Solving Problems with CP: Four common pitfalls to avoid

Jean-Charles Régin

Univ. Nice-Sophia Antipolis, France

- Constraints and Proofs team at Univ. Nice-Sophia Antipolis
- Program verification with CP
- In competition with the well known COQ Proof assistant program at INRIA

- COQ is more formal, more theory oriented
  - is it better?

- **Verification is undecidable**

Plan

- What kind of problem can we solve with CP?
- 4 pitfalls to avoid
- Conclusion

What can of problems can we solve?

- I want to do something that could be useful in the future (50 years?)

- Polynomial
- Unclassified
- NP-Complete

Solving polynomial problems

- If we know that the problem is in P
  why do we need CP?
- If a P algorithm is known we don't need CP
- The problem is in P but we don't have any P algorithm
  - This is rare! I don't have any problem like that

Solving unclassified problems

- There are some problems like that
  - Some scheduling problems are large PERT with some additional constraints
- Three possibilities
  - We will prove it is in P: no more need of CP
  - We will prove it is NP-Complete (see later)
  - We will not prove anything (good for us)

Solving NP complete problems

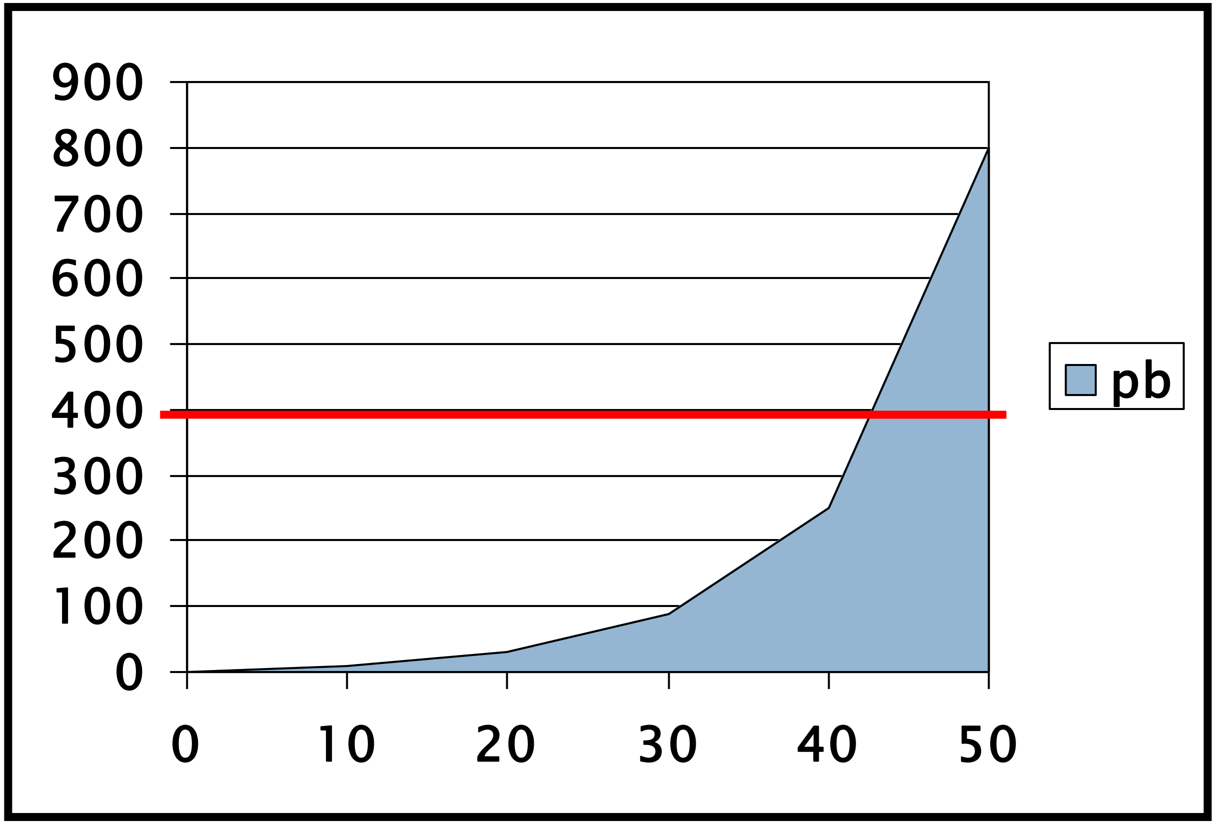
- Two possibilities
  - $P = NP$
  - $P \neq NP$

- The first case, is not good for us (see P part).
- Let's go for $P \neq NP$

$P \neq NP$

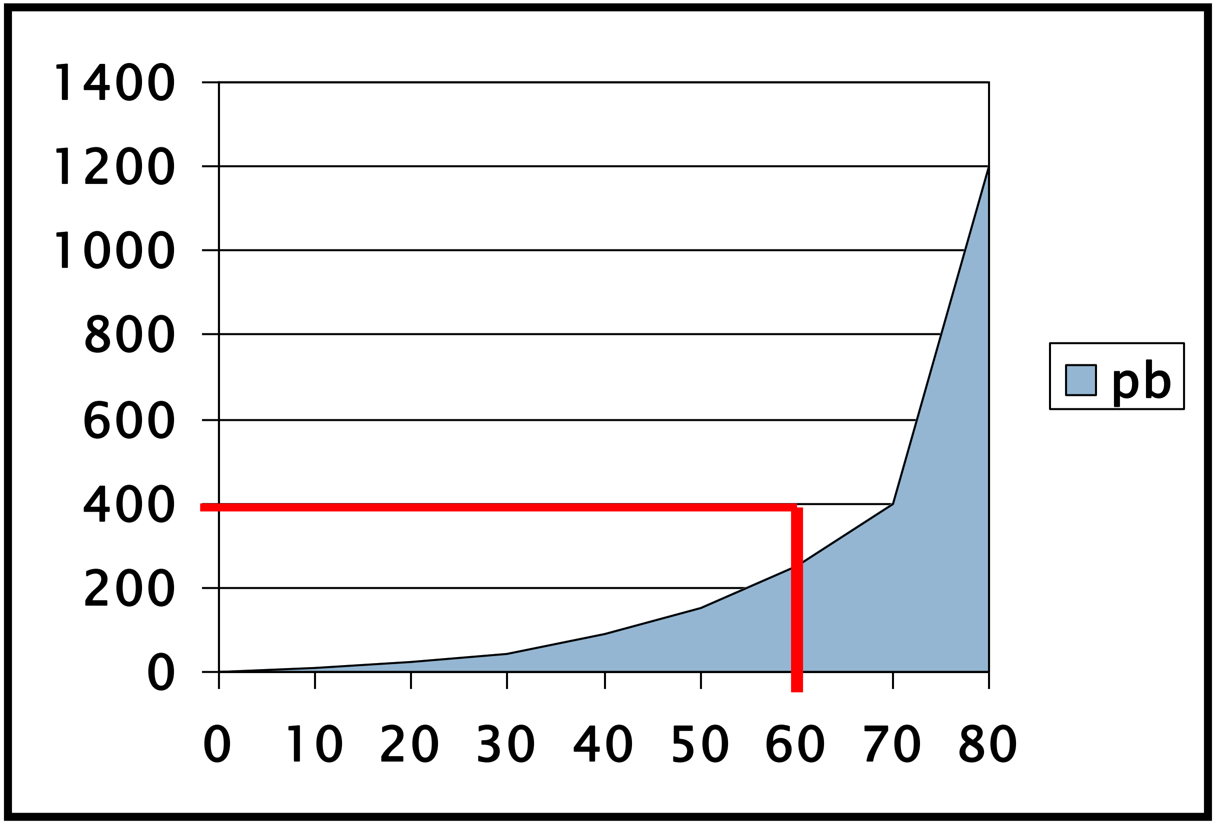
- Ok, **we cannot avoid an exponential behavior**
- For some instances, each NP Complete Problem will required an exponential time to be solved
- So, our only hope is to **shift the exponential** such that the problem is solvable for a size and a time that are acceptable

Shifting the exponential



We want to solve for n=60 in less than 400s

Shifting the exponential



We want to solve for n=60 in less than 400s

Sports scheduling models

# teams	# fails	Time (in s)
4	2	0.01
6	12	0.03
8	32	0.08
10	417	0.8
12	41	0.2
14	3,514	9.2
16	1,112	4.2
18	8,756	36
20	72,095	338
22	6,172,672	10h
24	6,391,470	12h

First Model

Second Model

# teams	# fails	Time (in s)
8	10	0.01
10	24	0.06
12	58	0.2
14	21	0.2
16	182	0.6
18	263	0.9
20	226	1.2
24	2702	10.5
26	5,683	26.4
30	11,895	138
40	2,834,754	6h

P ≠ NP

- We can only shift the exponential
- We will never solve the problem in general

CP and other techniques

- **It is not easy to compare CP with other techniques**
- **It is not easy to compare techniques aiming at solving NP-Complete problems**
  - **Because the problems are hard in general**
- Some instances are easy in CP and difficult with other techniques and conversely :
  - 2 examples: Sports scheduling (vs MIP) and Latin Square Completion (vs SAT)
  - SAT is able to solve some efficiently some instances of the Latin Square Completion but do not scale or not able to solve an empty problem

Comparison with CP

- It is difficult to define the difficulty of the resolution of some NP Complete problems
  - In theory: they are hard
  - In practice: the resolution uses a particular technique, so there is no absolute reference

P, NP and so what?

- **Problems in P or P = NP**: CP has almost no advantage
  - The propagation mechanism in itself is interesting (M. Wallace)
- **Problems in NP**: try to solve it to show the advantage of CP wrt the other techniques
- Interest of CP if we don't try to solve some problems?
  - Open question ☺

Problem resolution

- It is hard
- Common problems
  - Size
  - Intrinsic difficulty of some subparts
  - Combination of subparts

- Usually requires the implementation of a complex procedure divided into several steps

4 common steps

- Try to abstract some parts of the whole problem
  - Focus your attention on the difficult parts or on the combination of parts
- Work on smaller parts (benchmarking)
- Find good search strategies for the different parts
- Define a global model (combination of parts, scaling …)

Plan

- What kind of problem can we solve with CP?
- **4 pitfalls to avoid**
- Conclusion

4 common pitfalls

- Undivided model
- Rigid search
- Biased benchmarking
- Wrong abstraction

4 common pitfalls

- **Undivided model**
  - The global model is too much general
  - Split the resolution into different parts
- **Rigid search**
  - The search strategy is too much linked to a DFS
  - Wrong part must be left quickly
- **Biased benchmarking**
  - The results obtained for small size abstraction cannot be extrapolated for the whole problem
- **Wrong abstraction**
  - The part identified as relevant are not relevant
  - The resolution of some subparts could be improved

4 common pitfalls

- **Undivided model**
- Rigid search
- Biased benchmarking
- Wrong abstraction

Undivided model

- Either we directly deal with the whole problem in one step or we try to decompose it
- The decomposition of the problem is a classical idea in MIP
  - Column generation
  - Bender's decomposition
  - Lagrangian relaxation (close to abstraction)

Undivided model

- **Solving some subparts and recombine them for solving the whole problem**

Pre-resolution of a part of a problem

- Configuration Problem:
  - 5 types of components: {glass, plastic, steel, wood, copper}
  - 3 types of bins: {red, blue, green} whose capacity is red 5, blue 5, green 6
  - Constraints:
    - red can contain glass, cooper, wood
    - blue can contain glass, steel, cooper
    - green can contain plastic, copper, wood
    - wood require plastic; glass exclusive copper
    - red contains at most 1 of wood
    - green contains at most 2 of wood

    For all the bins there is either no plastic or at least 2 plastic
  - Given an initial supply of 12 of glass, 10 of plastic, 8 of steel, 12 of wood and 8 of copper;
    what is the minimum total number of bins?

