How to prevent tall trees from growing to the sky

Subtitle: don't do NAAR

Jean-Charles REGIN
ILOG, Sophia Antipolis
regin@ilog.fr

Tall tree can't grow to the sky

In CP

- Tall tree grow to the sky!
- A lot of problems addressed by CP have an exponential complexity
- In fact, it depends on P vs NP

Outline

- P = NP: role of CP?
- P $\neq$NP: shifting the exponential
- Theoretical research
- Applied research
- Benchmarking in CP
- Conclusion

Acknowledgements

- Philippe Baptiste
- Pascal Van Hentenryck
- Carla Gomes
- Diego Olivier Fernandez Pons
- Phokion G. Kolaitis and Thomas Raffill

P vs NP

- P=NP or P≠NP
- Consider the two possibilities in regards to the impact to CP

P = NP

- If a general algorithm for solving NP-Complete Problems exists then what is advantage of CP ?
- There are several reasons to believe that our community is going to disappear:
  - All the known polynomial algorithms have not a huge complexity (the max is currently close to n^10)
  - CP is not able to solve polynomial instances in polynomial time. There is no guarantee about that.

P = NP

- Fortunately, there are also some hopes:
  - A non constructive proof exists for proving a problem is in P (P. Jegou told me that, ACM 85 paper)
  - A very large constant is possible
  - The degree of the polynom can be very large (n^1000)
- This would mean that the general algorithm would not be usable in practice

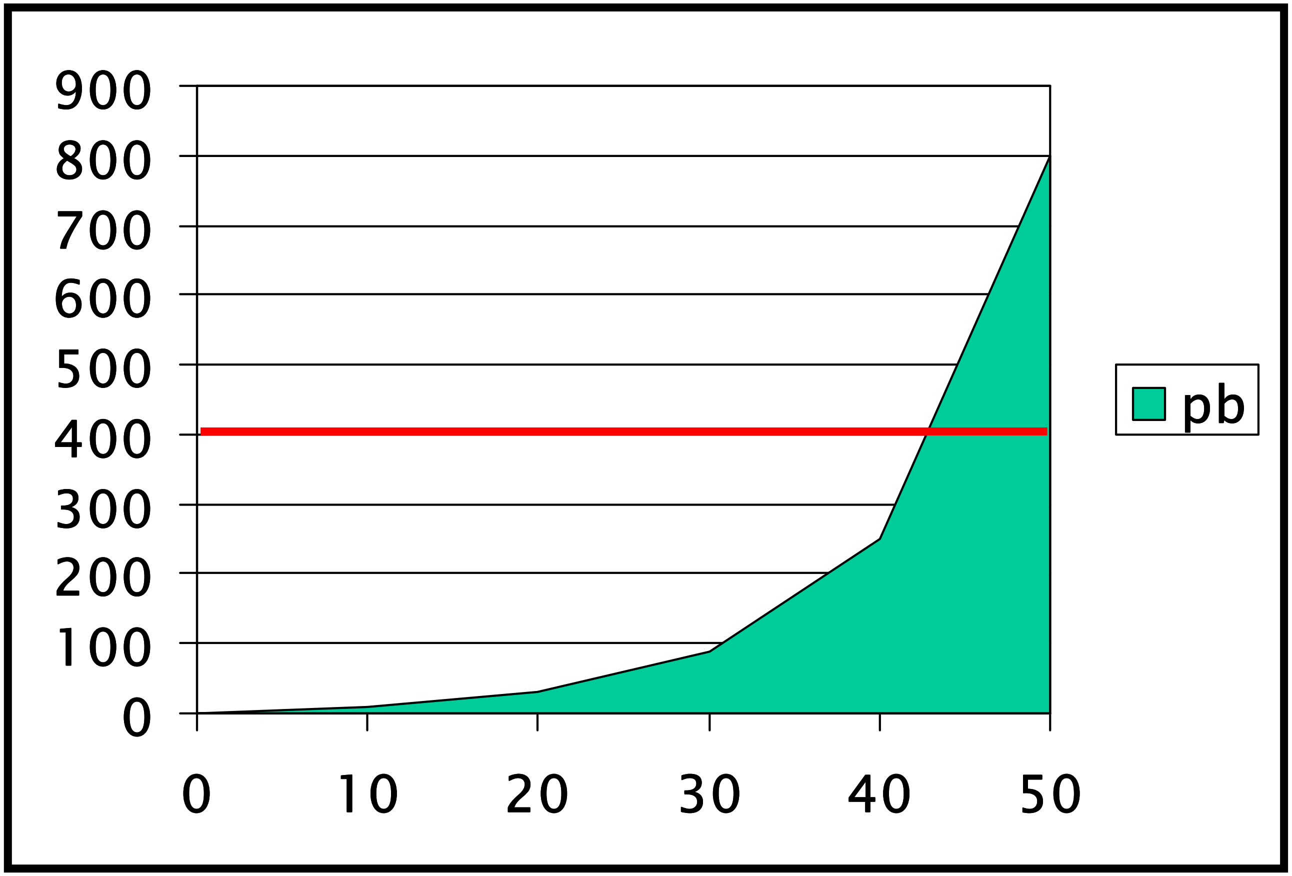
Outline

- P = NP: role of CP?
- **P $\neq$NP: shifting the exponential**
- Theoretical research
- Applied research
- Benchmarking in CP
- Conclusion

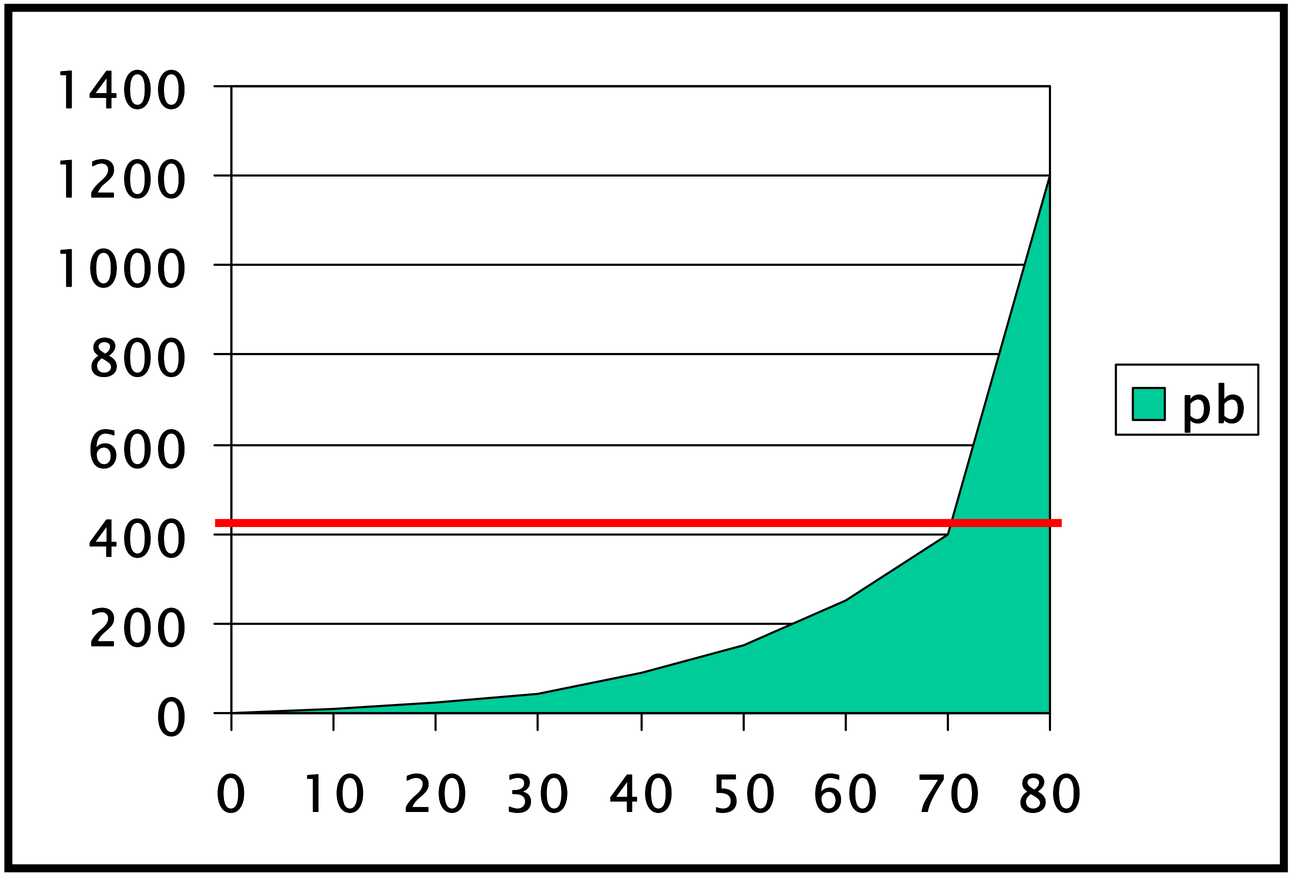
P ≠ NP

- Ok, we cannot avoid an exponential behavior
- For some instances, an 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



Shifting the exponential



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(Mij)=[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

MIP solver limit

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

Second model: results

# 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

MIPLIB

MIP limit

First model limit

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

MIP solver limit

$P \neq NP$

- We have only two reasonable possibilities:
  - We are doing pure theoretical research
  - We are trying to “solve” existing problems
- Working on the middle topics has no real meaning.

Outline

- P = NP: role of CP?
- P $\neq$NP: shifting the exponential
- **Theoretical research**
- Applied research
- Benchmarking in CP
- Conclusion

Theoretical Research

- IMHO, the work of C. Gomes and B. Selman is an excellent example of good theoretical study.
- I am not a specialist of this kind of research, so I can give my opinion about it ☺

Phase Transitions

- A phase transition is an abrupt change in the behavior of a property of a “system”.
- Extensive study of phase transitions in physics (statistical mechanics).
- Extensive study of phase transitions in NP-complete problems during the past decade.

Motivation and Goals

- Understand the “structure” of NP-complete problems.
- Relate phase transitions to the average-case performance of particular algorithms for NP-complete problems.

