ILOG
Changing the rules of business

Performance variability in mixed integer programming

Emilie Danna
ILOG CPLEX R&D
edanna@ilog.com

MIP 2008

Example: 10 teams, CPLEX 11, Linux

Tried aggregator 1 time.
MIP Presolve eliminated 20 rows and 425 columns.
Reduced MIP has 210 rows, 1600 columns, and 9600 nonzeros.
Presolve time = 0.01 sec.
Clique table members: 170.
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: none, using 1 thread.
Root relaxation solution time = 0.05 sec.

Nodes					Cuts/		
Node	Left	Objective	IInf	Best Integer	Best Node	ItCnt	Gap
0	0	917.0000	140		917.0000	1100	
0	0	924.0000	165		Cuts: 50	1969	
0	0	924.0000	167		Cuts: 17	2348	
0	0	924.0000	175		Cliques: 14	2731	
* 0+	0			924.0000	924.0000	2731	0.00%

Clique cuts applied: 16
Zero-half cuts applied: 3
Gomory fractional cuts applied: 1

Solution pool: 1 solution saved.

MIP - Integer optimal solution: Objective = 9.2400000000e+02
Solution time = 0.41 sec. Iterations = 2731 Nodes = 0

Example: 10 teams, CPLEX 11, AIX

```
Tried aggregator 1 time.
MIP Presolve eliminated 20 rows and 425 columns.
Reduced MIP has 210 rows, 1600 columns, and 9600 nonzeros.
Presolve time =    0.00 sec.
Clique table members: 170.
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: none, using 1 thread.
Root relaxation solution time =    0.18 sec.

        Nodes                                  Cuts/
   Node  Left     Objective  IInf  Best Integer     Best Node    ItCnt     Gap

      0     0      917.0000   151                  917.0000     1053
      0     0      924.0000   152                  Cuts: 53     1801
      0     0      924.0000   161                Cliques: 14     2336
      0     0      924.0000   163                Cliques: 12     2609
      0     2      924.0000   163                  924.0000     2609
*   100+   96                        952.0000      924.0000    12316    2.94%
   1000   520      926.7273    85      952.0000      924.0000    97832    2.94%
*  1425     0      integral     0      924.0000      924.0000   122948    0.00%

Clique cuts applied:  12
Zero-half cuts applied:  4
Gomory fractional cuts applied:  2

Solution pool: 2 solutions saved.

MIP - Integer optimal solution:  Objective =  9.2400000000e+02
Solution time =   41.39 sec.  Iterations = 122948  Nodes = 1426
```

Definition (1)

- Performance variability
  - change in performance (solving time, # nodes, # iterations, …)
  - for the same model
  - created by a change in the solver or the environment
  - that is seemingly performance neutral

  - in short, change in performance we do not understand

Definition (2)

- The topic of this talk is not (but is related to)

  - Change in performance due to improving algorithms
  - Change in performance due to change in MIP formulation for the same problem
  - Change in performance due to change in data

Questions

- What is the extent of variability?
- What are the causes of variability?
  - Variability generator
    - How to generate variability in order to study it?
  - Solver
    - What algorithms or implementations cause variability?
    - Can we make solvers more robust and therefore faster?
  - Model
    - What characteristics make a model variable?
- What are the consequences of variability?

Experimental setup (1)

- 368 models that solve to 0.01% gap between 10 and 100 seconds with CPLEX 11
  - Homogeneous set
  - Large enough for statistical analysis

- Performance is measured as the number of simplex iterations
  - We are really interested in the solving time
  - But solving time is difficult to measure precisely
  - The number of iterations is a better proxy than the number of nodes

Experimental setup (2)