Pre-resolution of a part of a problem

	#bk	time
standard model	1,361,709	430
GAC+allowed	12,659	9.7

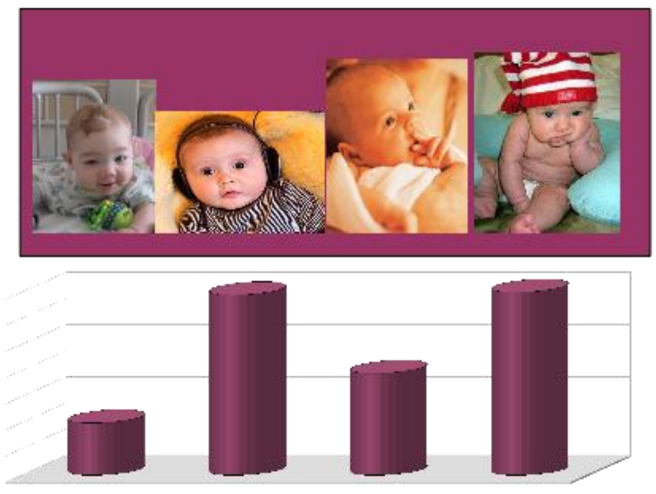
Undivided model

- **Solving some subparts and recombine them for solving the whole problem**

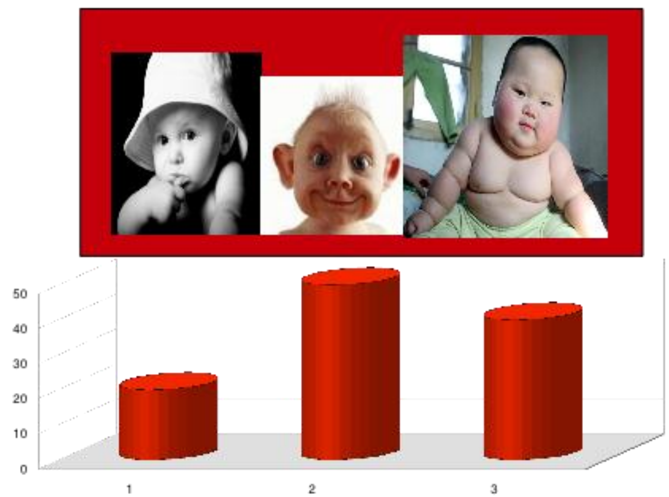
- « Scalable Load Balancing in Nurse to Patient Assignment Problems », P. Schaus, P Van Hentenryck, J-C Régin, CPAIOR 09

Description of the Problem

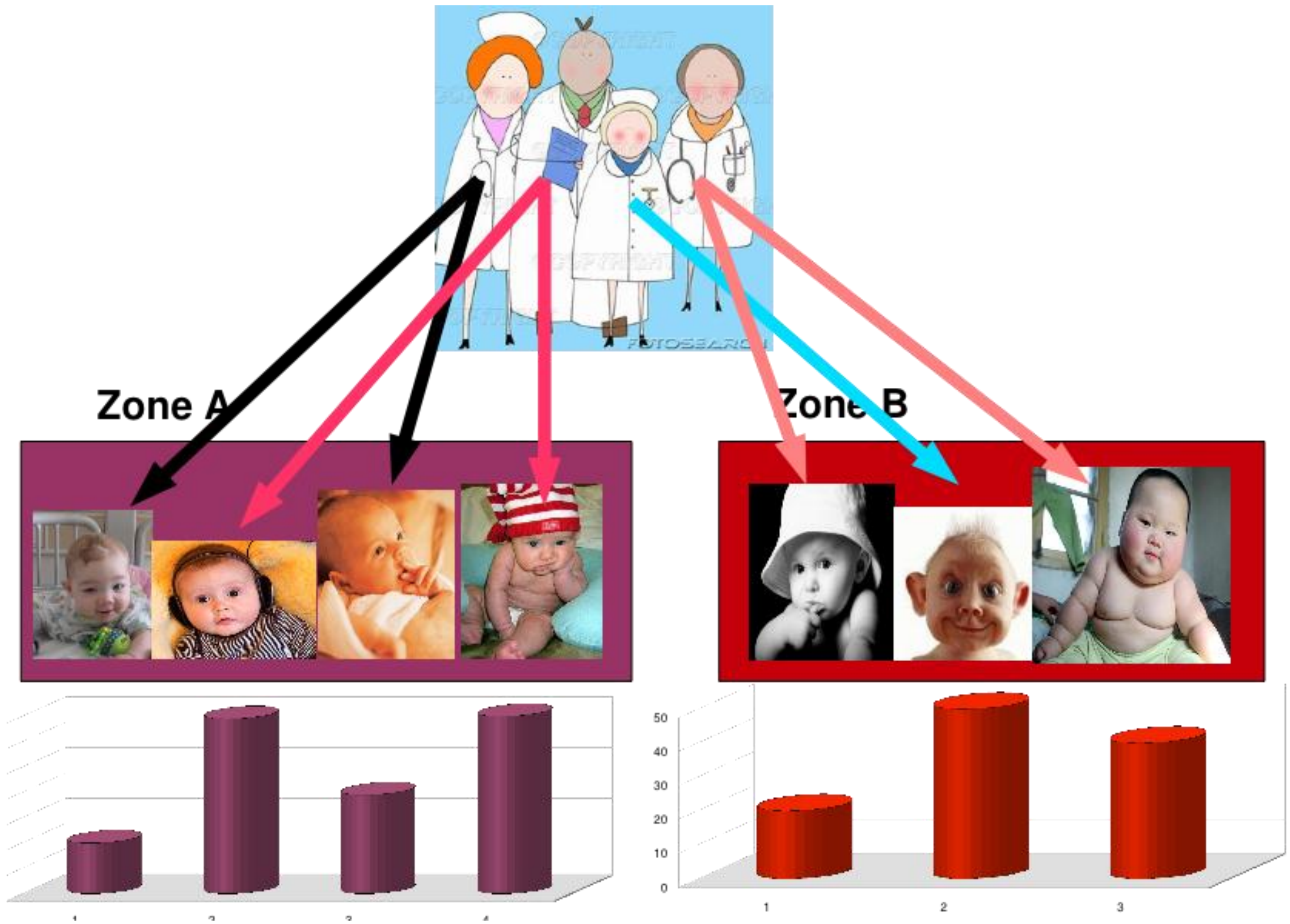
Zone A



Zone B



Description of the Problem



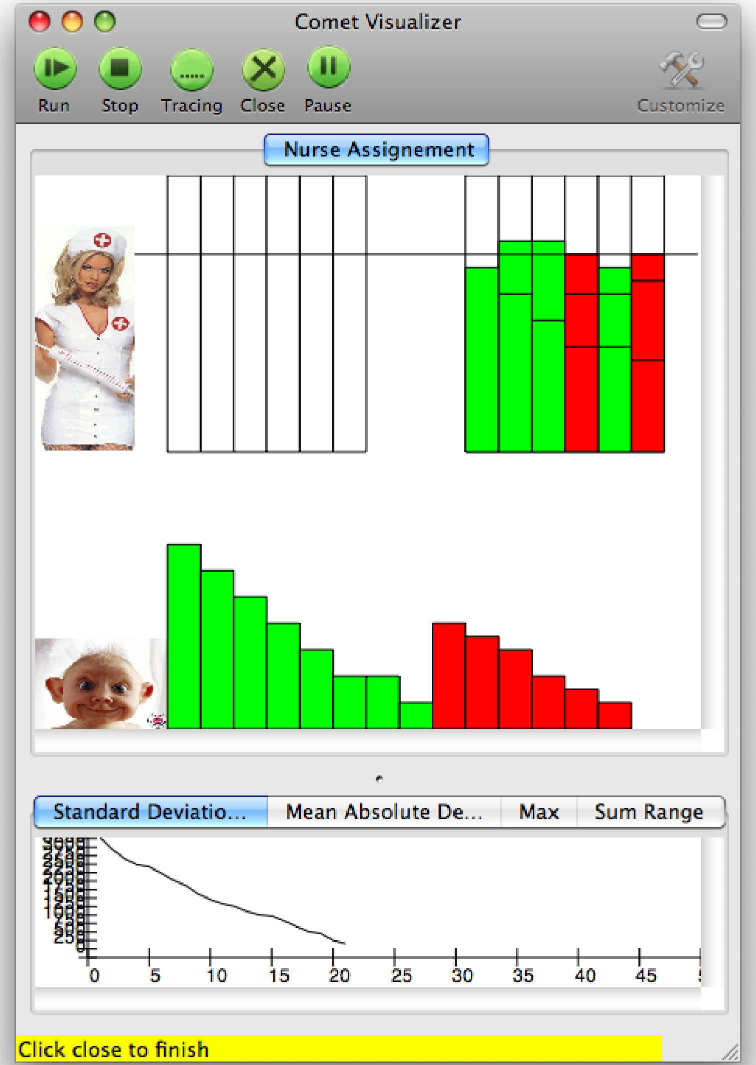
Description of the problem

- The constraints
  - Each patient must be allocated to one nurse.
  - One nurse can take at most 3 patients and at least 1.
  - One nurse can only work in one zone.
- The objective
  - Assign patients to nurses such that the nurse workload is balanced.

Assigning patients to nurses in neonatal intensive care,

C Mullinax and M Lawley, Journal of the Operational Research Society, 2002

Minimization of the variance



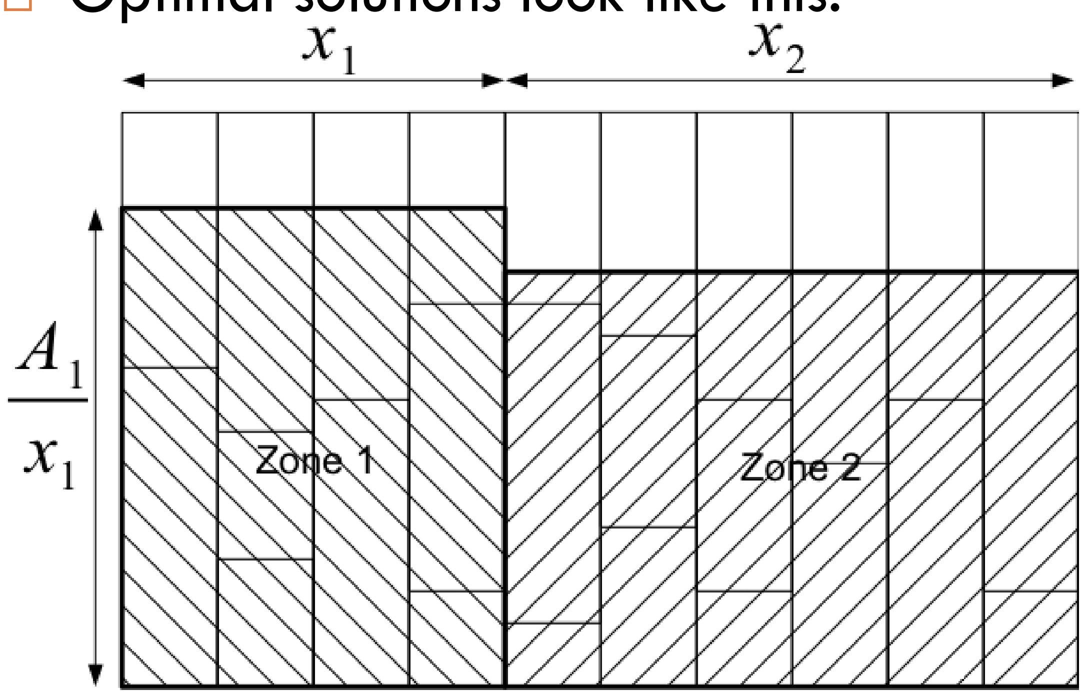
Results (2 zones instances)