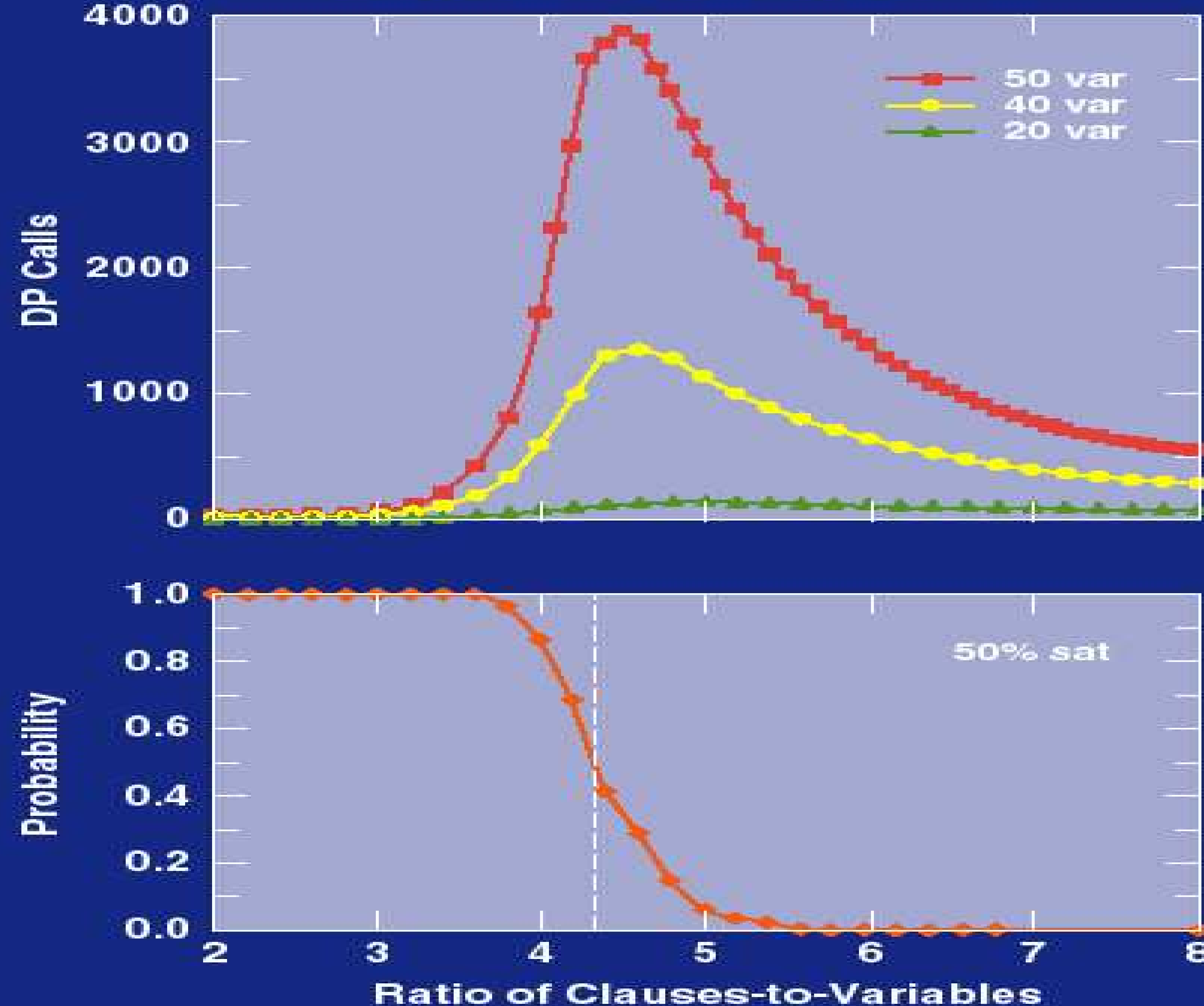
NP-Complete Problems

- Introduce a “constrainedness” parameter to partition the space of instances.
- Generate random instances at fixed parameter values.
- For some problems, probability of a “yes” instance abruptly changes from 1 to 0 at some critical value.
- For some problems and some solvers, average difficulty peaks sharply at the same critical value.

Main Example: 3-SAT

- Parameter: Ratio of number of clauses to number of variables.
- Intuition: Low ratios are underconstrained, high ratios are overconstrained.
- Critical Value: Experimental results suggest that it is about 4.3 clauses to variables.
- Average Performance: DPLL procedure peaks around 4.3

The 4.3 Point



Mitchell, Selman, and Levesque 1991

Phase Transition: advantages?

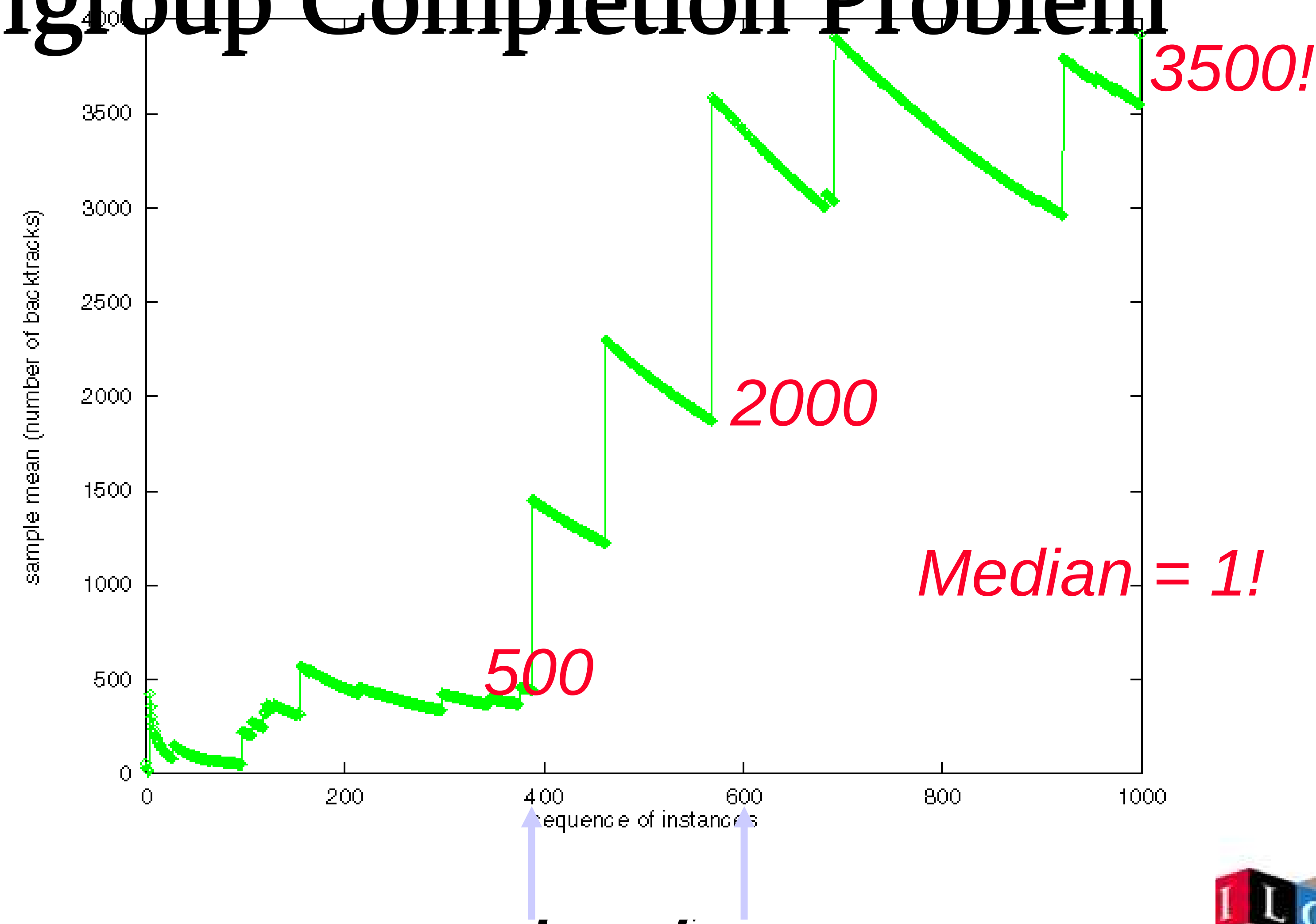
- Honestly, I don't know
- It is interesting and scientifically respectable

Heavy Tails

- These slides come from Carla Gomes's talk
- This is a perfect example of a successful theoretical study, because
  - It is interesting
  - It leads to huge improvement in the resolution of practical applications
  - It is integrated into ILOG CPOptimizer