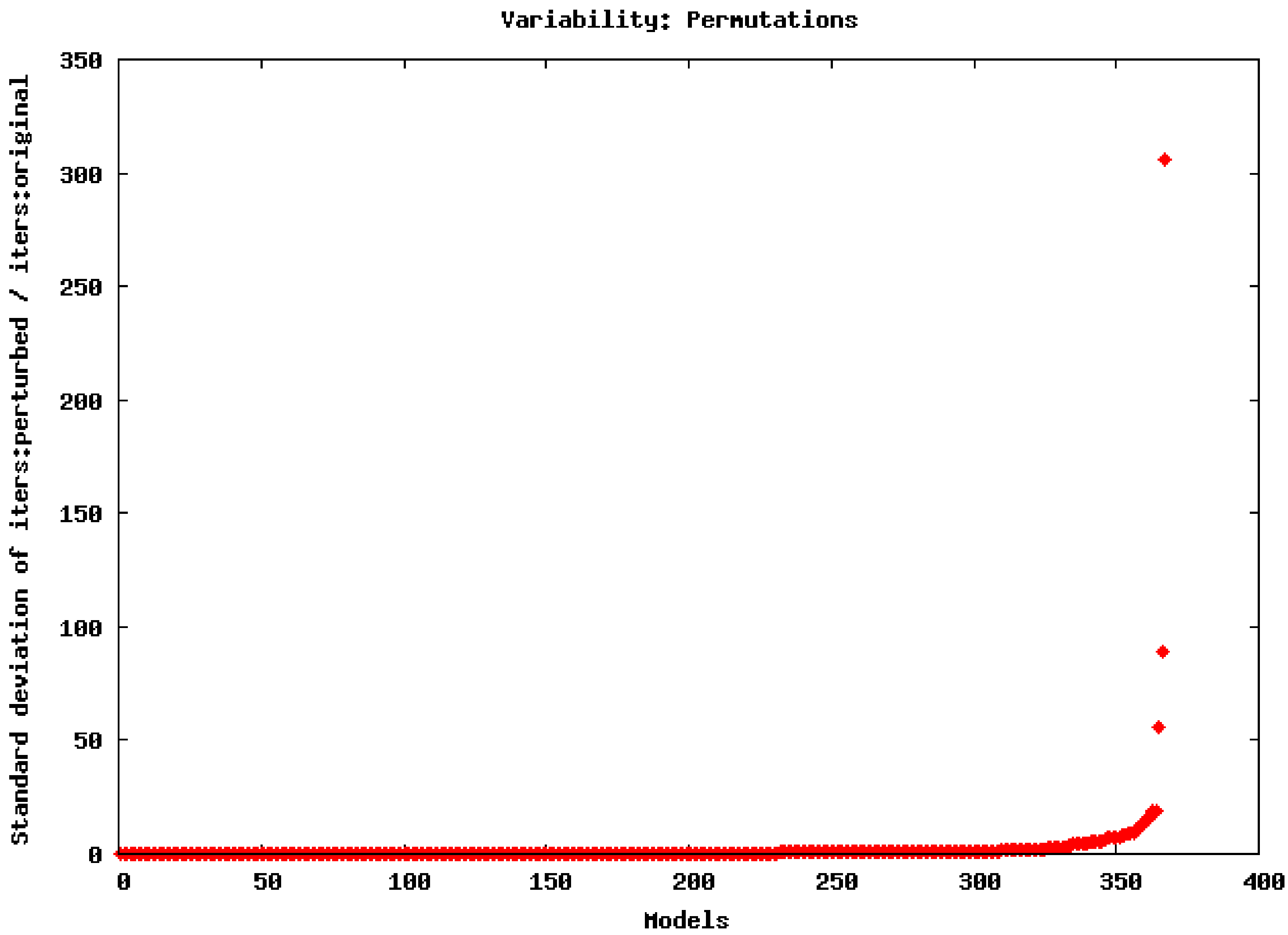
- For each model
  - Performance with the original conditions
    - niters(orig)
  - Performance with perturbed conditions
    - for i = 1...10 different instances of the same variability generator g
    - niters(g,i)
  - Variability (g) is estimated as
    - standard deviation of the sample
    - niters (g,i) / niters (orig), i = 1...10

First variability generator: permutations

Definition and properties

- Random permutations of rows and columns
- Should be performance neutral
- Affects all components of the solver

Distribution of variability (1)



In practice, what does a variability of 'v' mean?

- Variability ~ 0.1: small and reasonable variations

Variability										
0.1053	0.89	1.08	0.86	1.02	0.89	0.88	0.87	0.98	1.09	1.13
0.0900	1.06	1.08	0.97	1.21	1.03	1	1.1	0.91	1.1	1.17
0.1183	0.96	1.18	0.88	0.85	0.88	1	0.96	1.09	0.82	1.08

- Variability ~ 1 : already quite large

Variability										
0.9728	0.69	0.9	2.46	0.67	1.03	0.95	0.9	1.11	2.49	3.47
0.9448	1.7	0.4	2.39	0.9	0.74	0.63	0.35	0.51	3.14	0.81