- All solved optimally within 20 minutes (the MIP model cannot). m = #nurses; n = #infants

m	n	#fails	time(s)	avg workload	sd. workload
11	28	511095	170.2	86.09	2.64
11	29	1126480	302.0	80.27	1.76
10	26	104931	24.7	76.50	2.29
12	30	259147	136.5	83.42	1.93
10	28	2990450	1138.5	91.80	6.84
10	26	779969	206.9	88.40	2.29
12	29	555243	198.2	80.08	2.72
10	27	931858	343.9	90.60	5.33
10	25	1616689	434.5	82.70	7.32
8	22	4160	1.2	87.50	3.12

Observations for improving the model

- The number of nurses assigned to each zone has a huge influence on the quality of the balancing.
- Most of the inbalance comes from the inter-zone workloads. Very good balance inside each zone.
- Optimal solutions look like this:



A_i: acuity of the zone i

x_i: number of nurses in zone i

The Idea: A two steps approach

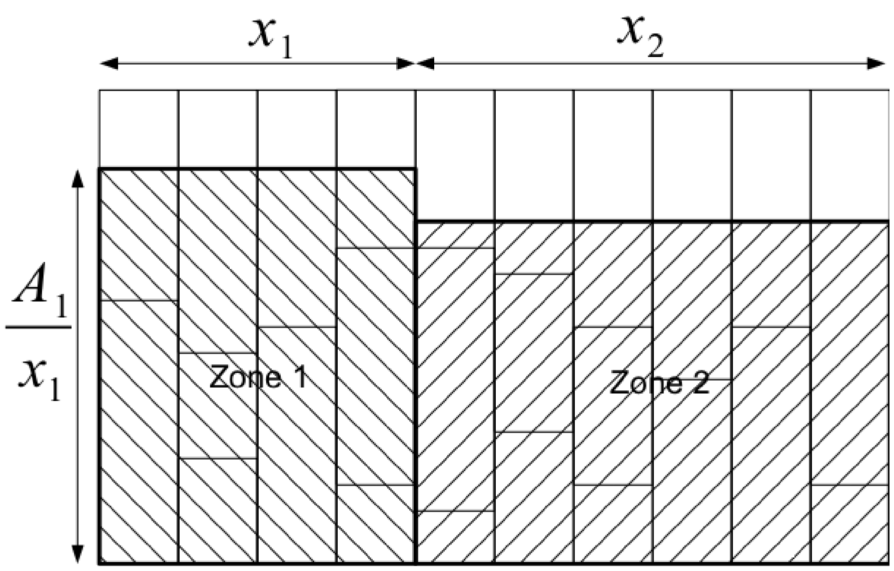
- We consider a relaxation of the initial problem
  - Compute the number of nurses assigned to each zone.
  - A patient can only take the pre-computed nurses (modification of the domains of variables).

- Optimal solutions of this relaxed problem are very close to optimal solutions of the general problem

→ How to compute the number of nurses assigned to each zone ?

Compute the number of nurses assigned to each zone

- We solve the optimally of this problem in O(p*m) with a greedy algorithm. (p = #zones; m = #nurses)



$$\min \sum_{k=1}^{p} x_k \cdot \left(\frac{A_k}{x_k} - \sum_{j=1}^{p} \frac{A_j}{m} \right)^2$$

$$s.t. \quad \sum_{k=1}^{p} x_i = m$$

$$x_k \in Z_0^+$$

A_i: acuity of the zone I (GIVEN)

x_i: number of nurses in zone I (UNKNOWNS)

Previous results (2 zones instances)

- All solved optimally within 20 minutes (the MIP model cannot). m = #nurses; n = #infants

m	n	#fails	time(s)	avg workload	sd. workload
11	28	511095	170.2	86.09	2.64
11	29	1126480	302.0	80.27	1.76
10	26	104931	24.7	76.50	2.29
12	30	259147	136.5	83.42	1.93
10	28	2990450	1138.5	91.80	6.84
10	26	779969	206.9	88.40	2.29
12	29	555243	198.2	80.08	2.72
10	27	931858	343.9	90.60	5.33
10	25	1616689	434.5	82.70	7.32
8	22	4160	1.2	87.50	3.12

New results on 2 zones instances

- Less than 10 seconds (m: #nurses; n = #infants)

m	n	#fails	time(s)	avg workload	sd. workload	lb. sd.
11	28	25385	4.5	86.09	2.64	2.23
11	29	4916	1.4	80.27	1.76	0.62
10	26	458	0.1	76.50	2.29	2.29
12	30	17558	6.7	83.42	1.93	1.19
10	28	29865	4.8	91.80	6.84	6.81
10	26	3705	1.0	88.40	2.29	1.43
12	29	6115	1.2	80.08	2.72	0.64
10	27	1109	0.4	90.60	5.33	5.22
10	25	3299	0.6	82.70	7.32	6.71
8	22	127	0.0	87.50	3.12	3.04

Results on 3 zones instances

- 6/10 instances solved optimally (m: #nurses; n = #infants)

sol	m	n	#fails	time(s)	avg. wl	sd. wl	lb. sd.
1	15	42	19488	5.3	84.20	3.04	2.93
1	18	43	3619310	919.2	79.78	5.84	5.49
0	17	43	9023072	1800.0	81.41	4.75	3.45
1	17	42	483032	106.9	83.82	5.65	5.59
0	18	43	7124370	1800.0	81.00	7.11	4.94
1	14	38	590971	145.2	85.36	3.08	2.16
0	19	48	3786580	1800.0	87.42	3.18	2.30
1	16	44	3888210	839.8	84.88	6.70	6.39
0	19	49	5697272	1800.0	86.00	2.70	1.95
1	17	41	61250	17.3	82.18	3.40	3.07

Good news: The decomposition can work

- Given a precomputation of the number of nurses for each zone:

minimizing the variance among all the nurses	=	minimizing the variance in each zone separately

2 Steps Approach with Decomposition

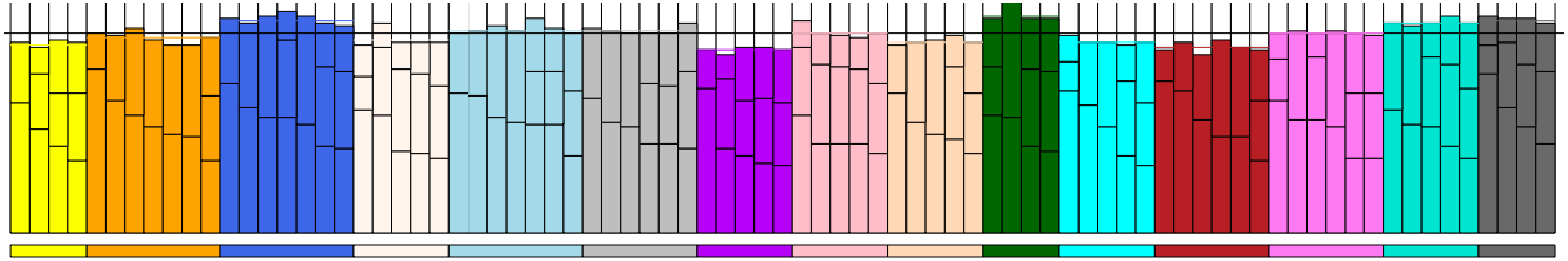
- Compute the number of nurses assigned to each zone.
- Solve independently the problems inside each zone.

New results on the 3 zones instances

- Easy now (less than 3 seconds) (m: #nurses; n = #infants)

m	n	#fails	time(s)	avg workload	sd. workload	lb. sd.
15	42	203	0.1	84.20	3.04	2.93
18	43	608	0.1	79.78	5.84	5.49
17	43	8134	1.1	81.41	4.46	3.45
17	42	345	0.1	83.82	5.65	5.59
18	43	24994	3.2	81.00	5.77	4.94
14	38	151	0.0	85.36	3.08	2.16
19	48	3695	0.8	87.42	3.07	2.30
16	44	384	0.1	84.88	6.70	6.39
19	49	2056	0.4	86.00	2.49	1.95
17	41	776	0.2	82.18	3.40	3.07

We can even solve 15 zones instances!



The problem

- n teams and n-1 weeks and n/2 periods
- every two teams play each other exactly once
- every team plays one game in each week
- no team plays more than twice in the same period

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Period 1	0 vs 1	0 vs 2	4 vs 7	3 vs 6	3 vs 7	1 vs 5	2 vs 4
Period 2	2 vs 3	1 vs 7	0 vs 3	5 vs 7	1 vs 4	0 vs 6	5 vs 6
Period 3	4 vs 5	3 vs 5	1 vs 6	0 vs 4	2 vs 6	2 vs 7	0 vs 7
Period 4	6 vs 7	4 vs 6	2 vs 5	1 vs 2	0 vs 5	3 vs 4	1 vs 3

CP model: variables

For each slot: 2 variables represent the teams and 1 variable represents the match are defined

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Period 1	0 vs 1	0 vs 2	4 vs 7	3 vs 6	3 vs 7	1 vs 5	2 vs 4
Period 2	2 vs 3	1 vs 7	0 vs 3	5 vs 7	1 vs 4	0 vs 6	5 vs 6
Period 3	4 vs 5	3 vs 5	1 vs 6	0 vs 4	2 vs 6	2 vs 7	0 vs 7
Period 4	6 vs 7	4 vs 6	2 vs 5	1 vs 2	0 vs 5	3 vs 4	1 vs 3

1 vs 6

M33 variable (M33=12)

T33a variable (T33a=6)

T33h variable (T33h=1)

Mij=1 <=> 0 vs 1 or 1 vs 0

Mij=12 <=> 1 vs 6 or 6 vs1

CP model: T variables

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Period 1	T11h vs T11a	T12h vs T12a	T13h vs T13a	T14h vs T14a	T15h vs T15a	T16h vs T16a	T17h vs T17a
Period 2	T21h vs T21a	T22h vs T22a	T23h vs T23a	T24h vs T24a	T25h vs T25a	T26h vs T26a	T27h vs T27a
Period 3	T31h vs T31a	T32h vs T32a	T33h vs T33a	T34h vs T34a	T35h vs T35a	T36h vs T36a	T37h vs T37a
Period 4	T41h vs T41a	T42h vs T42a	T43h vs T43a	T44h vs T44a	T45h vs T45a	T46h vs T46a	T47h vs T47a

$$D(Tija)=[1,n-1]$$
$$D(Tijh)=[0,n-2]$$

$$Tijh < Tija$$

CP model: M variables

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Period 1	M11	M12	M13	M14	M15	M16	M17
Period 2	M21	M22	M23	M24	M25	M26	M27
Period 3	M31	M32	M33	M34	M35	M36	M37
Period 4	M41	M42	M43	M44	M45	M46	M47

$$D(M_{ij})=[1,n(n-1)/2]$$

CP model: constraints

- n teams and n-1 weeks and n/2 periods
- every two teams play each other exactly once
- every team plays one game in each week
- no team plays more than twice in the same period

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Period 1	M11	M12	M13	M14	M15	M16	M17
Period 2	M21	M22	M23	M24	M25	M26	M27
Period 3	M31	M32	M33	M34	M35	M36	M37
Period 4	M41	M42	M43	M44	M45	M46	M47

Alldiff constraints defined on M variables

CP model: constraints

- n teams and n-1 weeks and n/2 periods
- every two teams play each other exactly once
- every team plays one game in each week
- no team plays more than twice in the same period

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Period 1	T11h vs T11a	T12h vs T12a	T13h vs T13a	T14h vs T14a	T15h vs T15a	T16h vs T16a	T17h vs T17a
Period 2	T21h vs T21a	T22h vs T22a	T23h vs T23a	T24h vs T24a	T25h vs T25a	T26h vs T26a	T27h vs T27a
Period 3	T31h vs T31a	T32h vs T32a	T33h vs T33a	T34h vs T34a	T35h vs T35a	T36h vs T36a	T37h vs T37a
Period 4	T41h vs T41a	T42h vs T42a	T43h vs T43a	T44h vs T44a	T45h vs T45a	T46h vs T46a	T47h vs T47a