Erratic Behavior of Search Cost
Quasigroup Completion Problem

sample mean

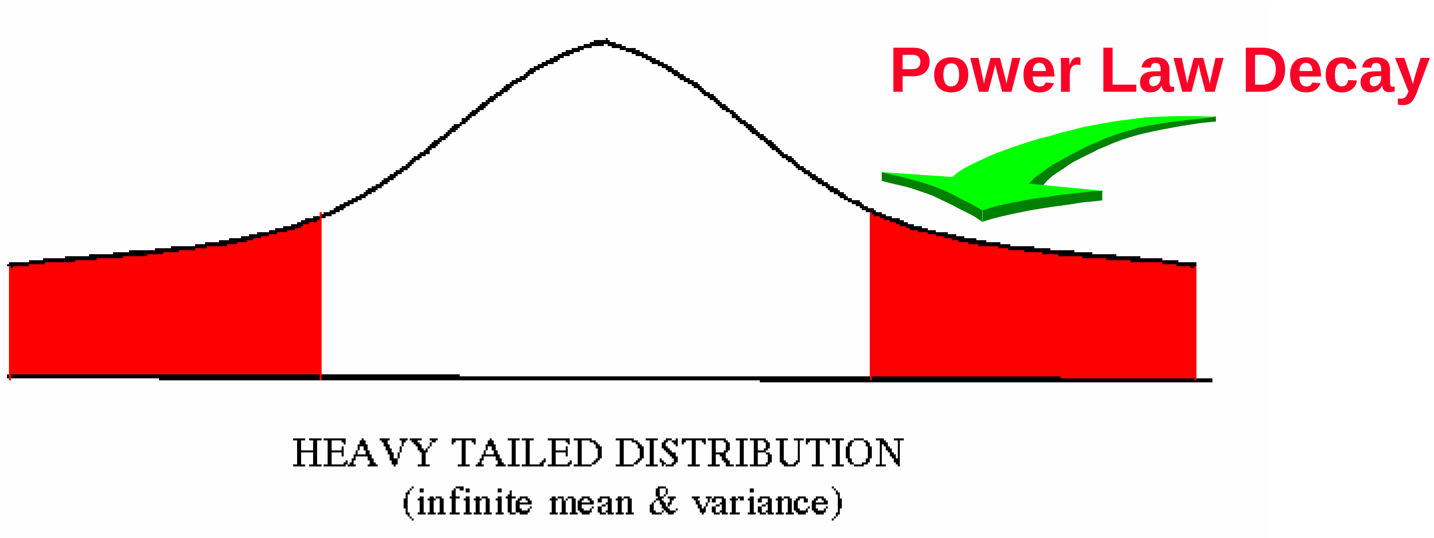


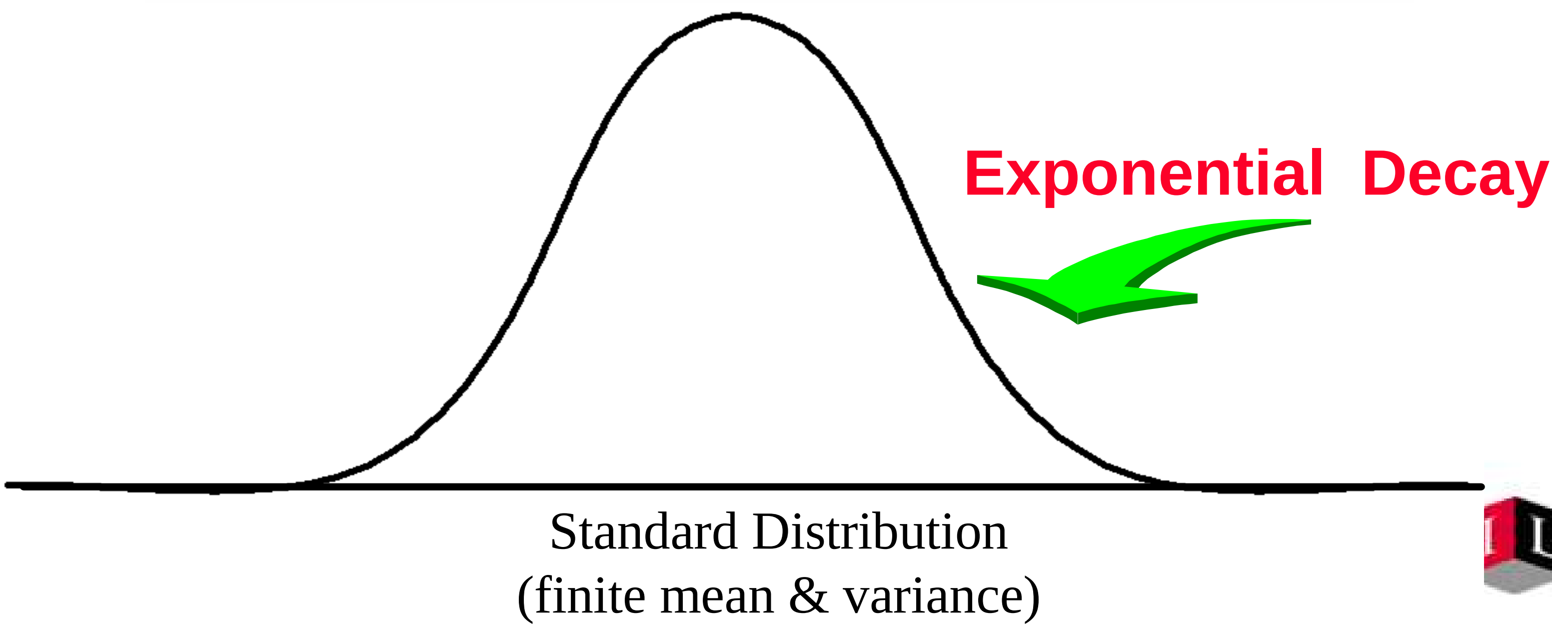
number of runs

Heavy-Tailed Distributions

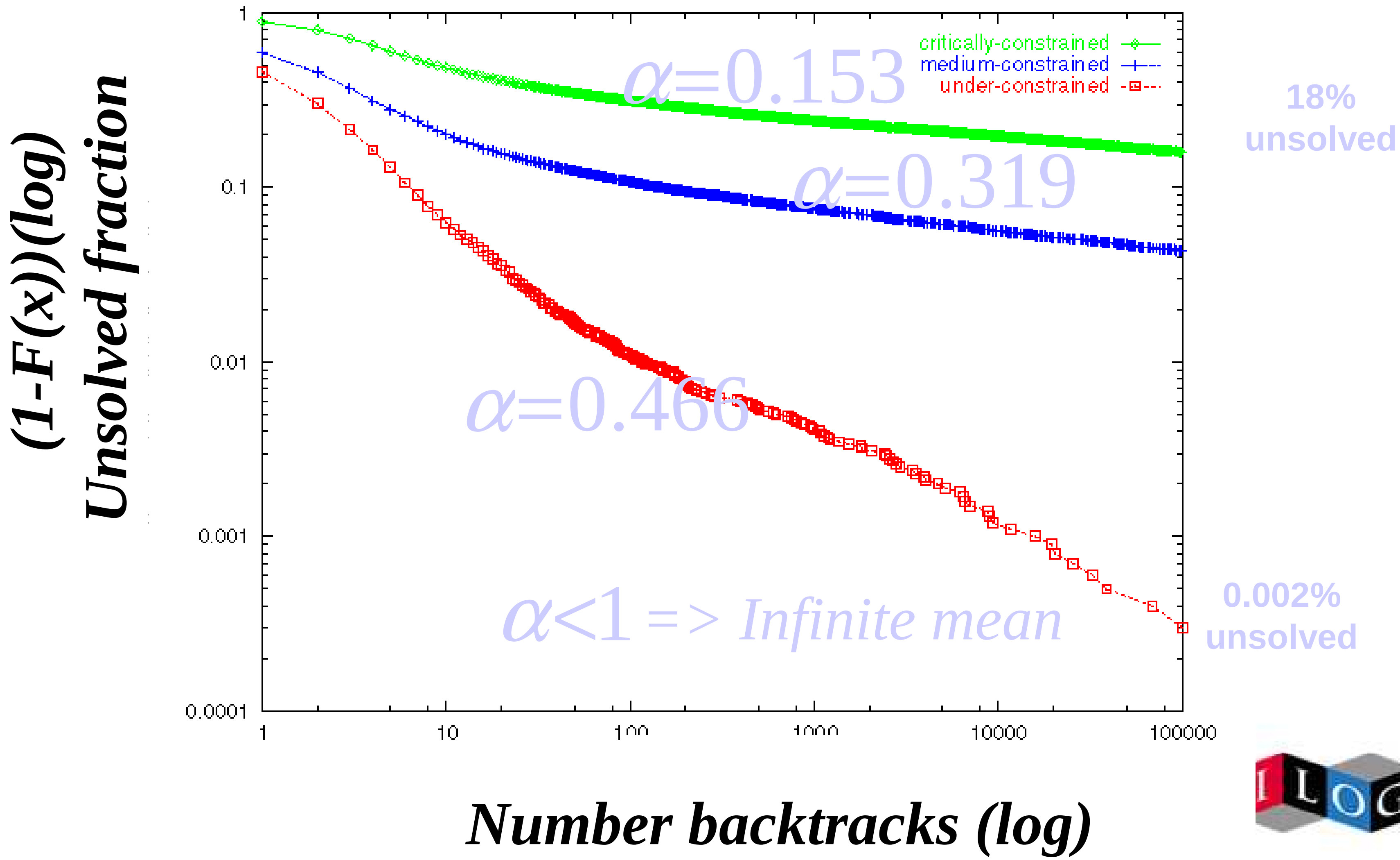
- … infinite variance … infinite mean

- Introduced by Pareto in the 1920's
- --- "probabilistic curiosity."
- Examples: stock-market, earth-quakes, weather,...


Power Law Decay
HEAVY TAILED DISTRIBUTION
(infinite mean & variance)


Exponential Decay
Standard Distribution
(finite mean & variance)

Heavy-Tailed Behavior in QCP Domain



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.

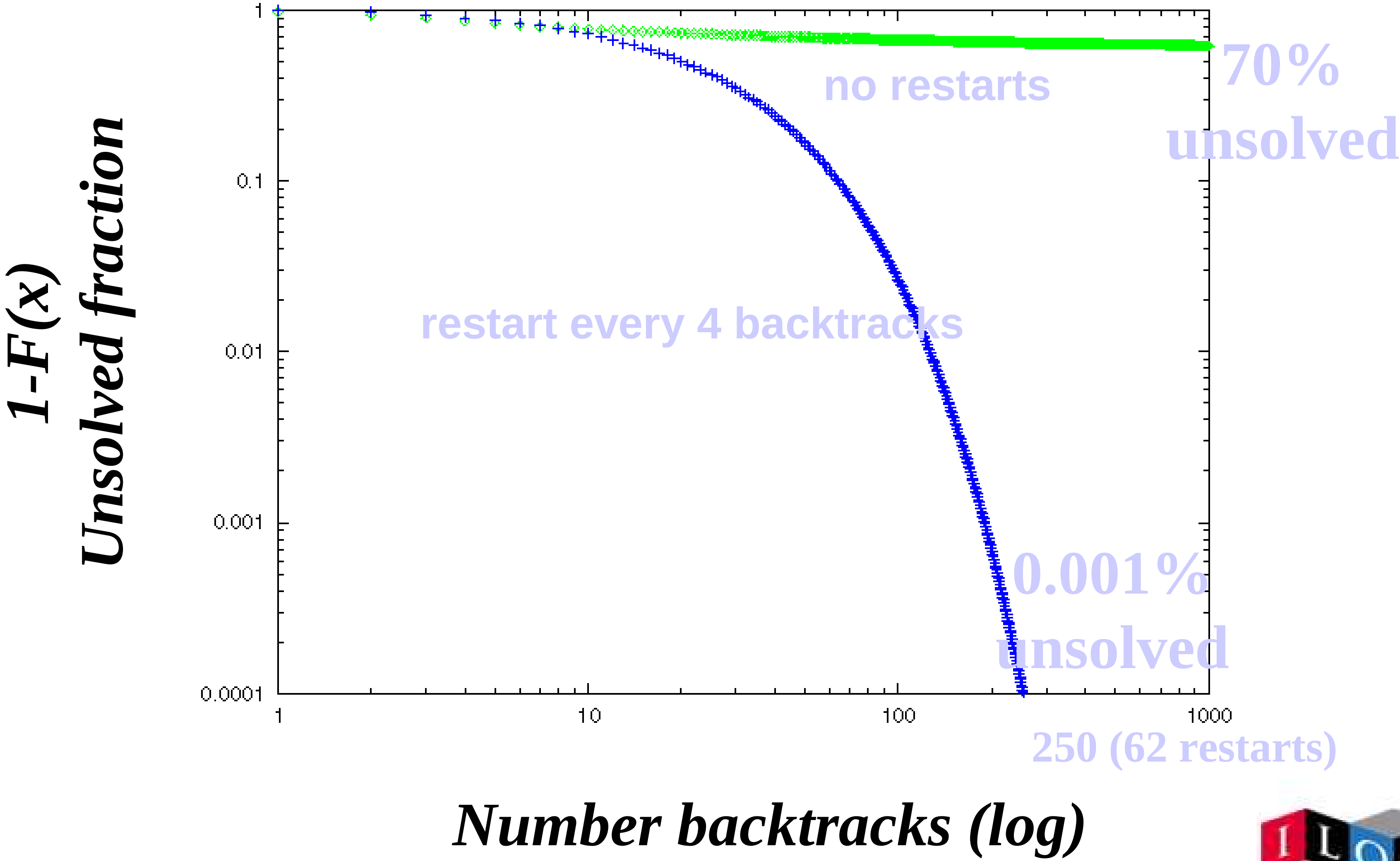
Exploiting Heavy-Tailed behavior

- 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)

- We implement this idea in ILOG CPOptimizer and it works!
- Main advantage: it is much more robust

Restarts


1
0.1
0.01
0.001
0.0001
1
10
100
1000
no restarts
70%
unsolved
restart every 4 backtracks
0.001%
unsolved
250 (62 restarts)
1-F(x)
Unsolved fraction
Number backtracks (log)

Outline

- P = NP: role of CP?
- P $\neq$NP: shifting the exponential
- Theoretical research
- **Applied research**
- Benchmarking in CP
- Conclusion

Applied Research

- We don’t need ad-hoc solutions
- We need more general concept that could be applied on applications
- Problem: how to find such a generalization?

