science Fiction?

- Our final algorithm implemented a **bunch of complicated ideas**

  … so we decided to clean it up

  … to make it **as simple as possible**

  … without deteriorating it too much

- This was for three good reasons:
  - A too complicated code is difficult to **maintain**
  - A sophisticated alg. with many parameters is **prone to overfitting**
  - **We wanted to know which are the "ideas that work best" so we could share them with our colleagues working on similar problems #PublishOrPerish**

- So we applied **Occam's principle** (law of parsimony) that is commonly used by scientists in other fields (physics etc.)

Occam’s razor

- **Occam's razor**, or law of parsimony (lex parsimoniae): a problem-solving principle devised by the English philosopher William of Ockham (1287–1347)

- **The simpler the better**: among competing hypotheses, the one with the fewest assumptions is more likely be true and should be preferred → less overfitting

- Used as a heuristic guide in the development of **theoretical models** (Albert Einstein, Max Planck, Werner Heisenberg, etc.)

- Warning: not to misinterpreted and used as an excuse to address oversimplified models: “Everything should be kept as simple as possible, but no simpler” (Albert Einstein)

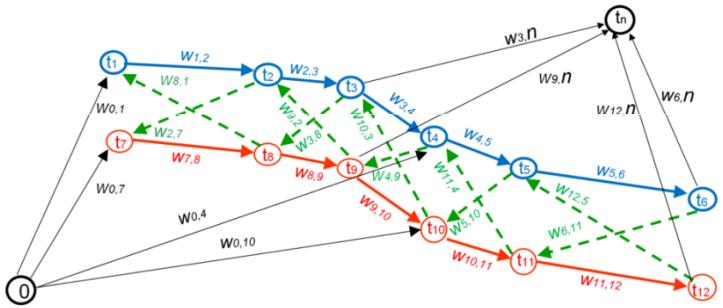
Keep it as simple as possible

- After a lot of tests and hard work, we were able to isolate **the two most effective ideas** in our code → removing all the others does not deteriorate (but actually **improves**) its overall performance !

- **Idea 1:** set a tight upper bound UB on the time of the very last event

  the last event t_n has a very special role in the model, in that it affects solution feasibility but also cost → a tight bound favors good sol.s!

  UB is determined by just trying an increasing sequence of values until feasibility is reached



  No need to implement clever/sophisticated ad-hoc **bound propagations** or **BIGM reductions**

  MIP technology improved dramatically in the last years → just USE it!

- **Idea 2**: exploit multiple cores/PCs by randomly fixing some sets of variables (e.g., choose one among alternative train reroutings)

Is simpler always better?

- So we know **two (actually very simple) ideas** that allow a general-purpose black-box MIP code to be effective in real-time train rescheduling

- How do you feel about this “very simple approach that works”? **Excited** or **disappointed**?

- A typical criticism is that

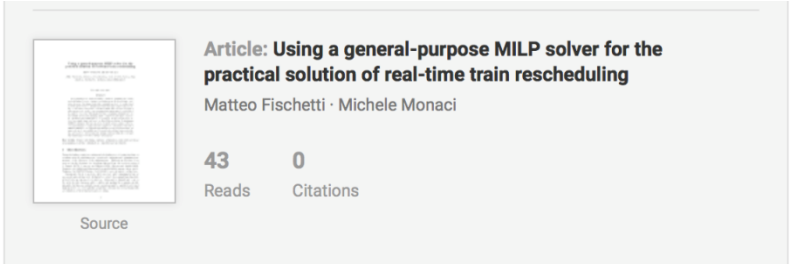
  … these ideas are **so simple** that “**do not deserve to be published**”

  … meaning that I should not bore you with this talk

  … but if insist you can always

  download our internal

  draft from ResearchGate



Lessons learned

- Today's technology does **not** allow a general-purpose MIP solver to be used as a black-box in real-time rescheduling

- However one can slightly modify the **input model** (and exploit a small number of **parallel PCs**) to achieve very satisfactory practical results

- Do not waste your time in **duplicating** stuff already in the MIP solver

THANK YOU FOR YOUR ATTENTION!