For each week w:
Alldiff constraint defined
on {Tpwh, p=1..4} U {Tpwa, p=1..4}

CP model: constraints

- n teams and n-1 weeks and n/2 periods
- every two teams play each other exactly once
- every team plays one game in each week
- no team plays more than twice in the same period

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Period 1	T11h vs T11a	T12h vs T12a	T13h vs T13a	T14h vs T14a	T15h vs T15a	T16h vs T16a	T17h vs T17a
Period 2	T21h vs T21a	T22h vs T22a	T23h vs T23a	T24h vs T24a	T25h vs T25a	T26h vs T26a	T27h vs T27a
Period 3	T31h vs T31a	T32h vs T32a	T33h vs T33a	T34h vs T34a	T35h vs T35a	T36h vs T36a	T37h vs T37a
Period 4	T41h vs T41a	T42h vs T42a	T43h vs T43a	T44h vs T44a	T45h vs T45a	T46h vs T46a	T47h vs T47a

For each period p:
Global cardinality constraint defined on
{Tpwh, w=1..7} U {Tpwa, w=1..7}
every team t is taken at most 2

CP model: constraints

- For each slot the two T variables and the M variable must be linked together; example:
  M12 = game T12h vs T12a
- For each slot we add Cij a ternary constraint defined on the two T variables and the M variable; example:
  C12 defined on {T12h,T12a,M12}

- Cij are defined by the list of allowed tuples:
  for n=4: {(0,1,1),(0,2,2),(0,3,3),(1,2,4),(1,3,5),(2,3,6)}
  (1,2,4) means game 1 vs 2 is the game number 4
- All these constraints have the same list of allowed tuples
- Efficient arc consistency algorithm for this kind of constraint is known

First model

Introduction of a dummy column

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Dummy
Period 1	0 vs 1	0 vs 2	4 vs 7	3 vs 6	3 vs 7	1 vs 5	2 vs 4	. vs .
Period 2	2 vs 3	1 vs 7	0 vs 3	5 vs 7	1 vs 4	0 vs 6	5 vs 6	. vs .
Period 3	4 vs 5	3 vs 5	1 vs 6	0 vs 4	2 vs 6	2 vs 7	0 vs 7	. vs .
Period 4	6 vs 7	4 vs 6	2 vs 5	1 vs 2	0 vs 5	3 vs 4	1 vs 3	. vs .

First model

Introduction of a dummy column

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Dummy
Period 1	0 vs 1	0 vs 2	4 vs 7	3 vs 6	3 vs 7	1 vs 5	2 vs 4	5 vs 6
Period 2	2 vs 3	1 vs 7	0 vs 3	5 vs 7	1 vs 4	0 vs 6	5 vs 6	. vs .
Period 3	4 vs 5	3 vs 5	1 vs 6	0 vs 4	2 vs 6	2 vs 7	0 vs 7	. vs .
Period 4	6 vs 7	4 vs 6	2 vs 5	1 vs 2	0 vs 5	3 vs 4	1 vs 3	. vs .

We can prove that:

- each team occurs exactly twice for each period

First model

Introduction of a dummy column

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Dummy
Period 1	0 vs 1	0 vs 2	4 vs 7	3 vs 6	3 vs 7	1 vs 5	2 vs 4	5 vs 6
Period 2	2 vs 3	1 vs 7	0 vs 3	5 vs 7	1 vs 4	0 vs 6	5 vs 6	2 vs 4
Period 3	4 vs 5	3 vs 5	1 vs 6	0 vs 4	2 vs 6	2 vs 7	0 vs 7	1 vs 3
Period 4	6 vs 7	4 vs 6	2 vs 5	1 vs 2	0 vs 5	3 vs 4	1 vs 3	0 vs 7

We can prove that:

- each team occurs exactly twice for each period
- each team occurs exactly once in the dummy column

First model: strategies

- Break symmetries: 0 vs w appears in week w
- Teams are instantiated:
  - the most instantiated team is chosen
  - the slots that has the less remaining possibilities (Tijh or Tija is minimal) is instantiated with that team

First model: results

# teams	# fails	Time (in s)
4	2	0.01
6	12	0.03
8	32	0.08
10	417	0.8
12	41	0.2
14	3,514	9.2
16	1,112	4.2
18	8,756	36
20	72,095	338
22	6,172,672	10h
24	6,391,470	12h

MIPLIB

Second model

- Break symmetry: 0 vs 1 is the first game of the dummy column

Second model

- Break symmetry: 0 vs 1 is the first game of the dummy column
- 1) Find a round-robin. Define all the games for each column (except for the dummy)
  - Alldiff constraint on M is satisfied
  - Alldiff constraint for each week is satisfied

Second model

- Break symmetry: 0 vs 1 is the first game of the dummy column
- 1) Find a round-robin. Define all the games for each column (except for the dummy)
  - Alldiff constraint on M is satisfied
  - Alldiff constraint for each week is satisfied
- 2) set the games in order to satisfy constraints on periods. If no solution go to 1)

Second model: strategy

M variables are instantiated

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Period 1	M11	M12	M13	M14	M15	M16	M17
Period 2	M21	M22	M23	M24	M25	M26	M27
Period 3	M31	M32	M33	M34	M35	M36	M37
Period 4	M41	M42	M43	M44	M45	M46	M47

Second model: strategy

M variables are instantiated

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Period 1	M11	M12	M13	M14	M15	M16	M17
Period 2	M21	M22	M23	M24	M25	M26	M27
Period 3	M31	M32	M33	M34	M35	M36	M37
Period 4	M41	M42	M43	M44	M45	M46	M47

Second model: strategy

M variables are instantiated

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Period 1	M11	M12	M13	M14	M15	M16	M17
Period 2	M21	M22	M23	M24	M25	M26	M27
Period 3	M31	M32	M33	M34	M35	M36	M37
Period 4	M41	M42	M43	M44	M45	M46	M47

Second model: strategy

M variables are instantiated

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Period 1	M11	M12	M13	M14	M15	M16	M17
Period 2	M21	M22	M23	M24	M25	M26	M27
Period 3	M31	M32	M33	M34	M35	M36	M37
Period 4	M41	M42	M43	M44	M45	M46	M47

Second model: strategy

M variables are instantiated

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Period 1	M11	M12	M13	M14	M15	M16	M17
Period 2	M21	M22	M23	M24	M25	M26	M27
Period 3	M31	M32	M33	M34	M35	M36	M37
Period 4	M41	M42	M43	M44	M45	M46	M47

Sports scheduling models

# teams	# fails	Time (in s)
4	2	0.01
6	12	0.03
8	32	0.08
10	417	0.8
12	41	0.2
14	3,514	9.2
16	1,112	4.2
18	8,756	36
20	72,095	338
22	6,172,672	10h
24	6,391,470	12h

First Model

Second Model

# teams	# fails	Time (in s)
8	10	0.01
10	24	0.06
12	58	0.2
14	21	0.2
16	182	0.6
18	263	0.9
20	226	1.2
24	2702	10.5
26	5,683	26.4
30	11,895	138
40	2,834,754	6h

4 common pitfalls

- Undivided model
- **Rigid search**
- Biased benchmarking
- Wrong abstraction

Rigid search

- I notice that there are 2 kinds of people in CP
  - Those focused on the search strategies, who « thinks » strategies
  - Those focused on constraints, who « thinks » constraints

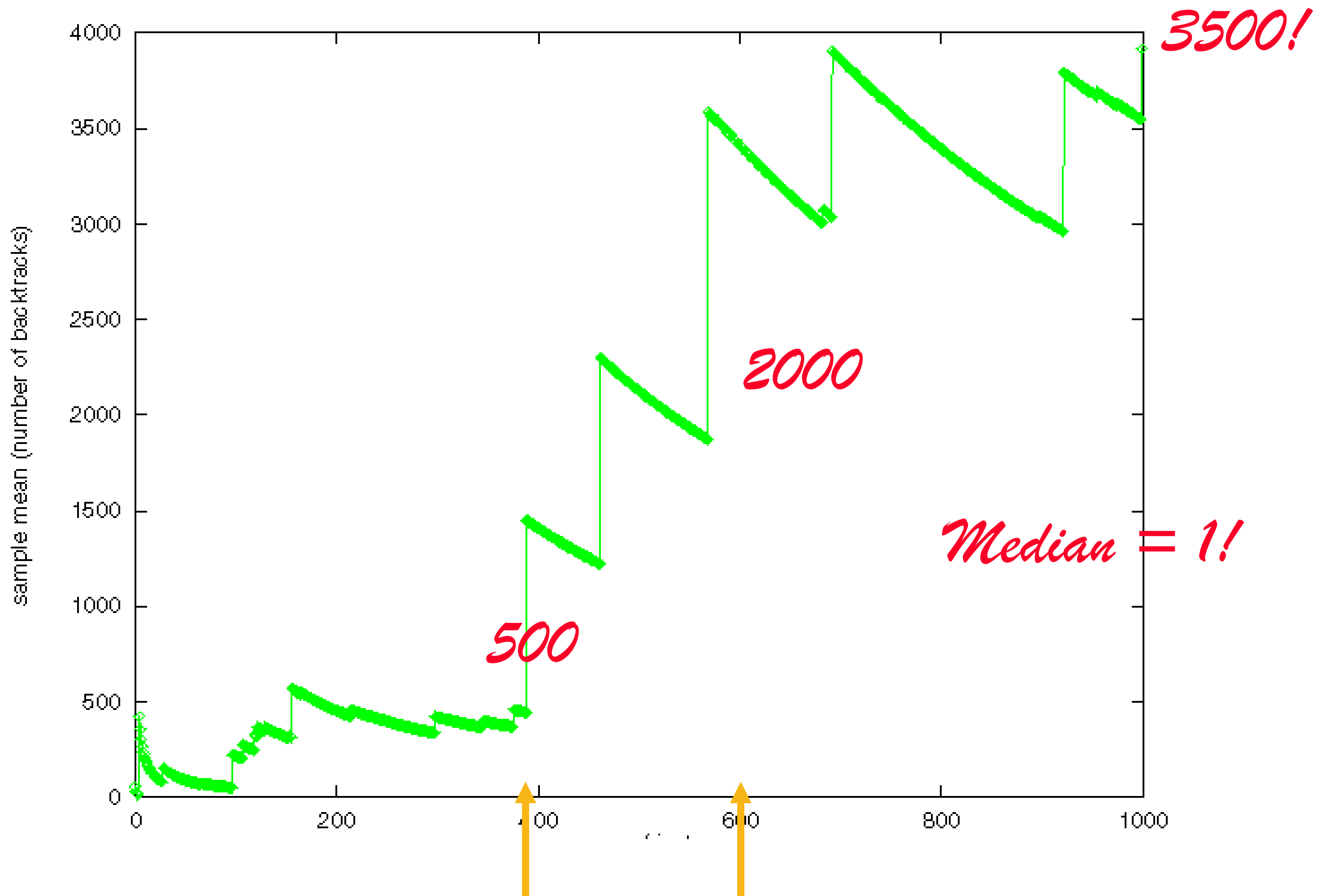
- I am not a big fan of search strategy

Rigid Search

- We can deal a lot and invent a lot of strategies fro solving a problem
- Random-restart is a method
  - performing very well
  - that can be used with any strategy