Random Problems

- Pure random problem is useless in CP except for a pure theoretical study
- There is no random problem in the world
- CP exploits the structure of the problems
- We can accept to generate some random data of structured instances but this is quite different

Identification of hard problems

- Often, we focus our attention on some small problems
- That’s reasonable but we have to be careful

Scheduling example

- "Is scheduling theory any useful to solve scheduling problems ?" Subtitle of P. Baptiste's invited talk at CPAIOR-07
- "Scheduling Theory = drastic simplification of real life problems"
- So much drastic that some considered problems have absolutely no real meaning:
- αβγ problems where αβγ stands for
  - α = "Machine Environment" (single,parallel machines…)
  - β = "Jobs characteristics" (preemption, same processing time …)
  - γ = "Objective function" (makespan, minimum tardiness…)

Scheduling example

- αβγ problems with α, β, and γ values corresponding to no problem of the real world.
- People invent problems on which they work, like if it could have any interest

CP example

- We can see in papers a lot of problems which are not realistic
- At CP2006 a paper about configuration presented some experiments with only 20 values per domain, and all variables known in advance.
- This is really strange knowing that configuration problems are problems in which we cannot use a reduction method.

Good problems

- Real world applications are difficult to solve
- Any resolution of a real world application even if it looks simple deserves more attention
- Some subparts of real world problems, but realistic subparts and not invented subparts
- Some problems represents very well some issues of CP

Good problems

- Some problems are more interesting than some others
- For instance, the Golomb ruler problem is more interesting than the allinterval series
- Allinterval Series: Find a permutation $(x_1, ..., x_n)$ of $\{0,1,...,n-1\}$ such that the list $(abs(x_2-x_1), abs(x_3-x_2), ..., abs(x_n - x_{n-1}))$ is a permutation of $\{1,2,...,n-1\}$.
- Golomb Ruler: a set of n integers $0=x_1 < x_2 < \ldots < x_n$ s.t. the $n(n-1)/2$ differences $(x_k - x_i)$ are distinct and x_n is minimized
- In the allinterval series there is no mix between the alldiff constraint and the arithmetic constraints, whereas such a mix exists in the Golomb ruler

Good problems

- Allinterval Series:
  Find a permutation $(x_1, ..., x_n)$ of $\{0,1,...,n-1\}$ such that the list $(abs(x_2-x_1), abs(x_3-x_2), ... , abs(x_n - x_{n-1}))$ is a permutation of $\{1,2,...,n-1\}$.
- Golomb Ruler:
  a set of n integers $0=x_1 < x_2 < \ldots < x_n$ s.t. the $n(n-1)/2$ differences $(x_k - x_i)$ are distinct and x_n 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

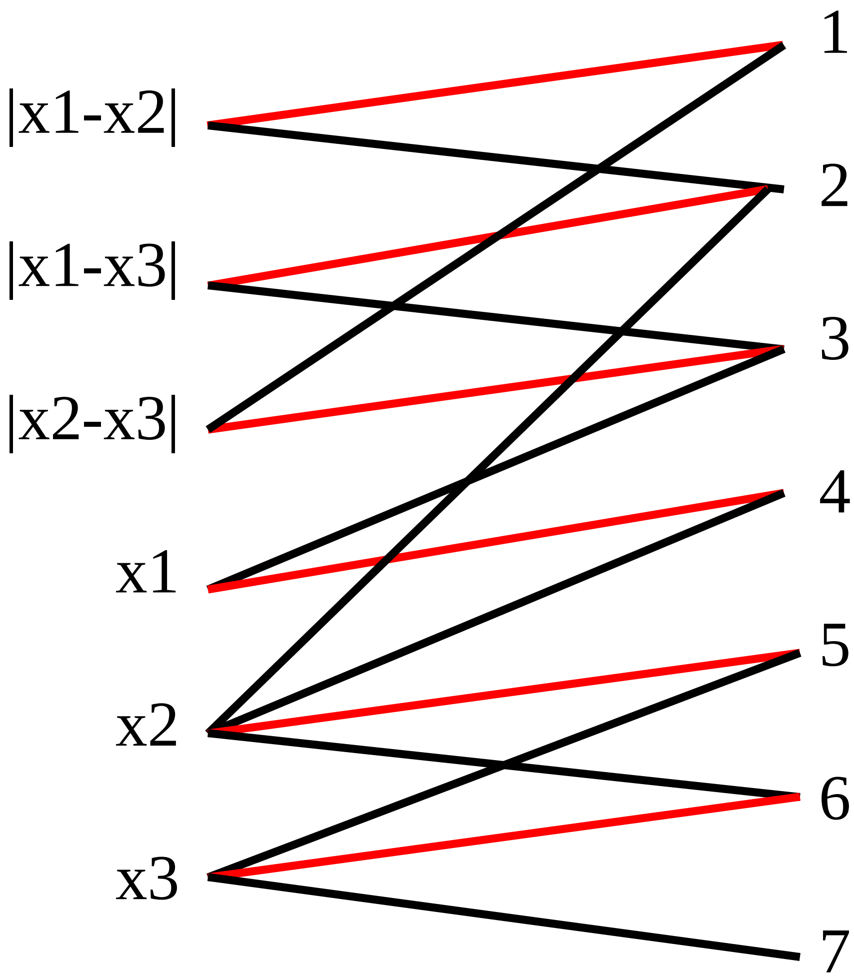
AllInterval series

- See Puget & Regin's note in the CSPLib
- 2 first solutions non symetrical:
  - 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

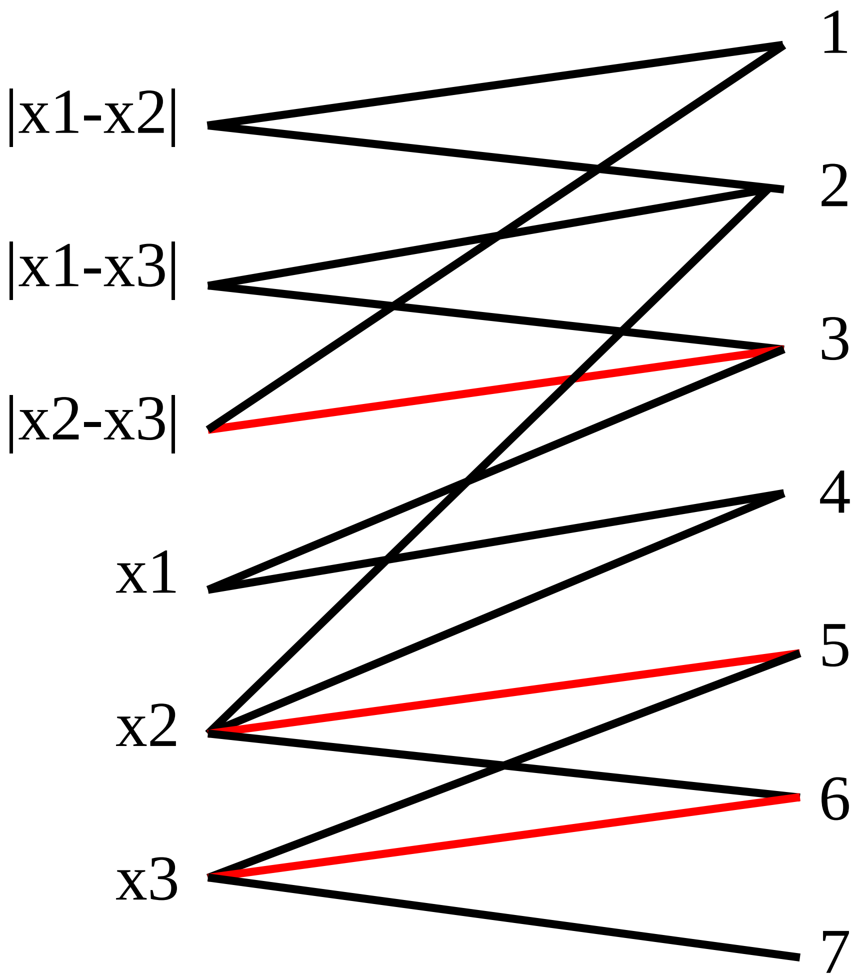
- x1,…,xn = variables; (xi-xj)= variables. Alldiff involving **all** the variables.
- with CP difficult for n > 13.

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

Outline

- P = NP: role of CP?
- P $\neq$NP: shifting the exponential
- Theoretical research
- Applied research
- **Benchmarking in CP**
- Conclusion

Benchmarking

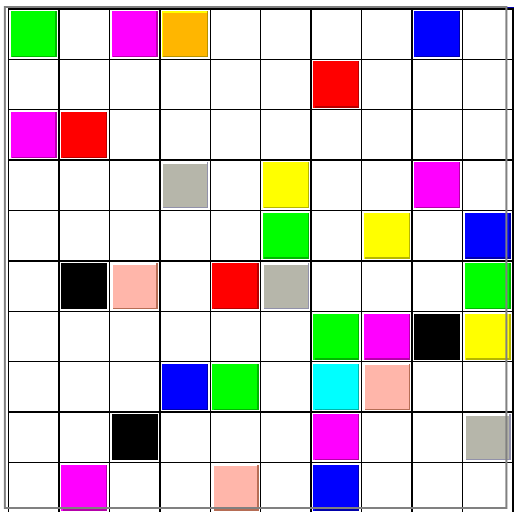
- This is serious and difficult
- The name of the problem is not sufficient: e.g. quasigroup completion problem, latin square. For instance, it is very hard to find hard instances of the latin square problem for small values (<100 or < 200). But there are some difficult instances for n=35
- When the problem is a common subproblem it is better to consider instances that are not empty at the beginning, because we could have a better picture of the integration of the work into another application
- 2 examples: latin square and network design

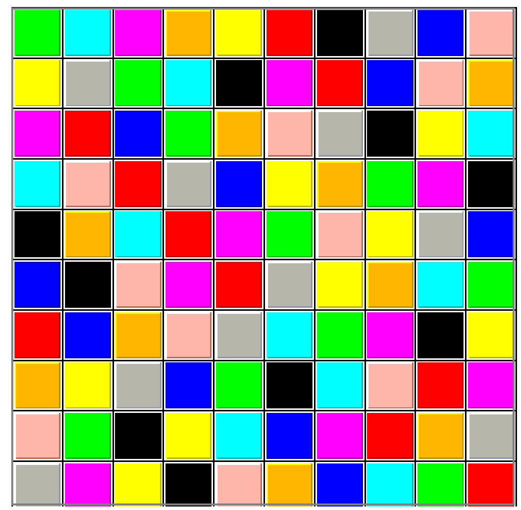
Latin Square Completion

Given a partial assignment of symbols to a Latin Square, can we complete it without repeating symbols in a row/column?

Example:

(Gomes & Selman 97)





32% preassignment

Underlying structure is found in many real world applications: Scheduling, Timetabling, Routing, Design of Experiments, Cryptography.