- Variability > 10 : terrible

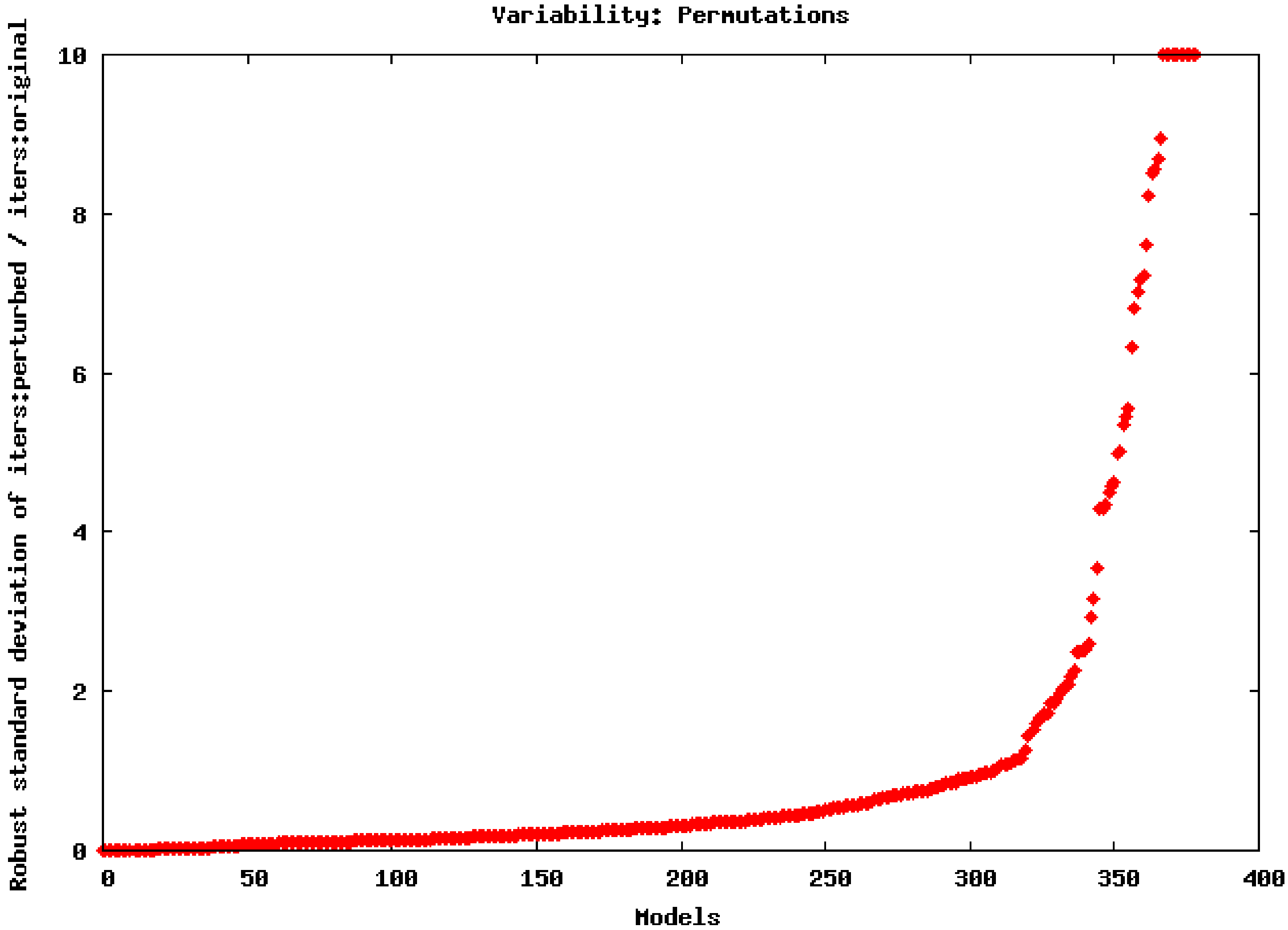
Variability										
10.4249	3.95	23.9	5.11	4.23	22.16	8.45	1.46	30.07	5.09	3.17
18.9003	3.78	2.1	2.35	1.92	3.02	3.72	62.29	1.41	3.5	1.48
88.7852	1.12	229.4	0.92	0.84	0.65	0.67	0.73	0.65	0.59	191.1

Distribution of variability (2)

Median (v)	0.29
v <= 0.1	16.6%
0.1 < v <= 1	64.4%
1 < v <= 10	15.8%
v > 10	3.3%

- Models for which v > 10
  - The number computed is a poor estimation of the true variability for these models
  - In the following, we truncate v > 10 to v = 10 to have a more “robust” estimate

Distribution of variability (3)



Are permutations really performance neutral?

- Performance on permuted vs. original models
  - +30% more time
  - +8% more branch & bound nodes
  - +13% more simplex iterations

- Why?

Locality

- Cache misses when solving the root LP on permuted vs. original models (measured with cachegrind):

	Median	Geometric mean	Impact of one cache miss
L1 data cache	+34%	+36%	~10 cycles
L2 data cache	+4%	+13%	~200 cycles

- The increase in cache misses is a good candidate to explain the increase in time per iteration and time per node
- We presume the locality also affects the discrete components of the MIP solver

Quantifying locality: matrix dispersion

$$A = \left(a_{i,j}\right)_{i=1..m,\, j=1..n}$$

$$nz_j = \#\left\{i \in 1..m,\, a_{i,j} \neq 0\right\}$$

$$\delta_{i,j,j+1} = 1 \text{ if } a_{i,j} = 0 \text{ and } a_{i,j+1} \neq 0 \quad \text{or} \quad a_{i,j} \neq 0 \text{ and } a_{i,j+1} = 0$$

$$\delta_{i,j,j+1} = 0 \text{ otherwise}$$

$$Dispersion(A) = \frac{\sum_{j=1}^{n-1} \sum_{i=1}^{m} \delta_{i,j,j+1}}{2 \sum_{j=1}^{n} nz_j}$$

$$0 \leq Dispersion(A) \leq 1$$

Matrix dispersion and permutations

- +91% dispersion on permuted models compared to original models
  - The order of rows and columns chosen by a human modeler creates matrices with a small dispersion

- But no correlation between the increase in dispersion and the performance degradation

Permutations: SCIP

- SCIP with CPLEX as the LP solver
- Subset of 135 models: models that solve to 0.01% gap with SCIP in less than 500 seconds

Permutations: SCIP vs. CPLEX

	CPLEX	SCIP
Median (v)	0.29	0.24
v <= 0.1	8.1%	19.3%
0.1 < v <= 1	76.3%	66.7%
1 < v <= 10	13.3%	12.6%
v > 10	2.2%	1.5%

- Variability and performance need to be interpreted together
  - SCIP is about 4 times slower than CPLEX on this subset of 135 models (and about 20 times slower on the entire set of 368 models)
- No correlation between variability for CPLEX and SCIP