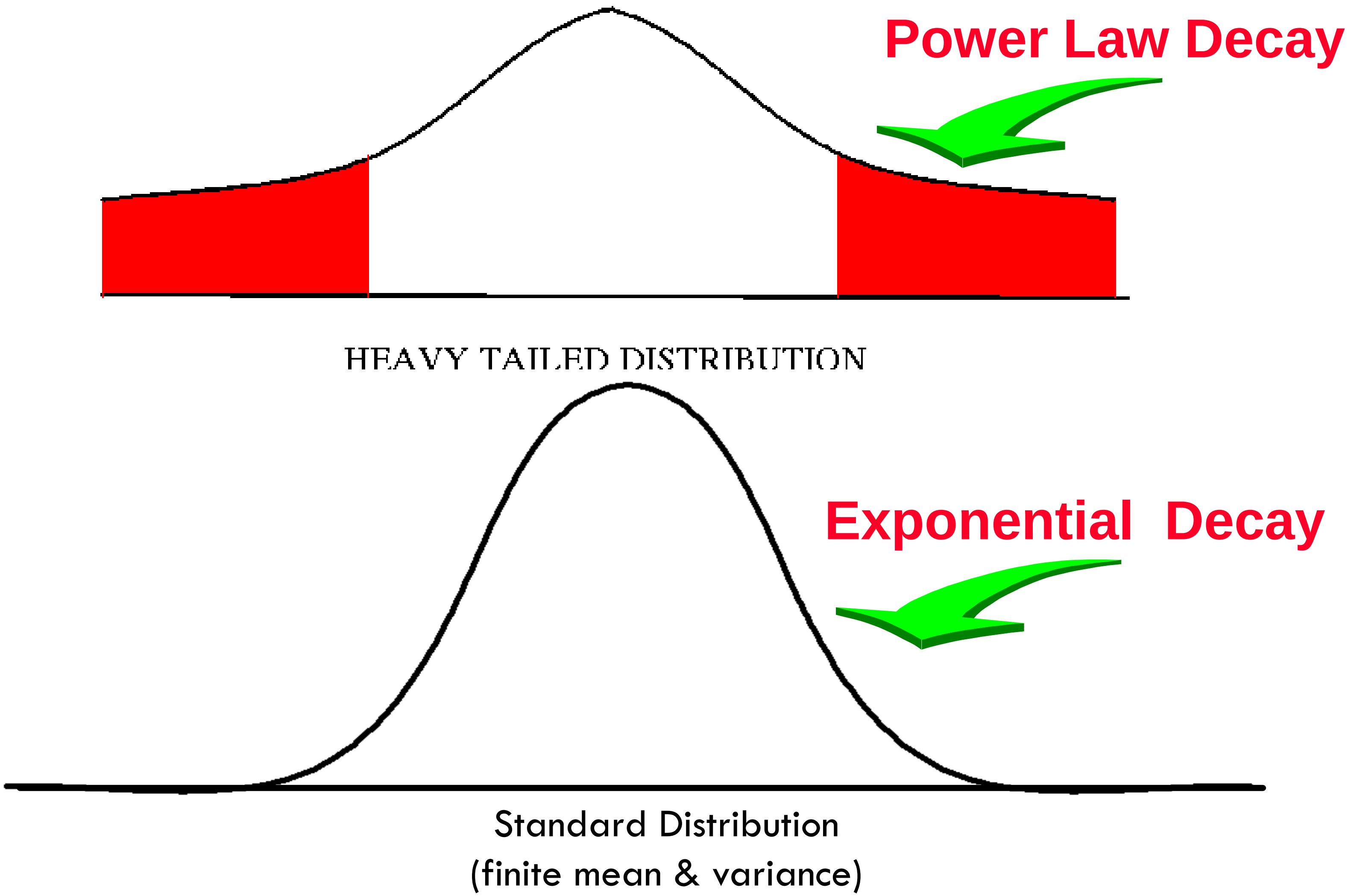
- Slides and work of Carla Gomes

Quasigroup completion

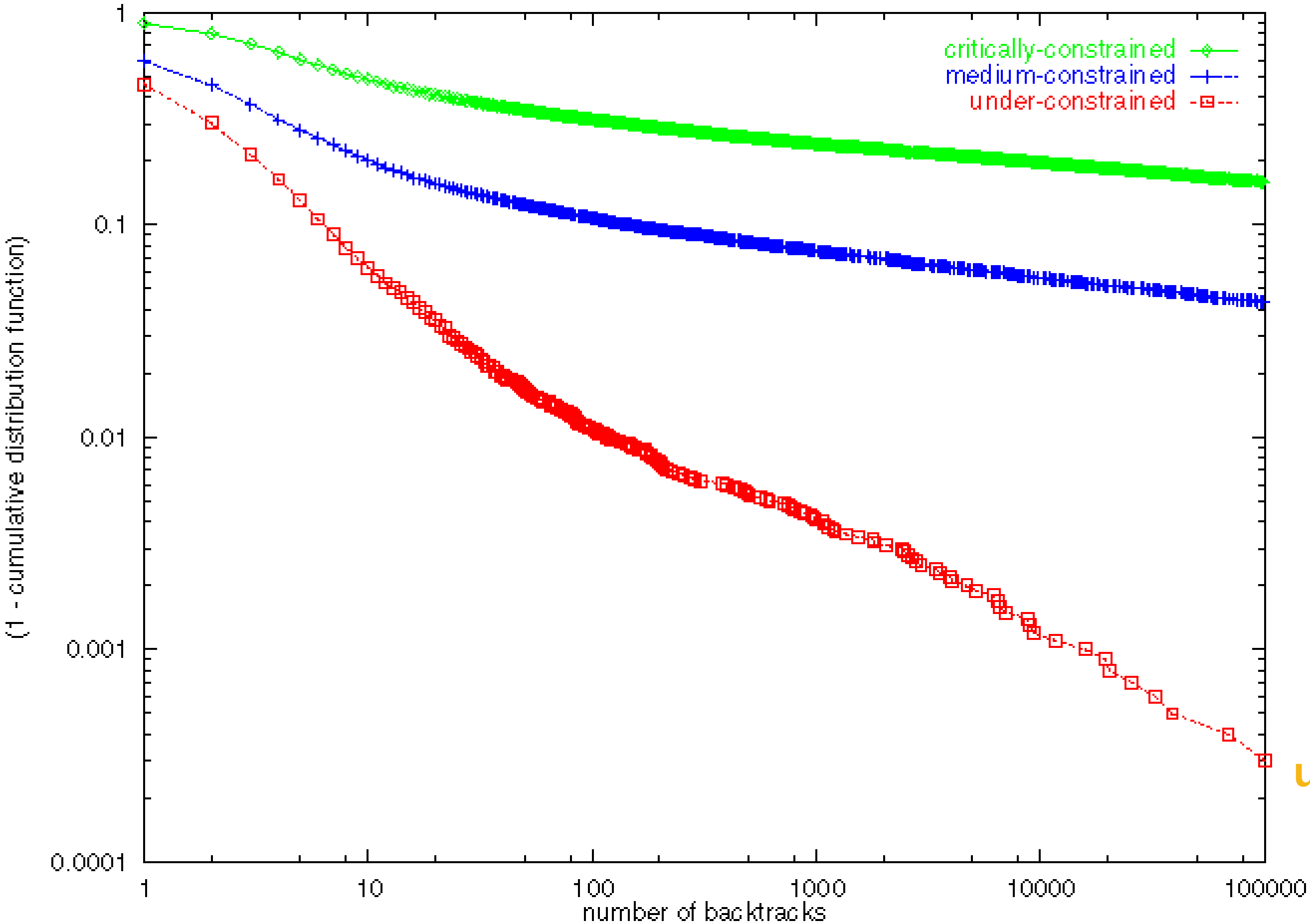


Erratic Mean Cost Behavior

Heavy tail distribution (Pareto 1920)



Quasigroup Resolution



18% unsolved

0.002% unsolved

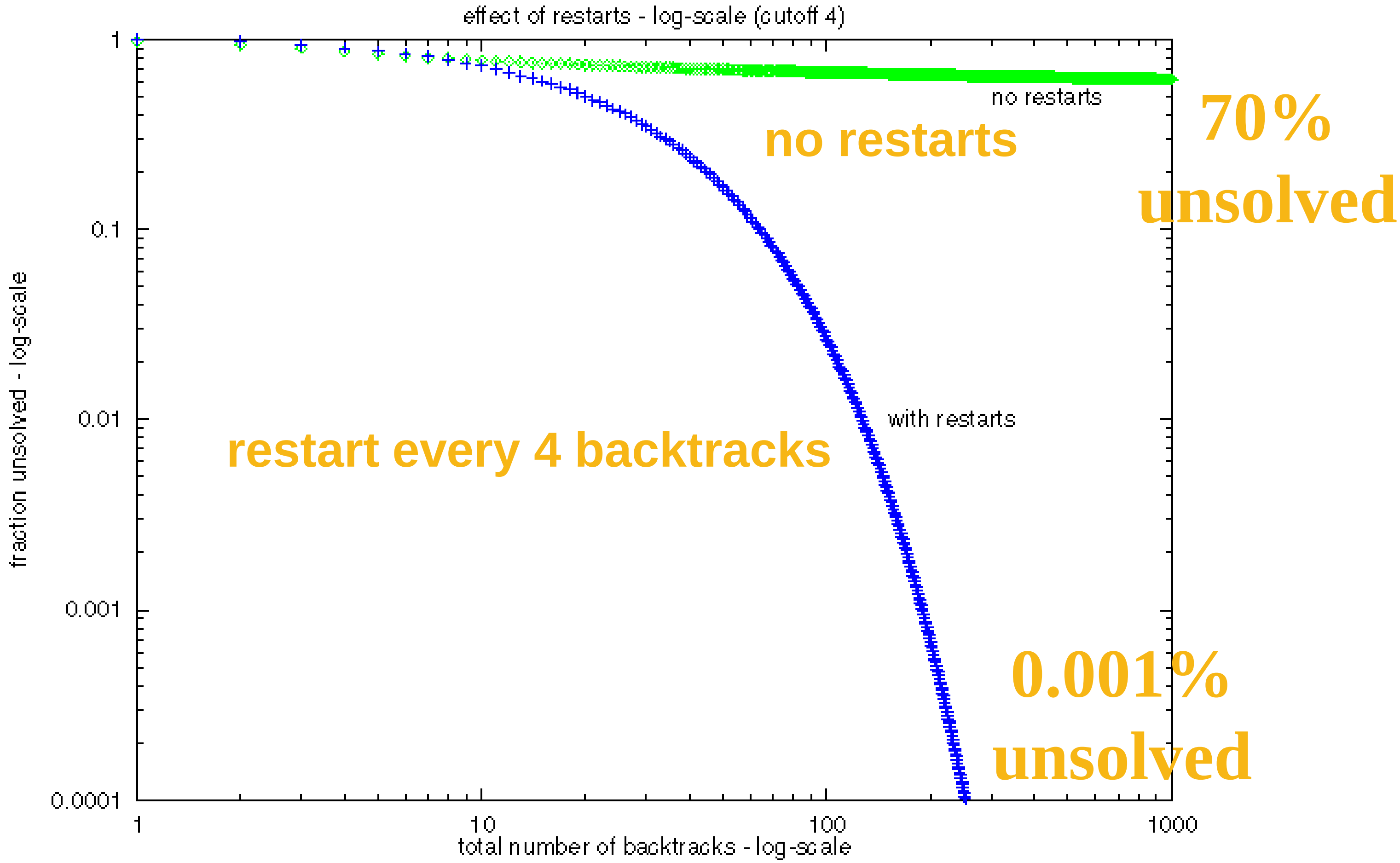
Heavy-Tailed Behavior (log-log scale)

Exploiting Heavy-Tailed behavior

- Heavy Tailed behavior has been observed in several domains: QCP, Graph Coloring, Planning, Scheduling, Circuit synthesis, Decoding, etc.

- Consequence for algorithm design: Use **restarts runs** to exploit the extreme variance performance.