Design of Statistical Experiments

- We have 5 treatments for growing beans. We want to know what treatments are effective in increasing yield, and by how much.
- The object is to eliminate bias and distribute the treatments somewhat evenly over the test plot
- Latin Square Analysis of Variance

Design of Treatment Experiment (5 Treatments: A,B,C,D,E)

A	D	E	B	C
C	B	A	E	D
D	C	B	A	E
E	A	C	D	B
B	E	D	C	A

(*) Already in use in this sub-plot

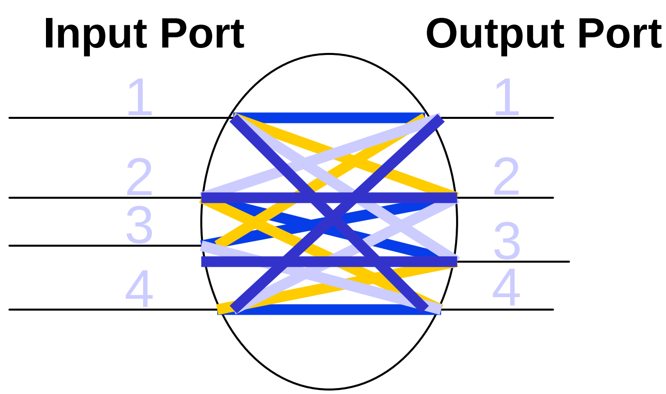
Round Robin Schedules

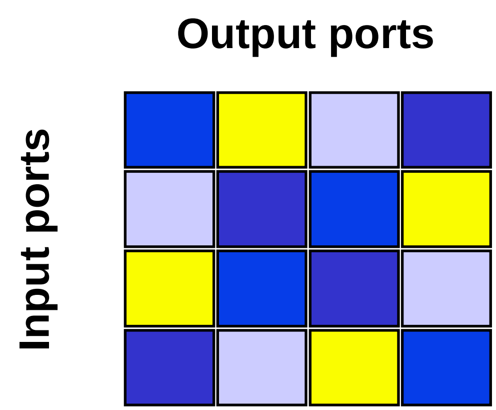
1	+ 2	- 3	+ 4	- 5	+ 6	- 7	+ 8
2	- 1	+ 4	- 6	+ 8	- 3	+ 5	- 7
3	- 8	+ 1	+ 5	- 7	+ 2	- 4	+ 6
4	+ 7	- 2	- 1	+ 6	- 8	+ 3	- 5
5	- 6	+ 8	- 3	+ 1	+ 7	- 2	+ 4
6	+ 5	- 7	+ 2	- 4	- 1	+ 8	- 3
7	- 4	+ 6	- 8	+ 3	- 5	+ 1	+ 2
8	+ 3	- 5	+ 7	- 2	+ 4	- 6	- 1

QCP Example Use: Routers in Fiber Optic Networks

•each channel cannot be repeated in the same input port (row constraints);

• each channel cannot be repeated in the same output port (column constraints);





CONFLICT FREE
LATIN ROUTER

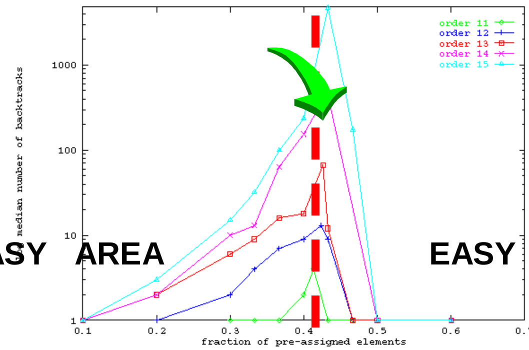
(Barry and Humblet 93, Cheung et al. 90, Green 92, Kumar et al. 99)

Complexity

Completing LS is
NP-Complete

Better characterization beyond worst case?

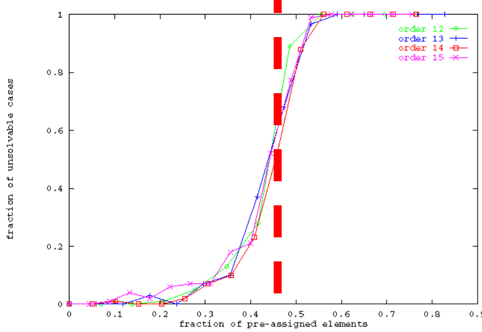
Critically
constrained area



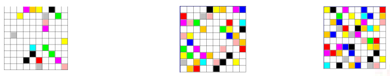
EASY AREA

EASY AREA

Complexity of Latin Square Completion



Phase Transition



20% 42% 50%

Gomes and Selman 97

Latin Square Completion

- This is a problem
- This is also a subproblem of a lot of problems
- From this benchmark some results have been obtained: AlldiffMatrix and CardinalityMatrix constraints

AlldiffMatrix constraint

		a	b		
		c	a		
		d	c		
		e	d		

Alldiff on rows and Alldiff on columns

AlldiffMatrix constraint

		a	b		
		c	a		
		d	c		
		e	d		
		b,f	e,f		
		b,f	e,f		

Alldiff on rows and Alldiff on columns

AlldiffMatrix constraint

		a	b		
		c	a		
		d	c		
		e	d		
a..f	a..f	b,f	e,f	a..f	a..f
		b,f	e,f		

Alldiff on row cannot deduce anything

Alldiff on rows and Alldiff on columns

Deductions: no

AlldiffMatrix constraint

		a	b		
		c	a		
		d	c		
		e	d		
		b,f	e,f		
		b,f	e,f		

Alldiff on rows and Alldiff on columns

Deductions: **Possible. There are only two solutions for the columns**

AlldiffMatrix constraint

		a	b		
		c	a		
		d	c		
		e	d		
		b,f	e,**f**		
		b,**f**	**e**,f		

Alldiff on rows and Alldiff on columns

Deductions: **Possible. There are only two solutions for the columns**

AlldiffMatrix constraint

			3	4		
			a	b		
			c	a		
			d	c		
			e	d		
5			b,**f**	**e**,f		
6			**b**,f	e,**f**		

For rows 5 and 6, value f
Can belong only to column 3 and 4

We can remove f from the other columns

Alldiff on rows and Alldiff on columns

Deductions: **Possible. There are only two solutions for the columns**

f e and **b f**

b f and **f e**

AlldiffMatrix constraint

			3	4		
			a	b		
			c	a		
			d	c		
			e	d		
5	-f	-f	b,**f**	**e**,f	-f	-f
6	-f	-f	**b**,f	e,**f**	-f	-f

For rows 5 and 6, value f
Can belong only to column 3 and 4

We can remove f from the other columns

Alldiff on rows and Alldiff on columns

Deductions: **Possible. There are only two solutions for the columns**

f e and **b f**

b f and **f e**

Cardinality Matrix Constraint

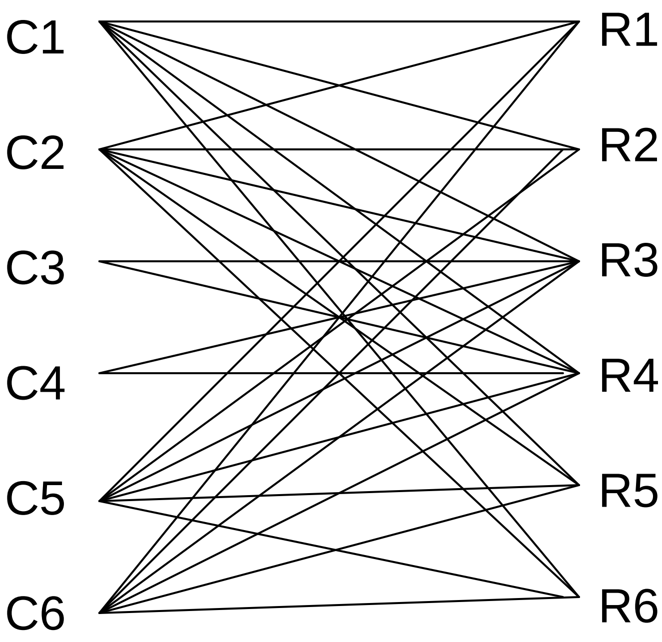
- Specific constraints which improves:
  - the communication between cardinality variables
  - the combination of rows and columns
- We also proposes a simple filtering algorithm for the cardinality variables

Cardinality Matrix Constraint

- This is a global constraint which is modeled by the conjunction of other global constraints. There is no specific filtering algorithm but a combination of filtering algorithm
- For the alldiffMatrix constraint the idea is quite simple and this idea is generalized for the cardinality matrix constraint

ILOG™

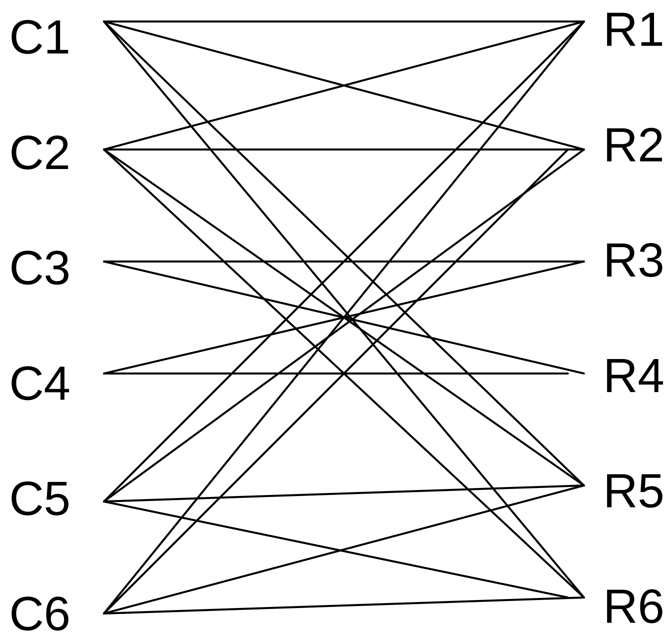
Alldiff on symbols



			3	4		
			a	b		
			c	a		
			d	c		
			e	d		
5						
6						

For every symbol an alldiff constraint is defined
If symbol f can be in cell Ci,Rj then there is an arc between Ci and Ri

Alldiff on symbols



			3	4		
			a	b		
			c	a		
			d	c		
			e	d		
5						
6						

For every symbol an alldiff constraint is defined

If symbol f can be in cell Ci,Rj then there is an arc between Ci and Ri

Results

	2alldiff-AC dom-lessO		3alldiff-AC dom-lessO		2alldiff-GAC dom-lessO		alldiff-matrix dom-lessO	
	time	#fails	time	#fails	time	#fails	time	#fails
qwh.order30.holes316		> 50,000		> 50,000	0.33	10	0.33	3
qwh.order30.holes320		> 50,000		> 50,000	1.16	1334	0.34	22
qwh.order50.holes2000		> 50,000	1.45	230	4.6	0	5.8	0
qwh.order60.holes1440		> 50,000		> 50,000		> 50,000		> 50,000
qwh.order60.holes1620		> 50,000		> 50,000		> 50,000	66.9	24,604
qwh.order60.holes1692		> 50,000		> 50,000	15.96	7,084	7.57	7,917
qwh.order60.holes1728		> 50,000		> 50,000		> 50,000	3.16	14
qwh.order60.holes1764		> 50,000		> 50,000	3.4	277	3.68	150
qwh.order60.holes1800		> 50,000		> 50,000	3.9	554	3.4	3
qwh.order70.holes2450		> 50,000		> 50,000	5.77	24	6.5	1
qwh.order70.holes2940		> 50,000		> 50,000	9.7	398	10.8	74
qwh.order70.holes3430		> 50,000		> 50,000	14.4	0	17	0

	2alldiff-GAC dom-lessO		2alldiff-GAC dom-maxB-lessO		alldiff-matrix dom-lessO		alldiff-matrix dom-maxB-lessO	
	time	#fails	time	#fails	time	#fails	time	#fails
qwh.order30.holes316	0.33	10	0.62	476	0.33	3	0.37	44
qwh.order30.holes320	1.16	1334	0.33	21	0.34	22	0.35	32
qwh.order50.holes2000	4.6	0	4.57	1	5.8	0	5.7	1
qwh.order60.holes1440		> 50,000		> 50,000		> 50,000	2.32	18
qwh.order60.holes1620		> 50,000		> 50,000	66.9	24,604	6.54	1,439
qwh.order60.holes1692	15.96	7,084	2.75	54	7.57	7,917	3.15	47
qwh.order60.holes1728		> 50,000	2.7	4	3.16	14	3.16	9
qwh.order60.holes1764	3.4	277	2.82	1	3.68	150	3.28	12
qwh.order60.holes1800	3.9	554	15.28	1,369	3.4	3	4.0	261
qwh.order70.holes2450	5.77	24	5.7	1	6.5	1	6.6	35
qwh.order70.holes2940	9.7	398	9.5	145	10.8	74	11.1	130
qwh.order70.holes3430	14.4	0	14.2	0	17	0	17.2	2

dom=dom min
lessO= min occurence
maxB= max var instantiate

Benchmarking

- I worked with C. LePape on the ROCOCO project
- C. LePape is very good to define benchmarks
- T. Benoist remind in his invited talk at CPAIOR-07 and JFPC-07 that some applications of Claude are still worldwide used to manage some part of the construction of buildings by Bouygues
- This is due to intensive benchmarking with a set of realistic benchmarks.