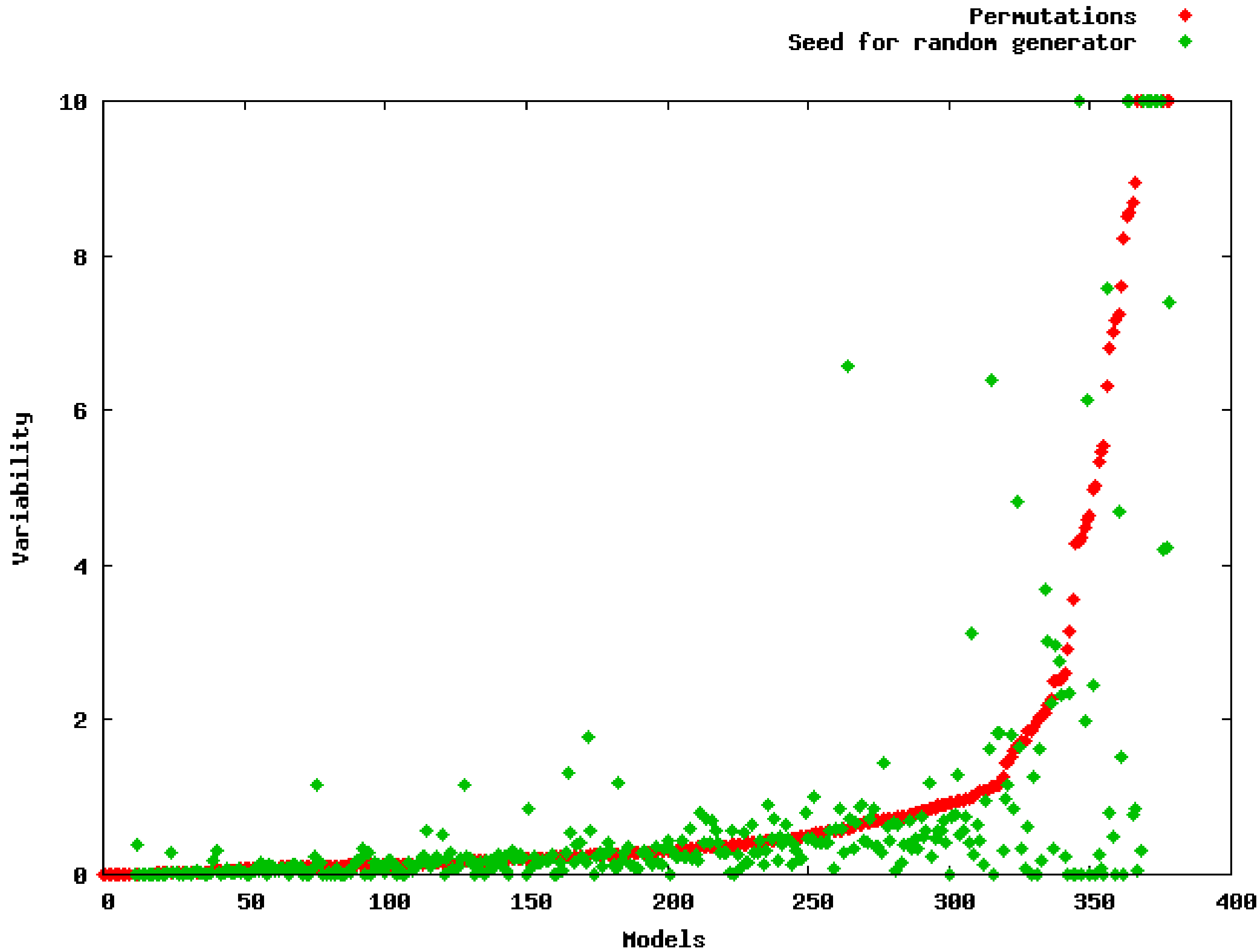
Changing the rules of business

Second variability generator: random generator initialization

Definition and properties

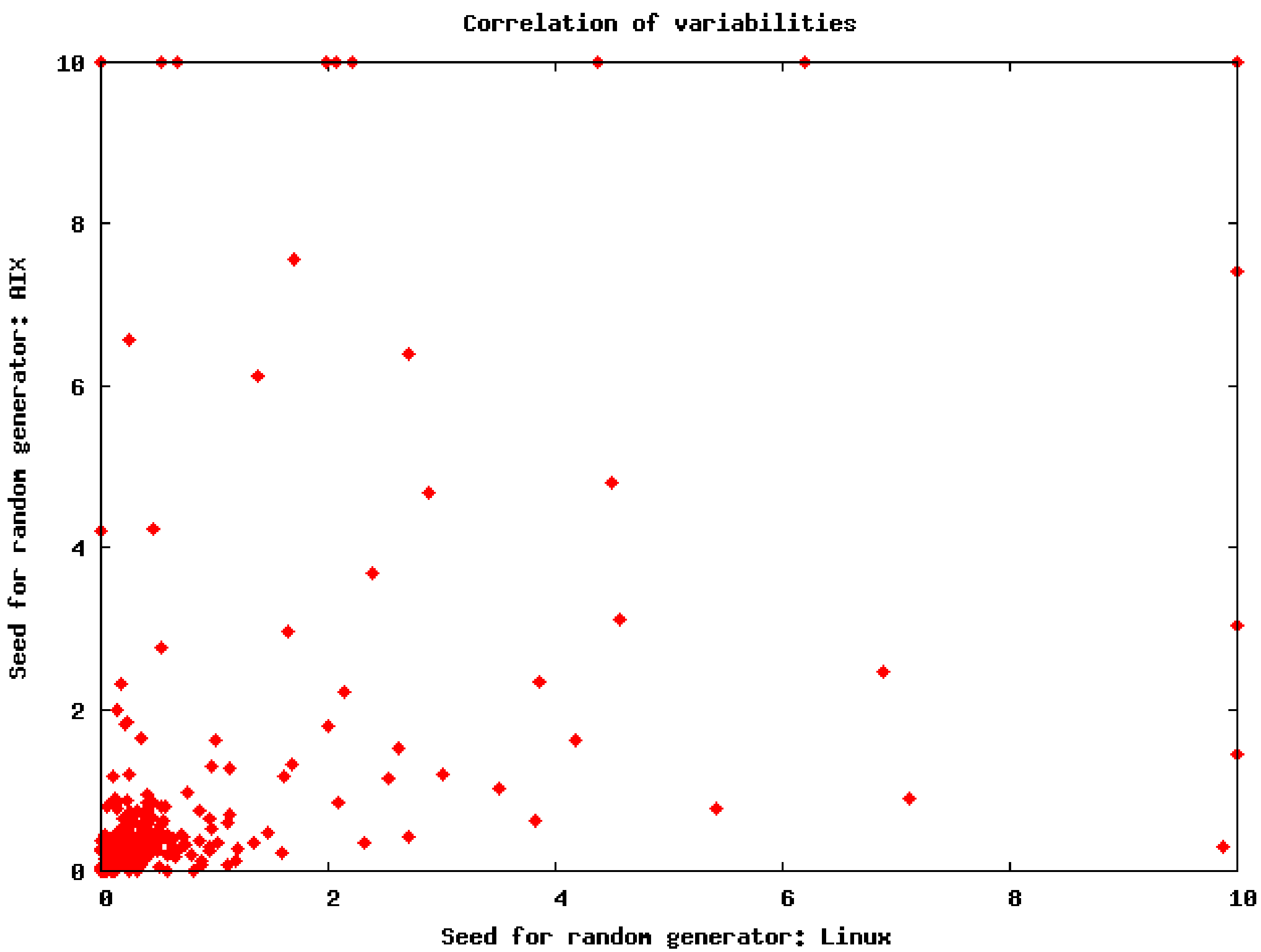
- Change the seed of the random number generator
- Should really be performance neutral
- Affects mainly heuristics