Restarts



Effect of restarts (cutoff 4)

Restarts

- Restarts **provably** eliminate heavy-tailed behavior. (Gomes et al. 97, Hoos 99, Horvitz 99, Huberman, Lukose and Hogg 97, Karp et al 96, Luby et al. 93, Rish et al. 97)
- This idea is implemented in ILOG CPOptimizer and it works!
- It is also implemented in ILOG Cplex under the name "Dynamic search"
- Main advantage: it is much more robust

4 common pitfalls

- Undivided model
- Rigid search
- **Biased benchmarking**
- Wrong abstraction

Biased Benchmarking

- The identification of an interesting subpart is a first step. The advantage is two fold:
  - We can focus our attention on a difficult part that we need to solve
  - We can work on smaller problems
- Be careful: it is also important to design some benchmarks from which we expect to derive general considerations

Biased Benchmarking

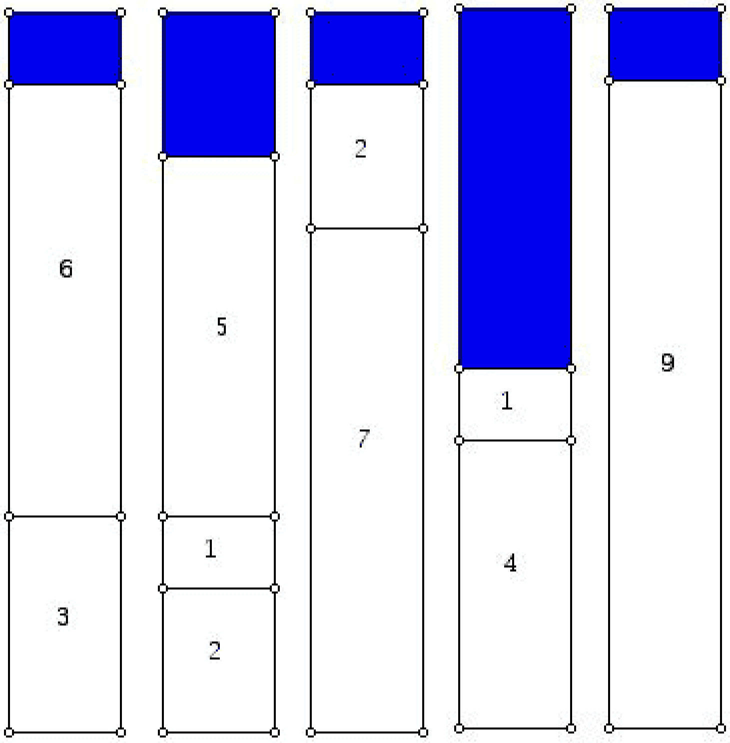
- Represent the fact that the results obtained from a benchmark can be not representative of the whole, problem
- Make sure that you can **extrapolate your results**!

Relevant and realistic Instances

- Benchmarking is serious and not easy
- The name of a problem is not enough (e.g. quasigroup completion problem (QCP), latin square).
  - It is an hard task to find hard QCP instances for small values (<100 or < 200).
  - However, there are some exceptionally hard instances (B. Smith) for n=35
- Avoid considering empty instances if you want to be able to generalize your results
- Example of biased benchmarking: the bin packing problem ("Comparison of Bin Packing models", JC Régin, M. Rezgui, A. Malapert, AIDC workshop at AAAI-11)

Bin packing problem

- Bin Packing Problem
- Range different sizes *items* in a number of *bins* with a limited capacity



Instances

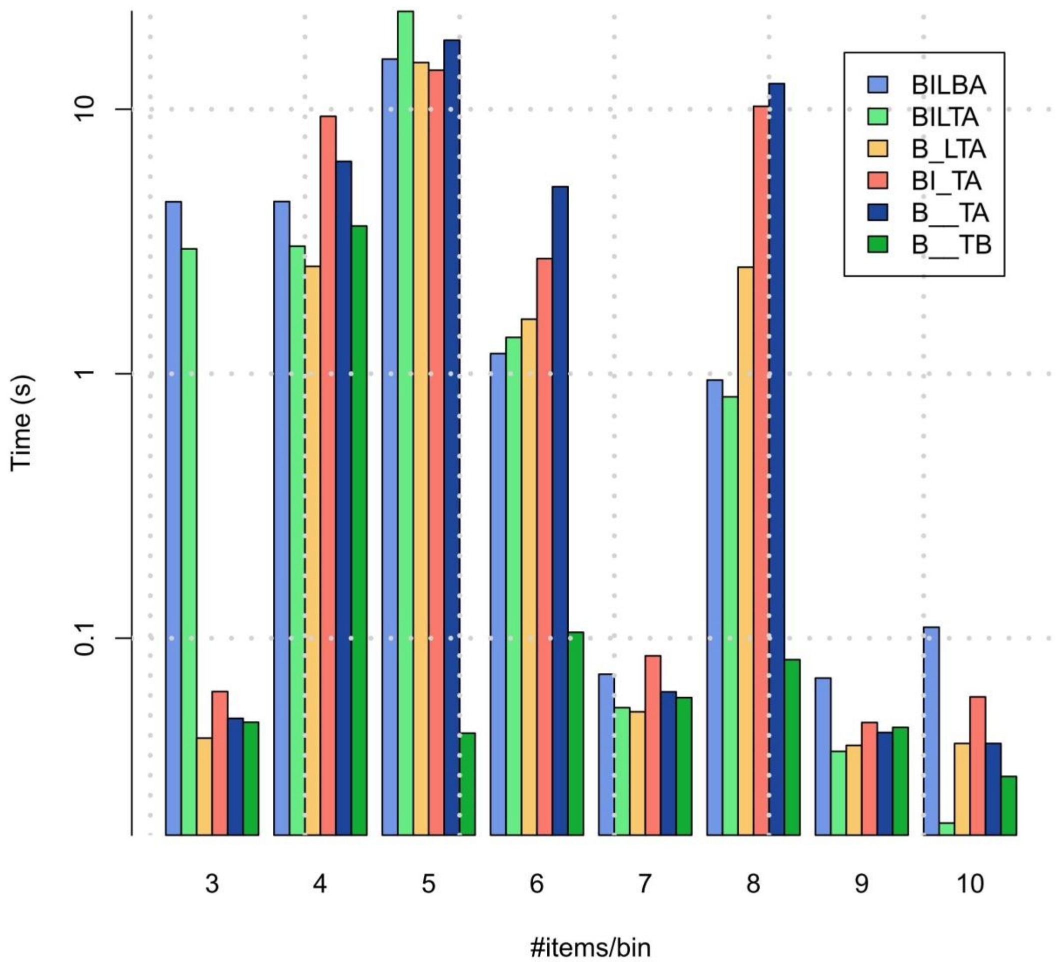
- Falkenauer, Scholl and Korf mainly consider instances with about 3 items per bins (Korf explicitly build instances with 3 items per bins)
- This lead to efficient methods.
- Some lower bounds may be used (Martello and Toth consider items whose size is more than half or a third of the bin capacity)

- I. Gent solved by hand some instances claimed to be difficult by Faulkenauer. He criticized the proposed instances

Instances

- I. Gent is right
- It is difficult to extrapolate from these instances
  - 4 items per bins are more difficult
  - Then, the difficulties of the instances decrease (in general) when the number of item per bin is increased!

Instances



Time (s)
10
1
0.1
3
4
5
6
7
8
9
10
#items/bin
BILBA
BILTA
B_LTA
BI_TA
B__TA
B__TB

Sum constraint

- We have seen that the number of items per bin is quite important
- We made an interesting remark about this
  - Consider Diophantine equation

Sum constraint

- Diophantine equation ax + by =c, solved for natural numbers
  - **Paoli's Theorem**
    - q is the quotient of c/ab and r the remaining part of c/ab
    - The number of positives (or =0) integer solutions of the equation $ax + by = c$ is q or q+1 depending on the fact that the equation $ax + by = r$ admits one or zero solution.
  - We set gcd(a,b)=1
    - If c > ab : always a solution : no (or almost no) filtering!
    - if c < ab : half of the values have a solution: almost no filtering

Sum constraint

- Diophantine equation $ax + by + cz = d$
- Is equivalent to
  - $ax + by = d - c$ OR $ax + by = d - 2c$ OR …
- The density of solution increases! We have less and less chance to not be able to satisfy the constraint…

- If our results are based on a sum with only few variables then we cannot extrapolate when we will have a lot of variables!

4 common pitfalls

- Undivided model
- Rigid search
- Biased benchmarking
- **Wrong abstraction**

Wrong abstraction

- It is difficult to identify relevant subparts of a problems, that is the one on which we should first focus our attention
- The wrong abstraction pitfall is the consideration of a subpart which is interesting but which is not relevant for the resolution of the whole problem

- Considered in 1997 by C. Bessière and J-C Régin (CP'97)
  - Before writing a filtering algorithm we should study if it could be worthwhile for solving the problem

Abstractions

- Some problems are more interesting than some others
- For instance, the Golomb ruler problem is more interesting than the allinterval series

Abstractions

- Allinterval Series:
  Find a permutation (x1, ..., xn) of {0,1,...,n-1} such that the list (abs(x2-x1), abs(x3-x2), ... , abs(xn - xn-1)) is a permutation of {1,2,...,n-1}.
- Golomb Ruler:
  a set of n integers 0=x1 < x2 < … < xn s.t. the n(n-1)/2 differences (xk - xi) are distinct and xn is minimized
- In the allinterval series there is no mix between the alldiff constraint and the arithmetic constraints (2 separate alldiff + absolute difference constraints), whereas such a mix exists in the Golomb ruler

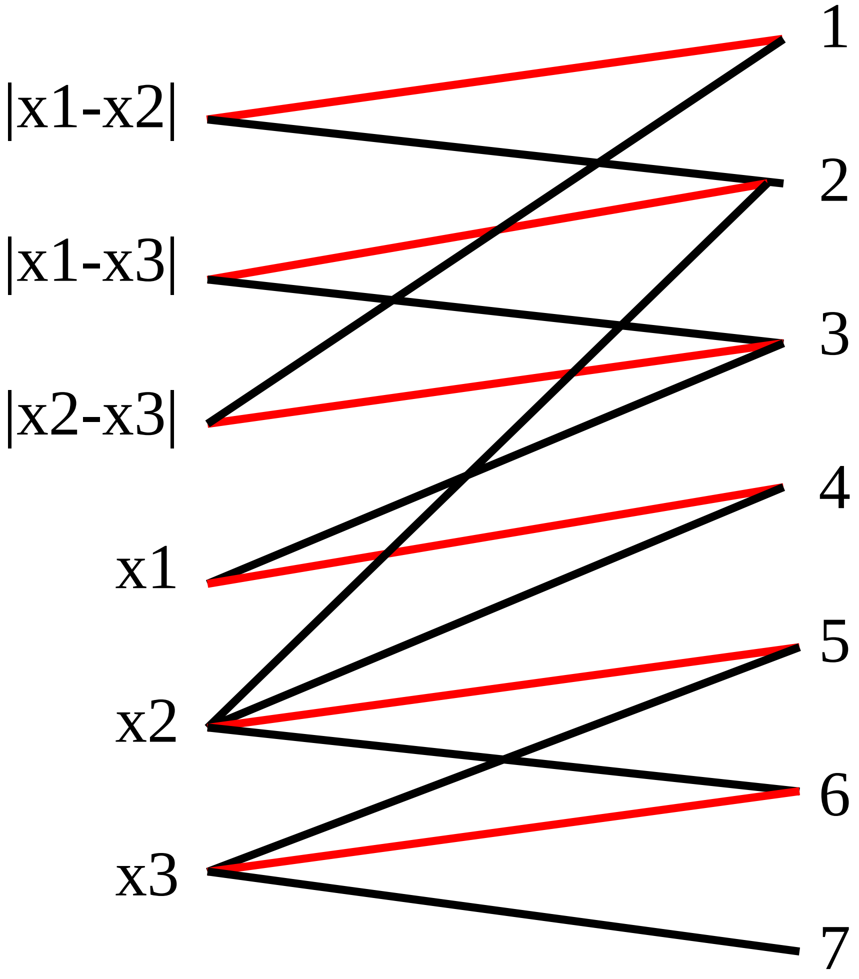
AllInterval series

- See Puget & Regin's note in the CSPLib
- 2 first solutions non symmetrical:
  - N=2000, #fails=0, time=32s (Pentium III, 800Mhz)
  - N <100 #fails=0, time < 0.02s
- All solutions:
  - N=14, #fails=670K, time=600s, #sol=9912
- This problem is not really difficult