A Case Study in Network Design

- Very good example and benchmark:
  - To illustrate the advantages and drawbacks of different optimization techniques
  - To illustrate the improvements that can be thought of when things do not work well
  - To test new ideas

The ROCOCO Project (1)

- France Telecom R&D ISE
  - Problem and benchmark definition
  - Algorithm validation
- Research laboratories: INRIA Numopt, LRI Orsay, PRiSM Versailles, Evry, …
  - Lower bounds: Lagrangean relaxation, column generation, cuts
  - Optimization techniques: genetic algorithms
- ILOG
  - Optimization techniques: constraint programming, mixed integer programming, column generation

The Problem (1)

- Routing of Communications
  - Mono-routing: each demand from a point p to a point q must follow a unique path
- Dimensioning of Links
  - The capacity of each link must exceed the sums of the demands going through the link
- Additional Constraints
  - Depend on the customer for whom the network is designed

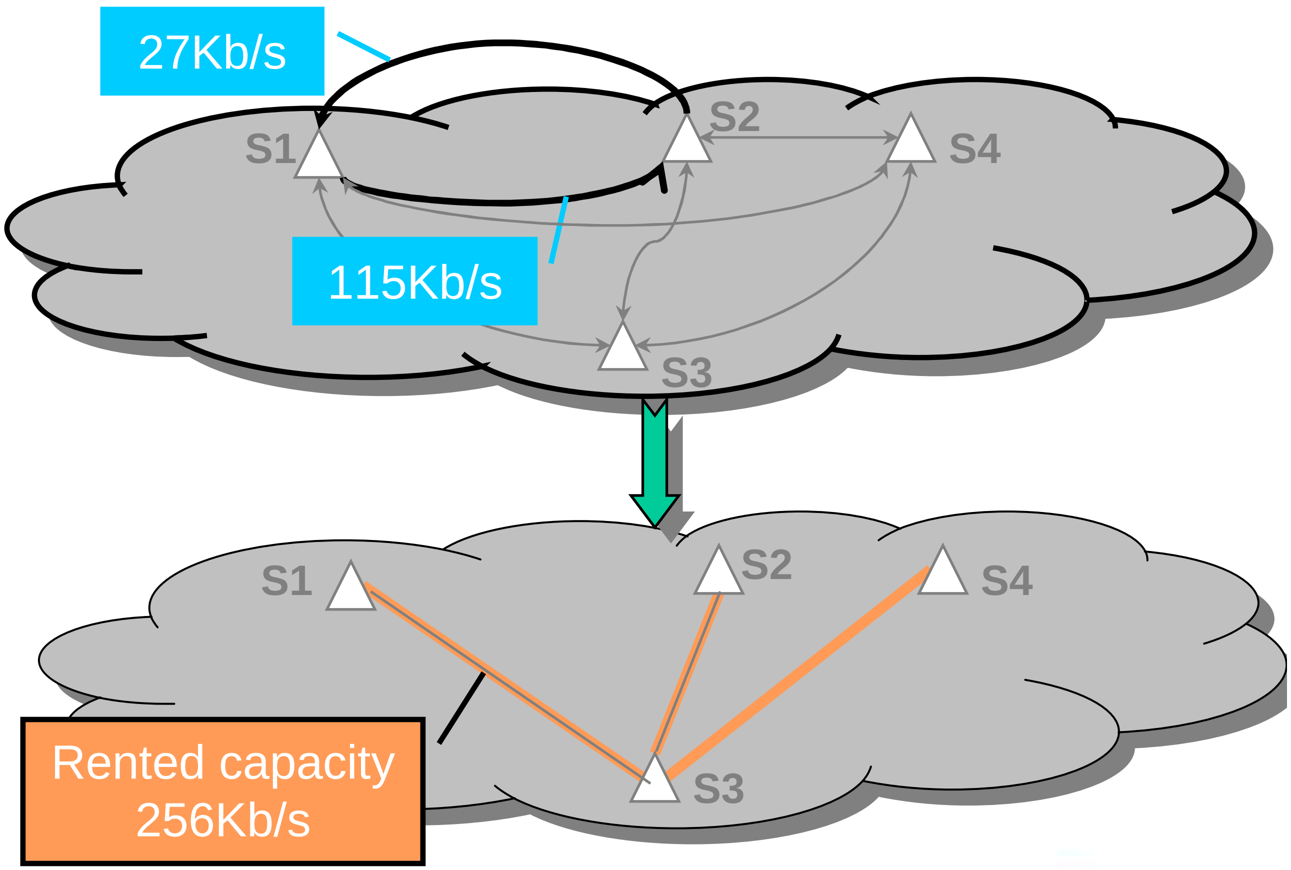
The Problem (2)

Data:

- Customer traffic demands
- Possible links, capacities and costs

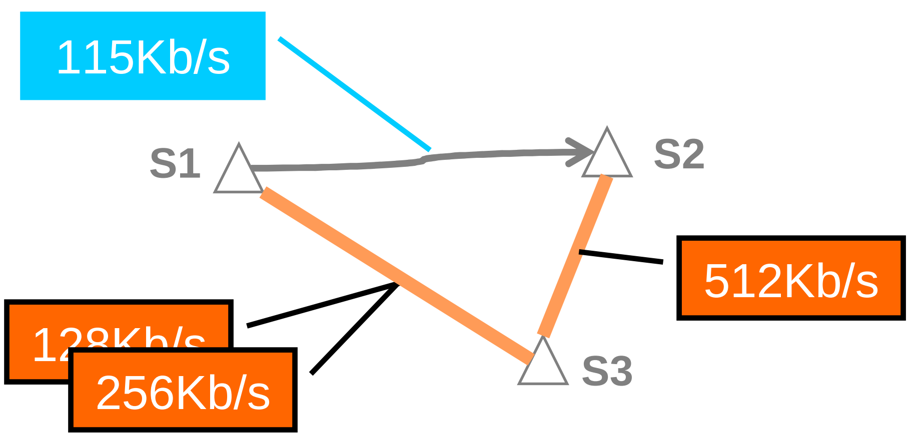
Result:

- **Minimal cost network able to simultaneously respond to all the demands**
- **Route for each demand**



The Problem (3)

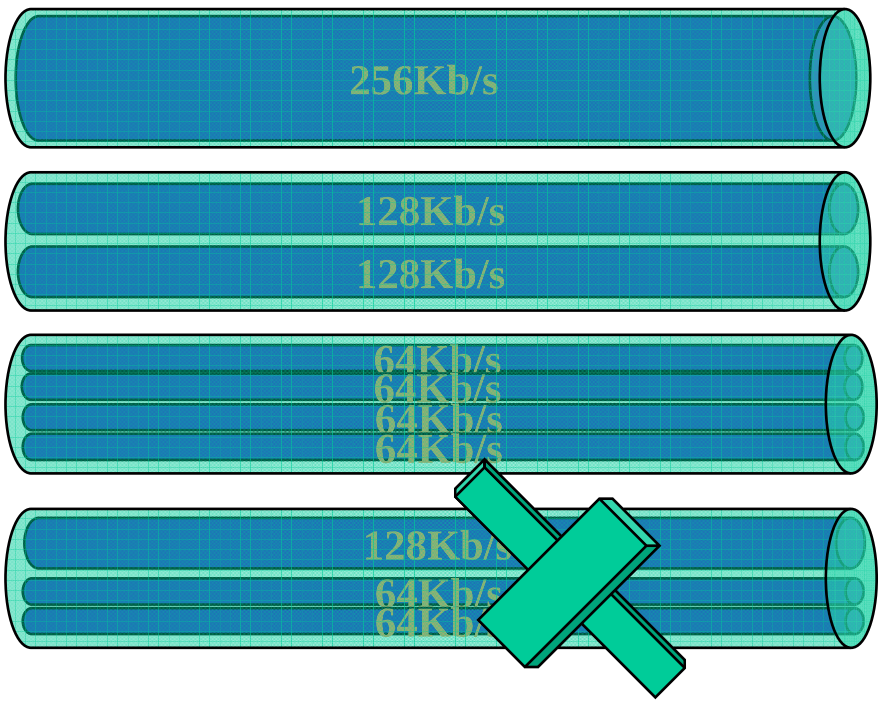
- Cost minimization principle
  - Traffic demands share link capacities



The Problem (4)

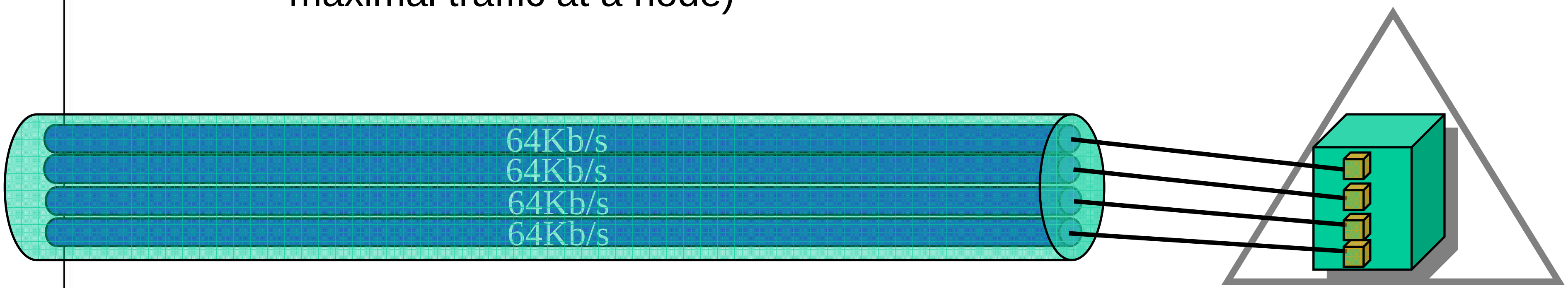
Demands share links

- $\sum$ demands$_{i \rightarrow j}$ $\leq$ capacity$_{i \rightarrow j}$
- Technological constraints