  - But once the path is changed, everything is affected
- Experiments with CPLEX on Linux and AIX

Comparison of variabilities



Correlation = 0.7 (0.59 without truncation)

Difference between AIX and Linux?



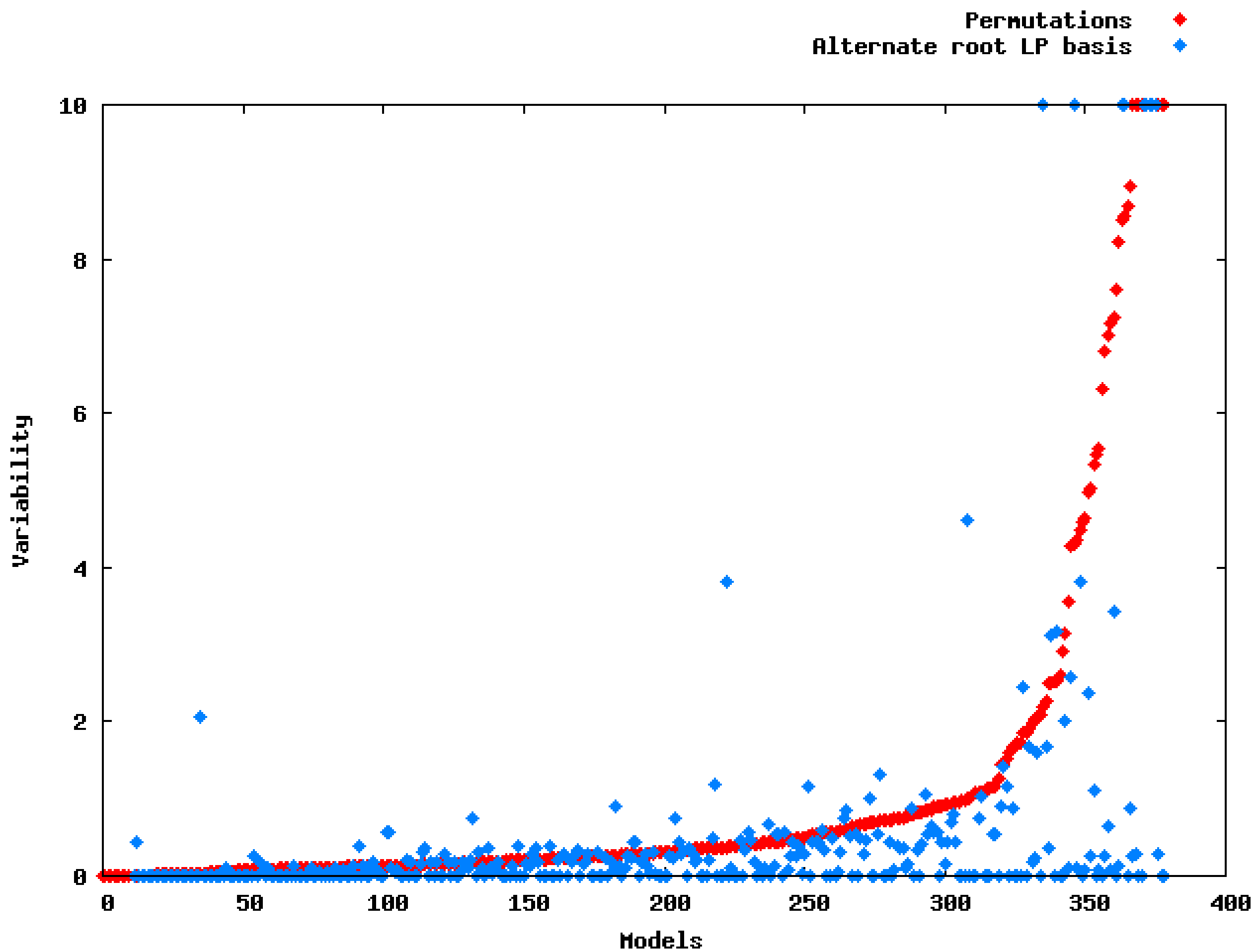
- Correlation = 0.51 (0.04 without truncation)