Golomb Ruler

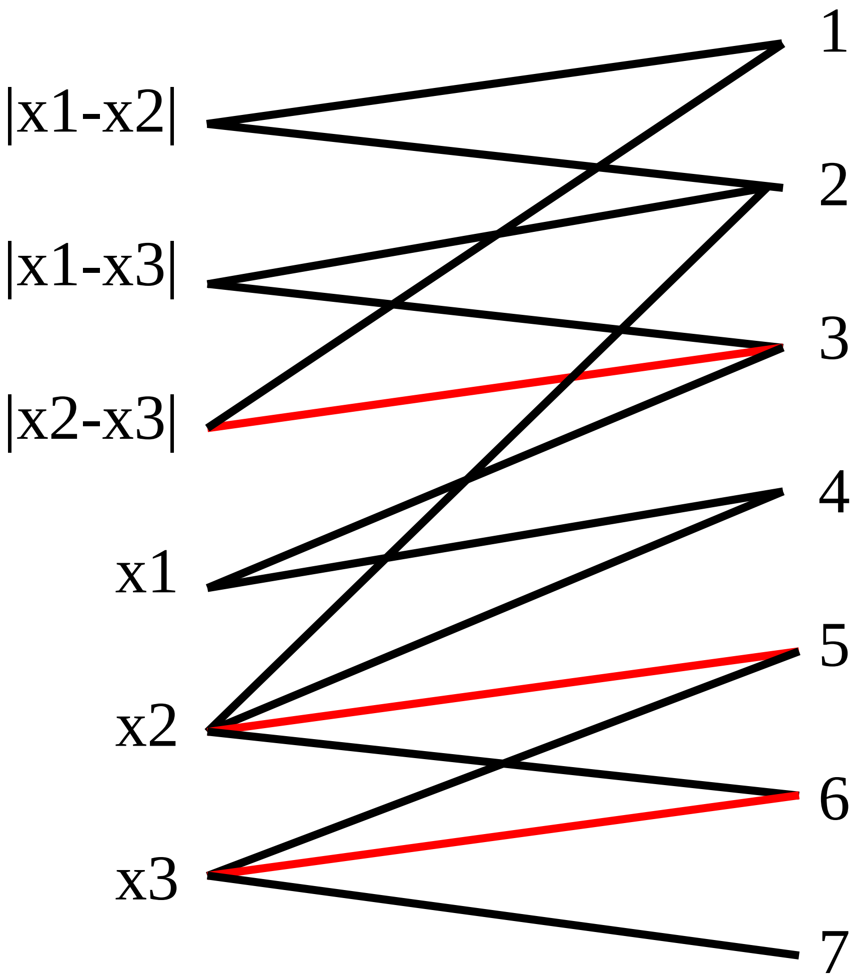
- x1,…,xn = variables; (xi-xj)= variables. Alldiff involving **all** the variables.
- with CP difficult for n > 13.

Alldiff



Not a good solution
Bad incorporation
of constraint
|xi – xj| in alldiff

Alldiff



Not a good solution
Bad incorporation
of constraint
|xi – xj| in alldiff

Golomb Ruler

- Conclusion about the Golomb Ruler: we are not able to integrate counting constraints and arithmetic constraints
- If we want to solve such a problem:
  - Either we are able to do that
  - Or we find a completely different model
- The Golomb Ruler Problem is not a subproblem of any problem, BUT it is a good representative of a type of combination we are not able to solve
- Improving the resolution of Golomb Ruler will help us to improve the resolution of a lot of problems

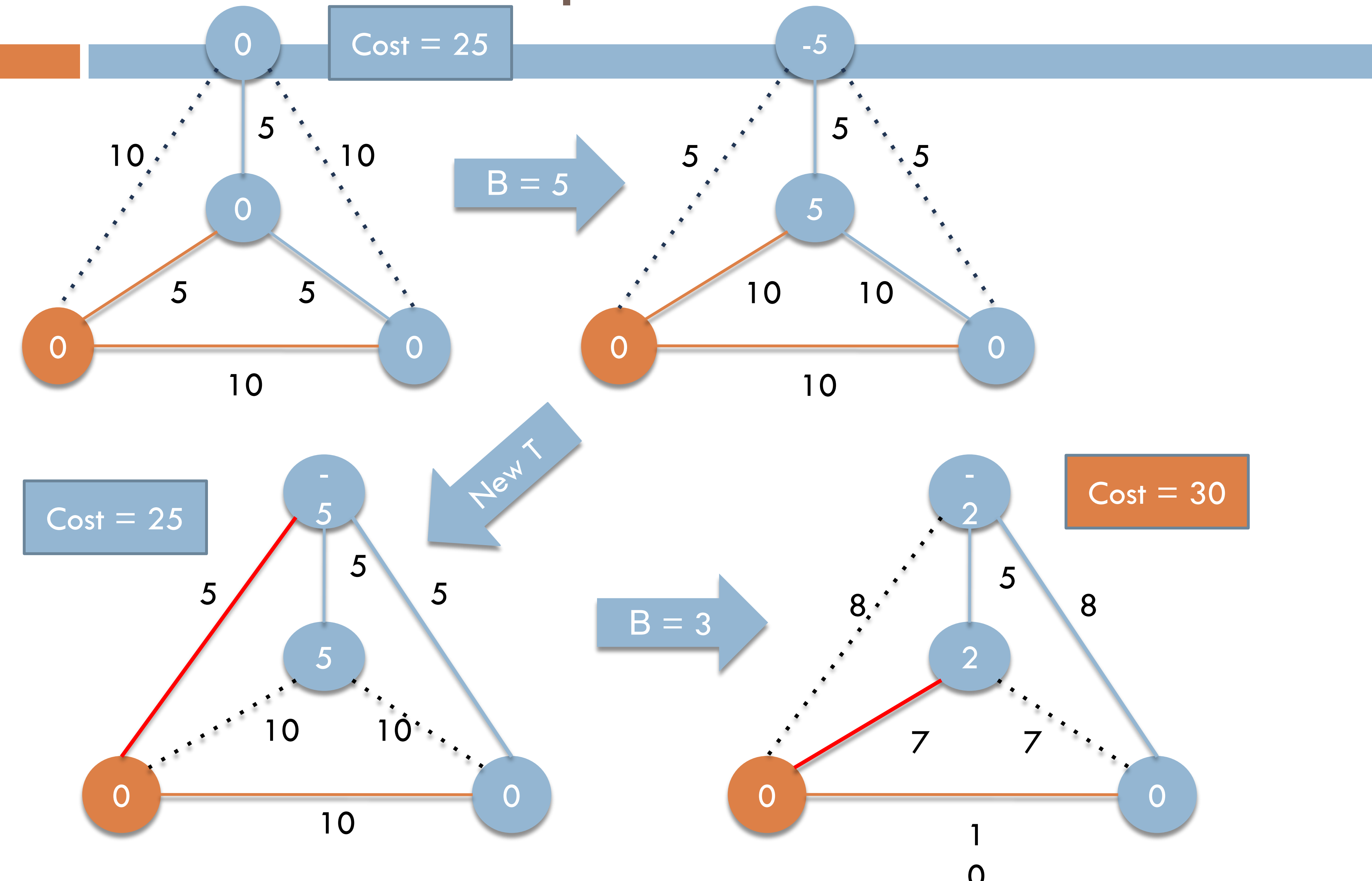
Abstraction

- Consider you have a mix of symbolic and arithmetic constraints
- If I solve the golomb ruler then I will be able to solve the allinterval series
- The opposite is not true
- Conclusion
  - The golomb ruler is a good abstraction
  - The allinterval series is not a good abstraction

Good abstraction

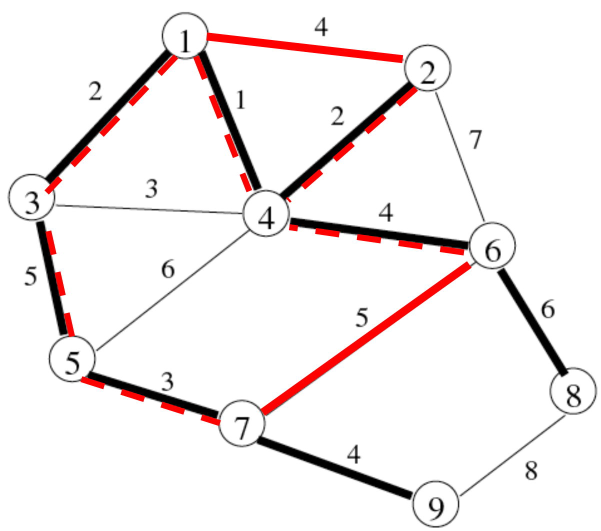
- An example of good abstraction is the 1-tree for the TSP (Traveling Salesman Problem)
  - P. Benchimol, J-C. Régin, L-M. Rousseau, M. Rueher and W-J. van Hoeve: "Improving the Held and Karp Bound with Constraint Programming", CP-AI-OR'10, Bologna, 2010
  - J-C. Régin, L-M. Rousseau, M. Rueher and W-J. van Hoeve: "The Weighted Spanning Tree Constraint Revisited", CP-AI-OR'10, Bologna, 2010

Held and Karp Bound for TSP



Replacement costs

- An edge e is inconsistent iff every spanning tree that contains e has weight > *K*
- Replacement edge
  - Replacement edge minimizes the increase of cost
  - Replacement edge = maximum edge on the *i-j* path in *T*



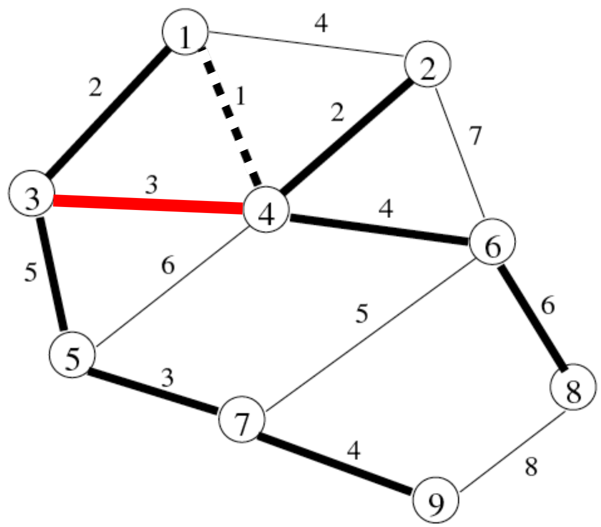
Replacement cost of

- (1,2) is 4 - 2 = 2
- (6,7) is 5 - 5 = 0

Replacement cost for tree edges

- The replacement cost of a *tree* edge e is
  w(*T'*) - w(*T*), where

  T is a minimum spanning tree of *G*, and *T'* is a minimum spanning tree of G \ e
- In other words, it represents the minimum marginal increase if we replace e by another edge
- An edge e is mandatory iff its replacement cost + w(*T*) > *K*