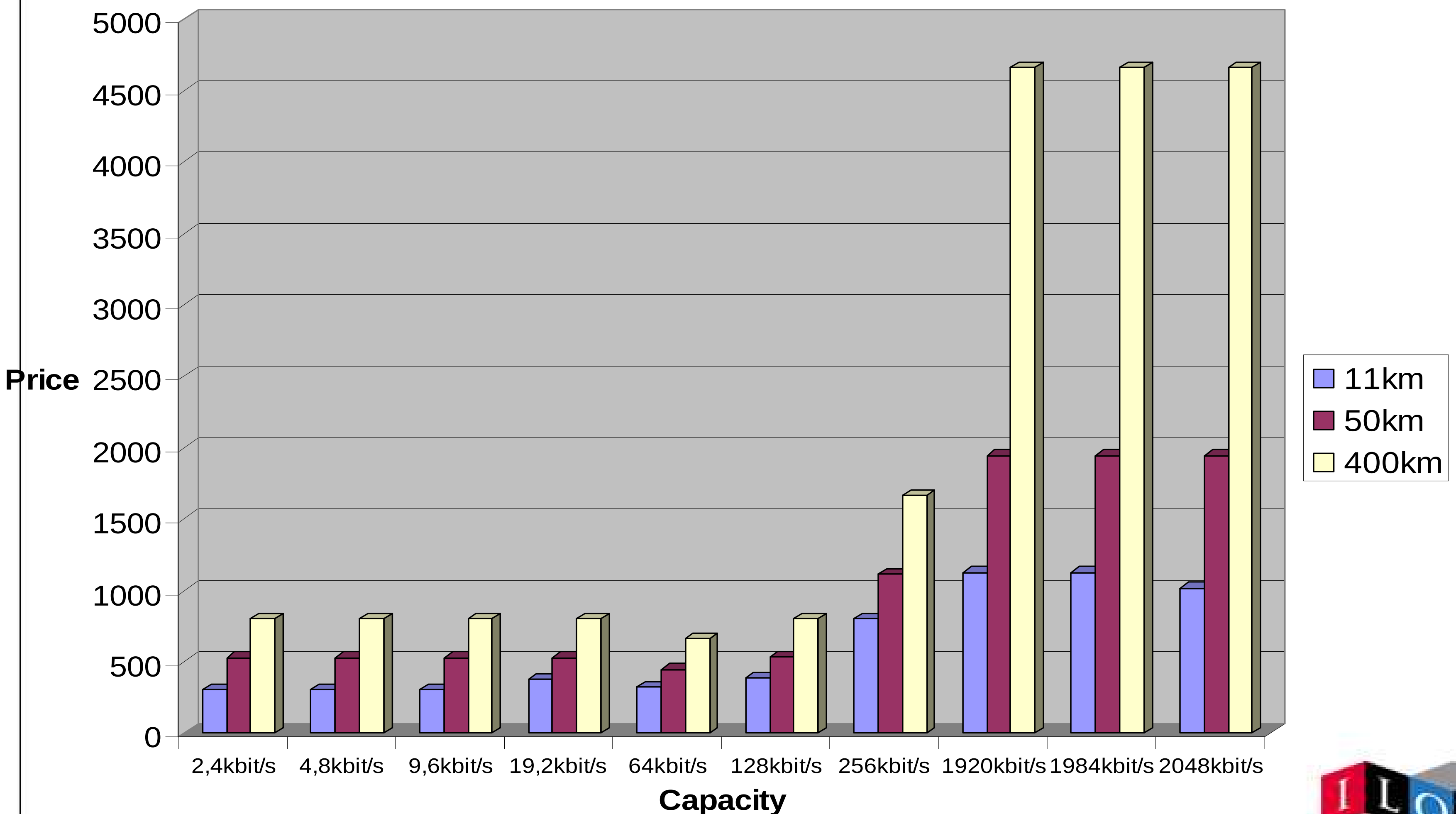
The Problem (5)

- Side constraints
  - Quality of service
  - Reuse of existing equipment (limit on the number of ports, maximal traffic at a node)



  - Commercial and legal constraints
  - Possible future network evolution
  - Network management (e.g., traffic concentration)

Benchmark Elaboration

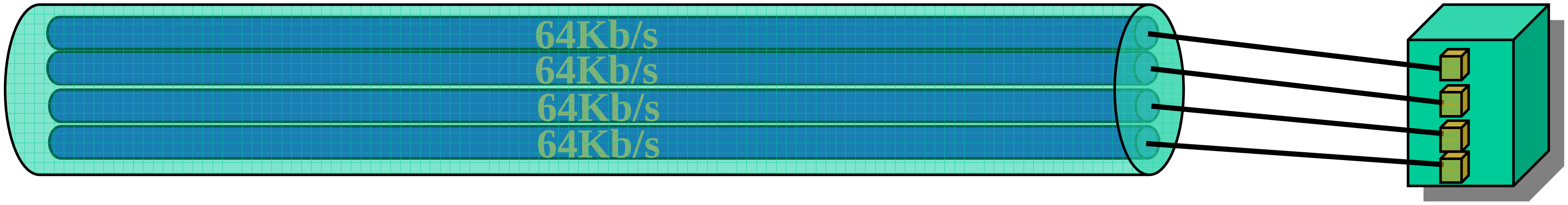


An Extensive Benchmark

- Built to test algorithm robustness
  - 21 instances organized in 3 series of 7
- Size
  - 4 to 25 nodes
  - 2*6 to 2*300 arcs
  - 2 to 25 possible levels of capacity for each arc (some levels being dominated depending on the constraints)
  - 12 to 462 commodities (demands)
- Optional constraints
  - 6 optional constraints, leading to 21*64 = 1344 problems
- Numerical characteristics

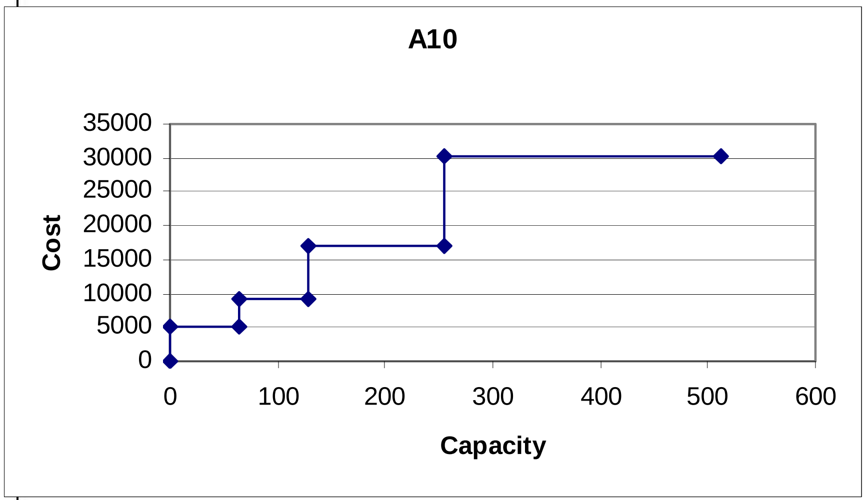
Optional Constraints

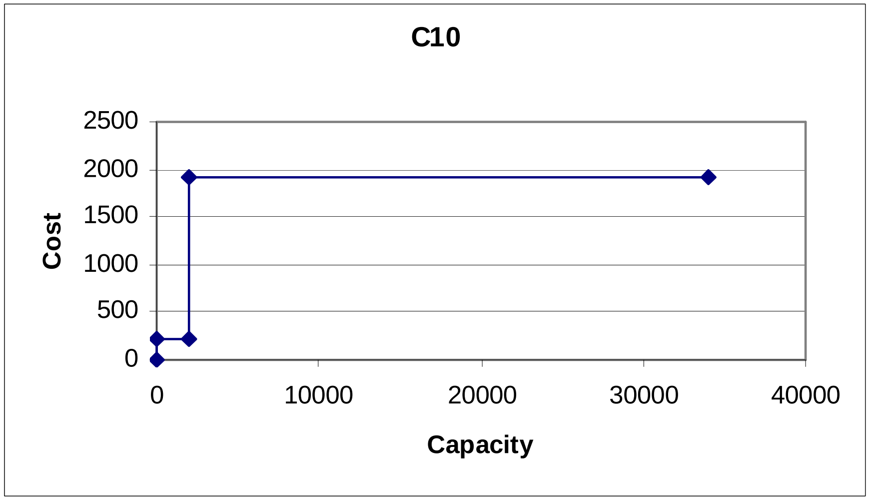
- **Security:** some commodities to be secured cannot go through unsecured nodes and links
- **No line multiplication:** at most one line per arc.
- **Symmetric routing:** demands from node p to node q and demands from node q to node p are routed on symmetric paths.
- **Number of bounds (hops):** the number of arcs of the path used to route a given demand is limited.
- **Number of ports:** the number of links entering into or leaving from a node is limited.

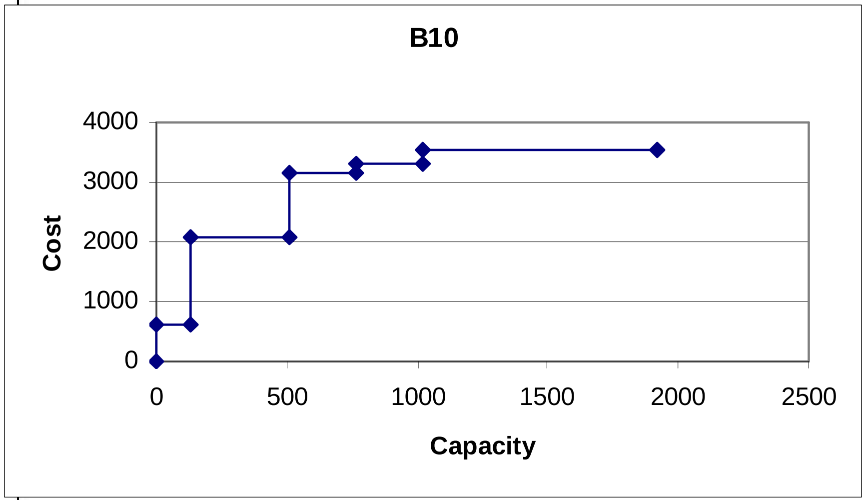


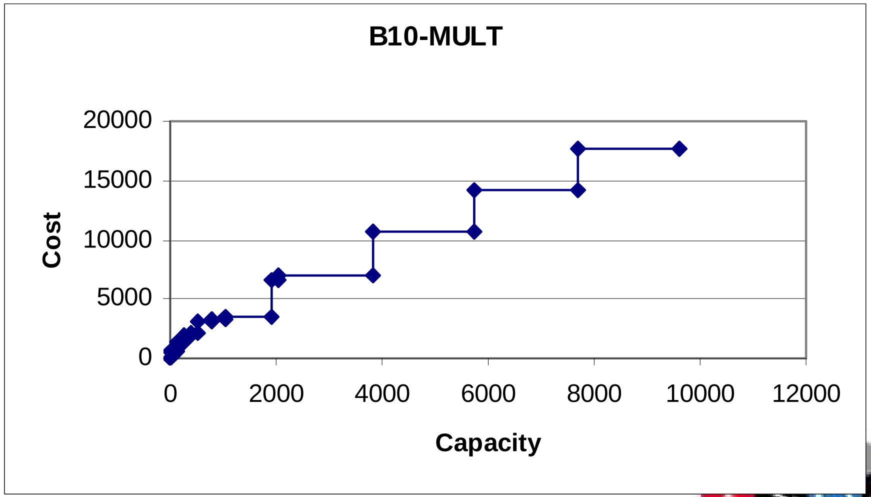
- **Maximal traffic:** the total traffic managed by a given node is limited.