Third variability generator: degenerate pivots

Definition and properties

- Make a few random dual degenerate pivots at the root
- The root LP basis is different, therefore all LP-based components that follow give different results: cuts, branching, most heuristics, …
- Cannot be applied to some problems
  - Either there are no degenerate pivots
  - Or CPLEX refuses to do those pivots for numerical reasons

Comparison of variabilities



- Correlation = 0.53 (0.93 without truncation)

Conclusions on variability generators (1)

- No clear winner: each generator has its advantages and drawbacks

Generator / property	Permutations	Seed for random generator	Degenerate pivots	Change of platform
Performance neutral	No	Yes	Yes	Yes
Many instances of the generator can be applied	Yes	Yes	Yes / No	No
Applies to all types of problems	Yes	No	No	Yes
Affects all components of the solver	Yes	No	Almost	Yes

Conclusions on variability generators (2)

- Variability depends on the model and the solver
- Given a model and a solver, variability does not depend much on the generator

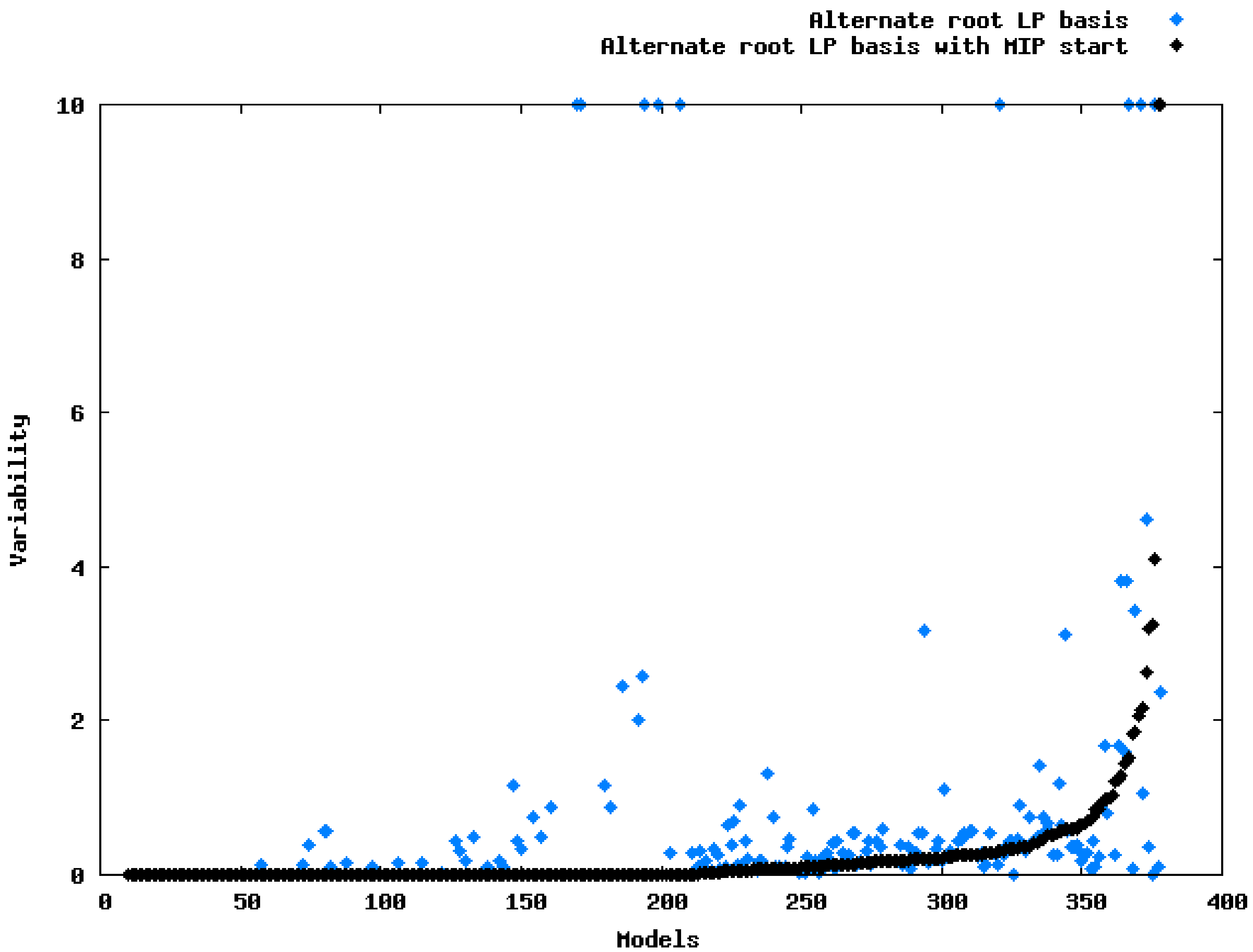
Changing the rules of business

Causes of variability

Possible causes

- Component of the solver
  - Heuristics, branching, cuts, …
- Aspect of optimization
  - Is obtaining the optimal solution less or more robust than proving its optimality?
- Characteristic of the model
  - Landscape of optimal solutions, numerical instability, …

Variability when the optimal solution is known

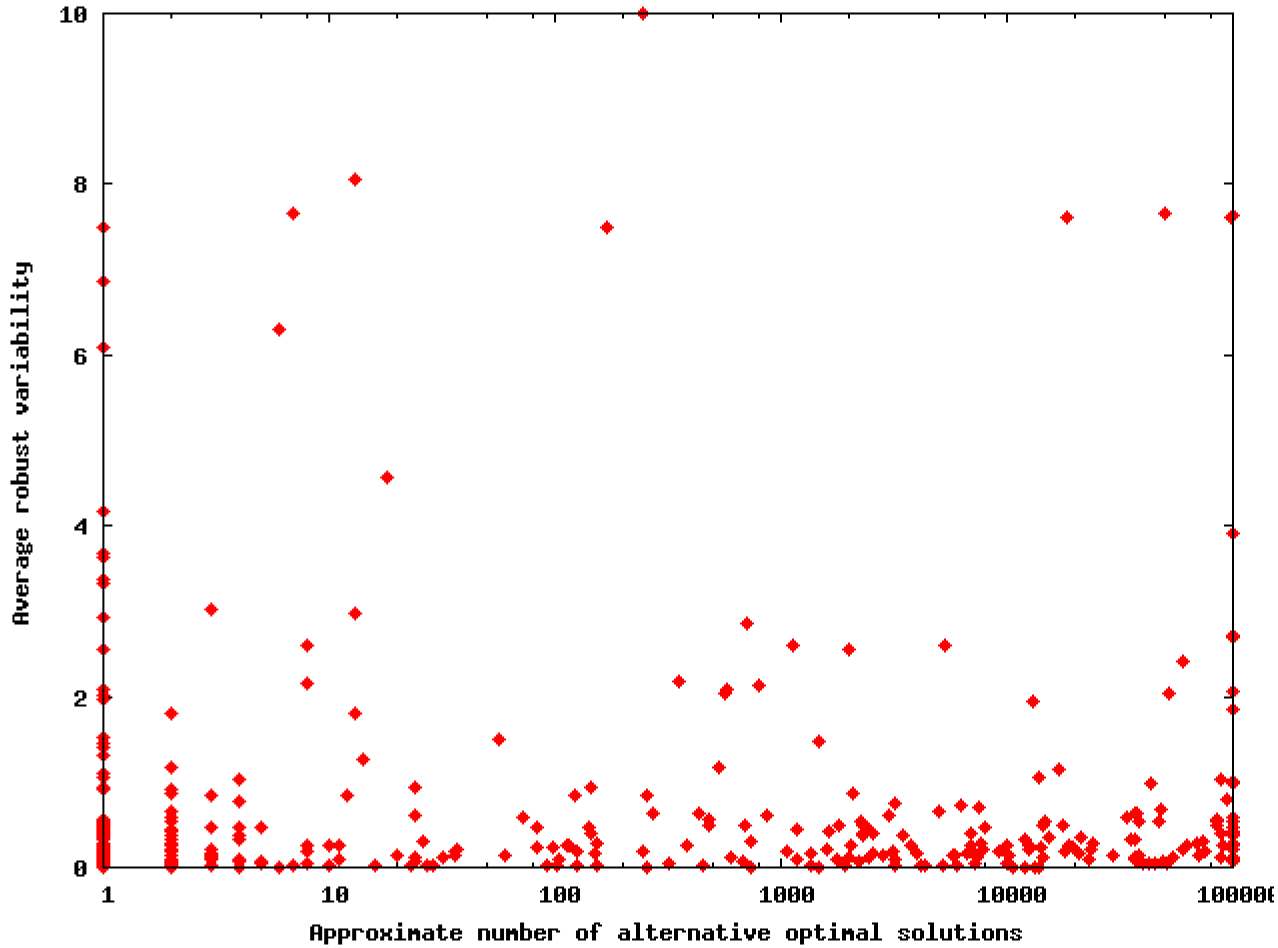


Variability when the optimal solution is known

	CPLEX without MIP start	CPLEX with the optimal MIP start
Median (v)	0.1019	0.0001
v <= 0.1	51.9%	65.8%
0.1 < v <= 1	39.4%	29.1%
1 < v <= 10	6.2%	4.6%
v > 10	2.4%	0.5%