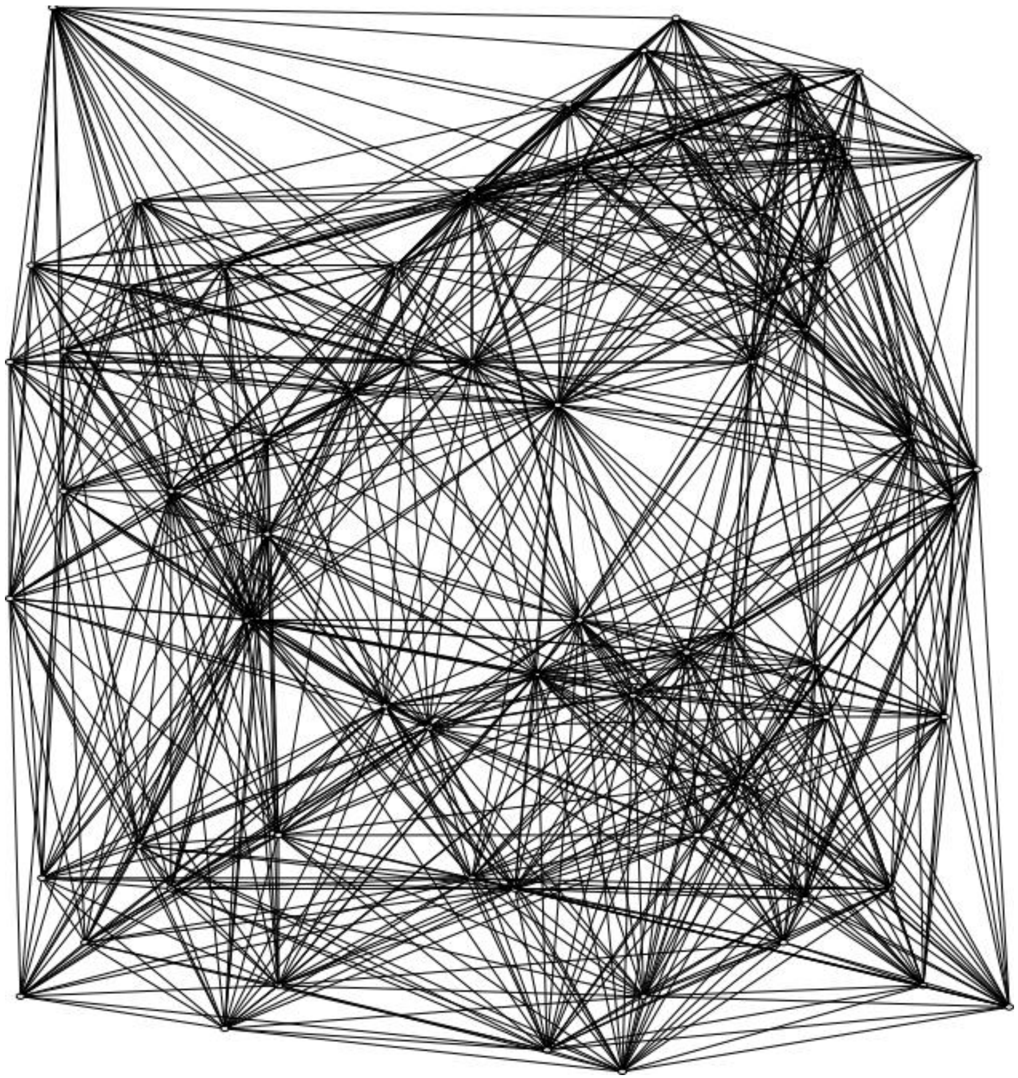


Replacement cost of (1,4)?

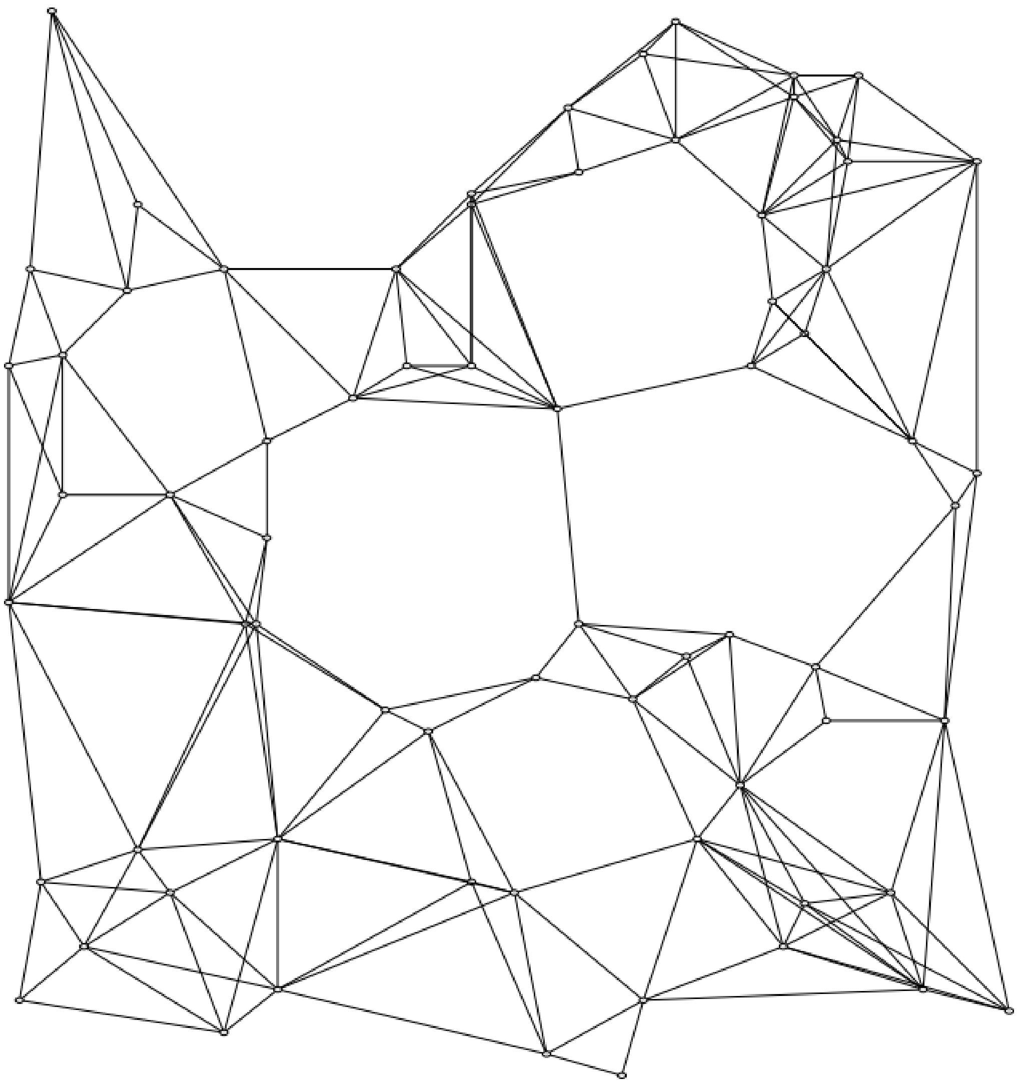
we need to find the cheapest

edge to reconnect: 3 - 1 = 2

St70 opt = 675 upper bound 700



St70 opt=685 upper bound=675



TSP: results

	HK no filtering			HK with filtering			Concorde		
size	solved	time	nodes/s	solved	time	nodes/s	solved	time	nodes/s
50	1.00	0.13	299.26	1.00	0.03	712.39	1.00	0.18	19.59
100	1.00	3.19	55.10	1.00	0.34	160.65	1.00	0.31	6.10
150	1.00	18.31	13.83	1.00	1.42	46.91	1.00	0.59	4.52
200	1.00	132.30	5.16	1.00	4.68	33.00	1.00	0.97	3.18
250	0.97	409.88	2.13	1.00	10.98	25.76	1.00	1.98	2.83
300	0.80	770.67	1.38	1.00	24.35	20.29	1.00	2.32	2.15
350	0.67	1,239.25	0.61	1.00	39.54	15.96	1.00	3.74	1.92
400	0.33	1,589.71	0.42	0.97	108.45	11.04	1.00	4.57	1.64
450	0.17	1,722.56	0.34	1.00	121.08	12.16	1.00	4.99	1.68
500	0.00	1,800.00	0.21	0.97	194.32	8.81	1.00	6.42	1.38
550	0.00	1,800.00	0.20	0.97	206.99	7.98	1.00	5.00	1.00

TSP: results

		HK no filtering			HK with filtering			IBM-ILOG CPO		
instance	UB	tour	search nodes	time	tour	search nodes	time	tour	search nodes	time
burma14	3323	3323	0	0.00	3323	0	0.00	3323	26,158	1.90
ulysses16	6859	6747	0	0.01	6747	0	0.01	6859	396,557	34.72
gr17	2085	2085	0	0.02	2085	0	0.01	2085	1,780,915	171.37
gr21	2707	2707	0	0.01	2707	0	0.01	2707	29,188	5.61
ulysses22	7013	6901	2,275	1.77	6901	0	0.03	-	12,595,543	1,800.00
gr24	1272	1272	19	0.12	1272	2	0.04	1272	3,804,284	621.70
fri26	937	937	41	0.18	937	2	0.03	937	13,627,564	1,800.00
bayg29	1610	1610	30	0.22	1610	4	0.05	-	5,883,592	1,800.00
bays29	2020	2020	35	0.22	2020	14	0.07	-	5,746,472	1,800.00
dantzig42	699	699	203	1.02	699	24	0.15	-	4,371,803	1,800.00
swiss42	1273	1273	58	0.74	1273	8	0.09	-	3,070,529	1,800.00
att48	10628	10628	101	1.72	10628	13	0.16	-	1,838,805	1,800.00
gr48	5046	5046	16,949	40.31	5046	11,832	7.20	-	1,921,955	1,800.00
hk48	11461	11461	45	1.50	11461	7	0.13	-	1,967,630	1,800.00
eil51	426	426	1,600	4.84	426	965	0.67	-	1,578,126	1,800.00
berlin52	7542	7542	0	0.02	7542	0	0.02	-	1,355,675	1,800.00
brazil58	25395	25395	725	5.40	25395	294	0.72	-	693,846	1,800.00
st70	675	675	10,800	48.68	675	5,145	5.18	-	972,717	1,800.00
eil76	538	538	300	7.07	538	106	0.52	-	839,789	1,800.00
rat99	1211	1211	1,872	35.40	1211	777	2.01	-	902,510	1,800.00
kroD100	21294	-	158,663	1,800.00	21294	95,733	169.13	-	435,816	1,800.00
rd100	7910	7910	927	35.23	7910	375	1.58	-	474,627	1,800.00
eil101	629	629	3,596	49.57	629	935	2.72	-	399,901	1,800.00
lin105	14379	14379	103	32.63	14379	5	1.09	-	579,675	1,800.00
pr107	44303	-	44,371	1,800.00	44303	62	11.87	-	1,962,612	1,800.00
gr120	6942	-	66,201	1,800.00	6942	126,966	288.64	-	214,480	1,800.00
br17	39	39	1,830,596	647.19	39	728,627	249.35	39	29,695,684	1,771.71
ftv33	1286	1286	37	2.12	1286	2	0.22	1286	4,164,410	1,629.24
ftv35	1473	1473	259	3.58	1473	174	0.62	-	4,957,650	1,800.00
ftv38	1530	1530	297	4.97	1530	223	0.89	-	4,570,728	1,800.00
ftv44	1613	1613	1,297	14.09	1613	855	2.64	-	3,766,601	1,800.00
ftv47	1776	1776	1,769	19.81	1776	1,059	3.92	-	2,623,052	1,800.00
ry48p	14422	14422	967	21.19	14422	629	3.47	-	1,763,828	1,800.00
ft53	6905	6905	52	2.98	6905	0	1.02	-	1,856,745	1,800.00
ftv55	1608	1608	6,237	68.47	1608	5,146	15.78	-	2,332,466	1,800.00
ftv64	1839	1839	14,215	168.14	1839	10,104	41.31	-	1,653,098	1,800.00
ftv70	1950	1950	67,436	1,022.77	1950	54,484	246.73	-	1,608,638	1,800.00
kro124p	36230	36230	18,539	1,178.73	36230	13,175	117.16	-	581,790	1,800.00

TSP

- Good abstraction
- Random-restart
- Good benchmarks

- Lack of decomposition?

Conclusion

- When you want to solve a problem or when you are not able to solve a problem. Think about the 4 common pitfalls
  - Undivided model
  - Rigid search
  - Biased benchmarking
  - Wrong abstraction
- Try to solve some real world problems
- Try to solve some well known problems (clique max, TSP, coloring, …)