Numerical Characteristics (1)

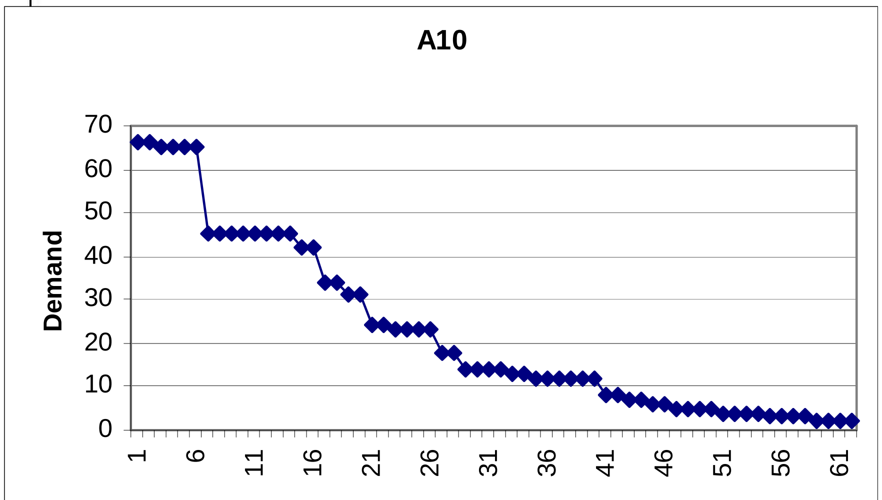


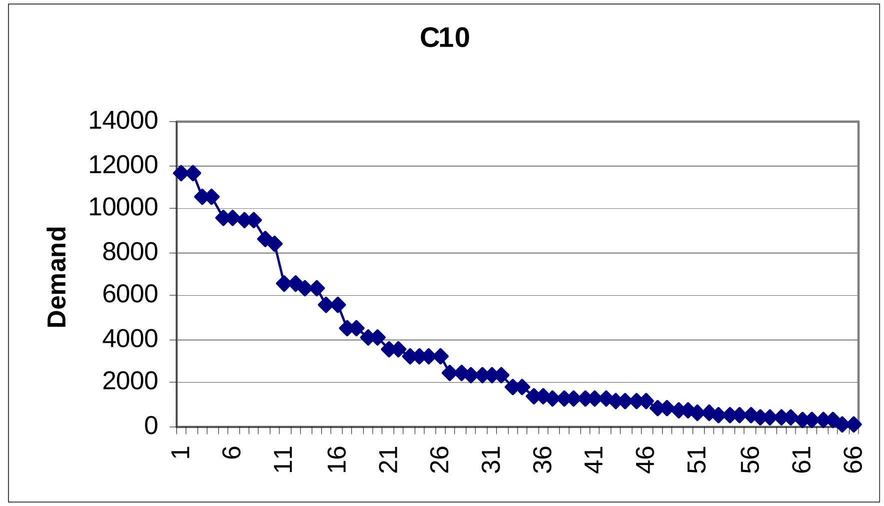


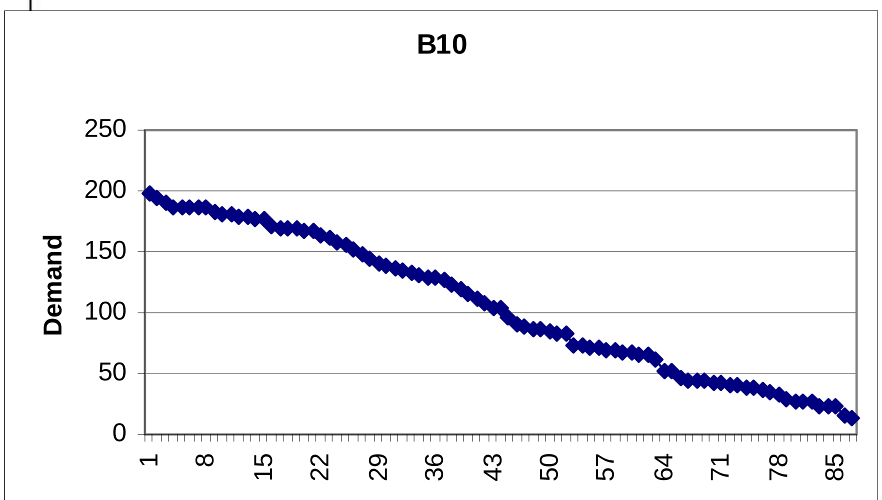


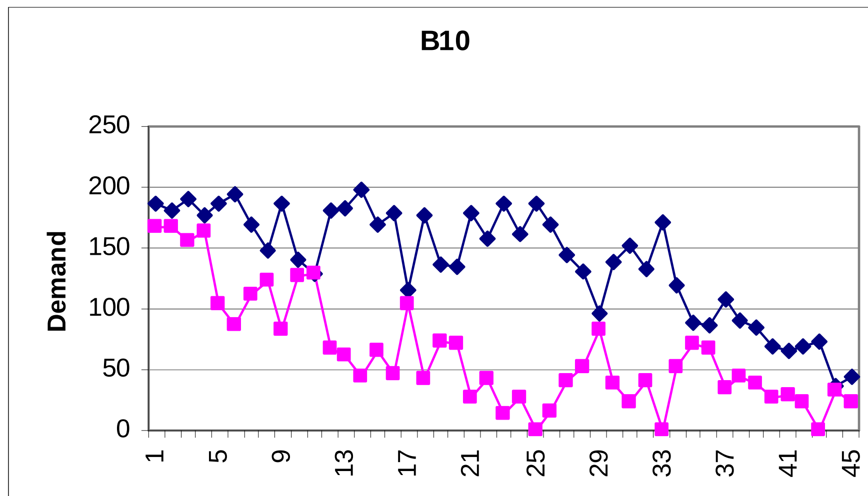


Numerical Characteristics (2)









Particular Cases

- Loop network (C10)
- Unidirectional lines (C11)
- Extension of an existing network (C16)
- Several commodities (with different « security » and « number of hops » constraints) between different sites and a central site (C20)

Input File (1)

```
4 6 12
0 256 1   2   2
1 256 0 256 256
2 256 1   2   2
3 256 0 256 256
0 1 3 64 64 6423 0 3 0 128 128 11853 0 1 1 256 256 22779 0 1 0
0 2 3 64 64 5496 0 3 0 128 128  9999 0 1 1 256 256 19071 0 1 0
0 3 3 64 64 3865 0 3 0 128 128  6831 0 1 1 256 256 12829 0 1 0
1 2 3 64 64 4698 0 3 0 128 128  8403 0 1 1 256 256 15879 0 1 0
1 3 3 64 64 5838 0 3 0 128 128 10683 0 1 1 256 256 20439 0 1 0
2 3 3 64 64 4884 0 3 0 128 128  8775 0 1 1 256 256 16623 0 1 0
```

Input File (2)

```
0 1 65 2 1
1 0 65 2 1
0 2 23 2 0
2 0 23 2 0
0 3 14 2 0
3 0 14 2 0
1 2 42 2 0
2 1 42 2 0
1 3  7 2 0
3 1  7 2 0
2 3  4 2 0
3 2  4 2 0
```

Solution File (Symmetric Case)

```
6   6   36226
0   1   1   256 256 22779   0
0   2   0   0   0   0   0
0   3   1   64  64  3865    0
1   2   1   64  64  4698    0
1   3   0   0   0   0   0
2   3   1   64  64  4884    0
0   1   65  65  1   0 1
0   2   23  23  2   0 3 2
0   3   14  14  1   0 3
2   1   42  42  1   2 1
3   1   7   7   2   3 0 1
3   2   4   4   1   3 2
```

Comparison / Other Benchmarks

	ROCOCO	Gabrel et al.	Rothlauf et al.	Gendron & Crainic
Nodes	4 to 25	8 to 20	15 to 26	20 to 100
Arcs	2*6 to 2*300	12 to 37	210 to 650	230 to 1600
Capacities	2 to 5*5	6 (average)	3 to 5	1
Commodities	12 to 462	56 to 380	15 to 240	10 to 200
Routing	Mono-routing	Multi-flow	Tree	Multi-flow
Cost functions	Scale	Scale	Scale	Fix + variable cost
Constraints	Security		Tree	Limits for each
	Symmetry			commodity on
	Number of arcs			each arc
	Number of ports			
	Node capacities			
	Existing network			
Instances	21*64 = 1344	50	4	18

Rococo: what do we learn?

- The basic entity of this problem is a path and not the arc of the path
- This is certainly a good information for some other problems of the same type
- We developed a graph VAR API

Benchmark of different domains

- It is not always easy to use benchmarks of other domains in CP
- Because CP exploits the structure of the problem and not the other technique. We need the original problem to try different kind of models. This is not the case for SAT

Benchmark of different domains

- We have a problem to compare the results with other domains, because some instances are hard in CP and easy for the other domains and conversly:
  - 2 examples: Sports scheduling (vs MIP) and Latin Square Completion (vs SAT)
  - SAT is able to solve some very hard instances of Latin Square Completion but cannot solve empty Latin Square! Or Latin Square of Size 70
- Difficult to define an hard problem, because a problem is hard in respect to one technology

General considerations

- **When solving a problem in CP:**
- Potential performance gain:
  - data structure optimization (code): x 10
  - search strategies: x 1 000
  - model : x 1 000 000
- Repartition of effort for ROCOCO
  - data structure optimization (code): 75 %
  - search strategies: 20 %
  - model: 5 %

General considerations

- **When solving a problem in CP:**
- Potential performance gain:
  - data structure optimization (code): x 10
  - search strategies: x 1 000
  - model : x 1 000 000
- Repartition of effort for ROCOCO
  - data structure optimization (code): 75 %
  - search strategies: 20 %
  - model: 5 %
- Objective of CP Optimizer
  - data structure optimization (code): 0 %
  - search strategies: 10 %
  - model: 90 %

Conclusion

- If P = NP then CP has great chance to disappear
- If P $\neq$ NP then whe can only shift the exponential
- Either we do theoretical research or we do applied research

Conclusion

- Theoretical research should be based on something scientifically strong and not be only an experimental research on random problems.
- Doing something which has no real world application is not doing theoretical research. It is fortunately more complex than that
- Don't do NAAR: Non Applicable Applied Research! (C. Allegre from someone else)

Conclusion

- Applied research should be based on realistic problems. They can be small but they have to correspond either to a known issue or to a problem/subproblem. They also should not be solved by a simple and known CP model.
- Don’t forget that if we refuse applications then only the theoretical part remains!