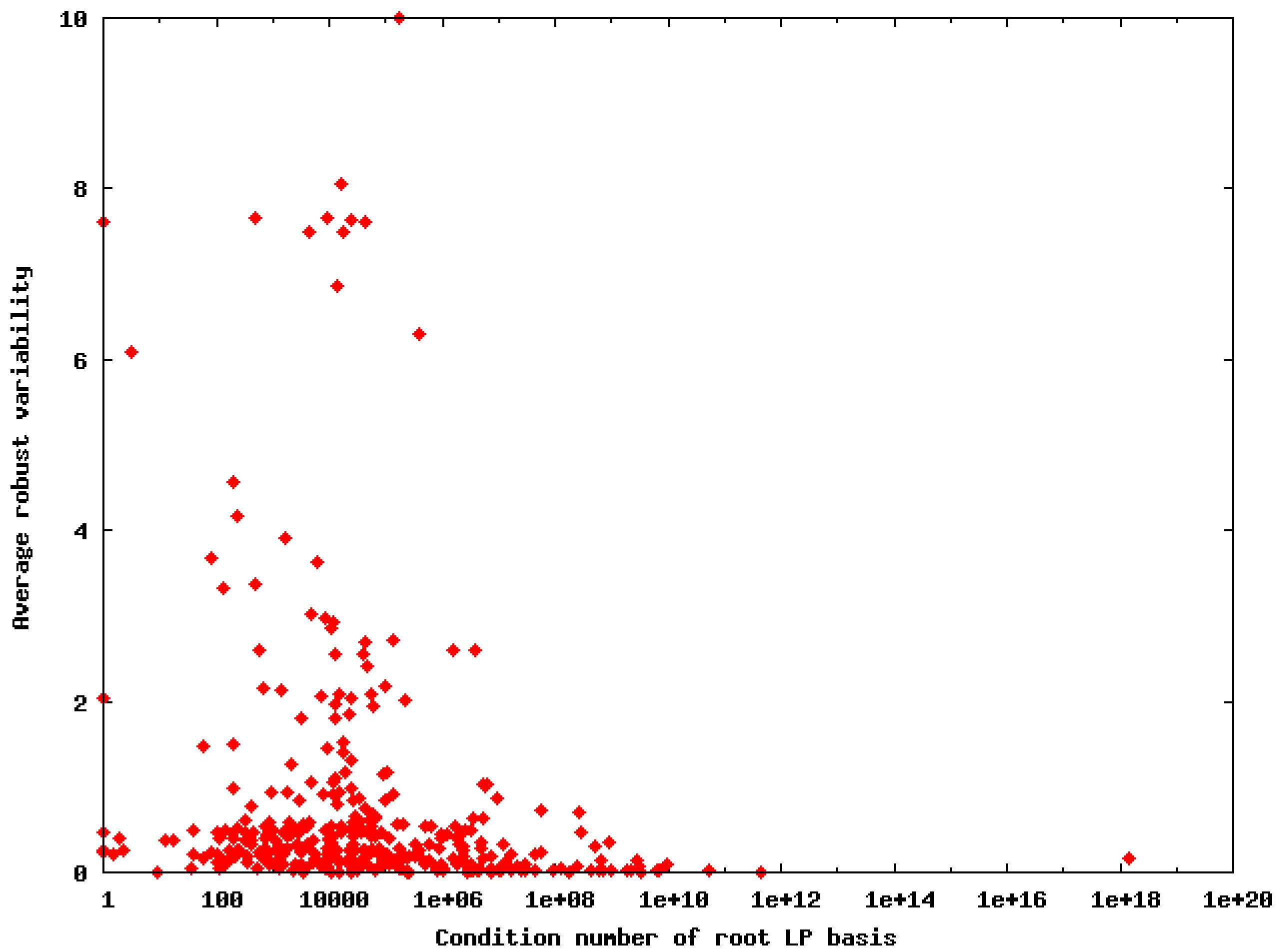
- Finding the optimal solution is a significant cause of variability, especially for extreme cases

Variability and number of alternative optima



- No correlation

Variability and numerical instability



- No correlation

Conclusions on causes of variability

- One known factor: finding the optimal solution
- Many unknown causes
- Future work: looking at the correlation between variability and a given model characteristic is simplistic
  - The correlation is very much influenced by extreme cases
  - Variability might be determined by several factors at once
  - Looking in more details at the results (did not find the essential cut, did not find the optimal solution, did not branch on the right variable, …?) should give more insight

Consequences of variability

Consequences for benchmarking (1)

- The performance difference between code A and code B measures
  - The true effect of the code change
  - Noise (variability)

- The analysis of benchmarking results needs to distinguish between the two

Consequences for benchmarking (2)

- Large model sets
  - Or, artificially increase the size of test sets with a variability generator
- Statistical analysis of results
  - The majority of papers use descriptive statistics
  - Performance profiles are a step in the right direction
  - But benchmarking observations provide only an estimation of the true effect
  - Therefore, we need inferential statistics (statistical tests, confidence intervals, …) to answer questions such as:
  - How likely is it that the performance difference observed is created by variability rather than by my algorithmic change?

Consequences for R&D

- Variability is annoying

- But it is an opportunity for
  - performance improvement
  - better understanding what makes optimization hard in practice