Parallelism in Linear and Mixed Integer Programming

Ed Rothberg

GUROBI OPTIMIZATION

Problem Statement – LP

A *linear program* (LP) is an optimization problem of the form

$$\begin{aligned} &\text{Minimize} && c^T x \\ &\text{Subject to} && Ax = b \\ & && l \le x \le u \end{aligned}$$

GUROBI OPTIMIZATION

Problem Statement – MIP

A *mixed-integer program* (MIP) is an optimization problem of the form

$$\begin{aligned} &\textit{Minimize} && c^T x \\ &\textit{Subject to} && Ax = b \\ & && l \leq x \leq u \\ &\text{some or all } x_j \text{ integer} && \end{aligned}$$

Three Important Characteristics

- Broadly applicable
- Computationally demanding
- Solutions have significant financial value
  - Can be worth millions of $'s

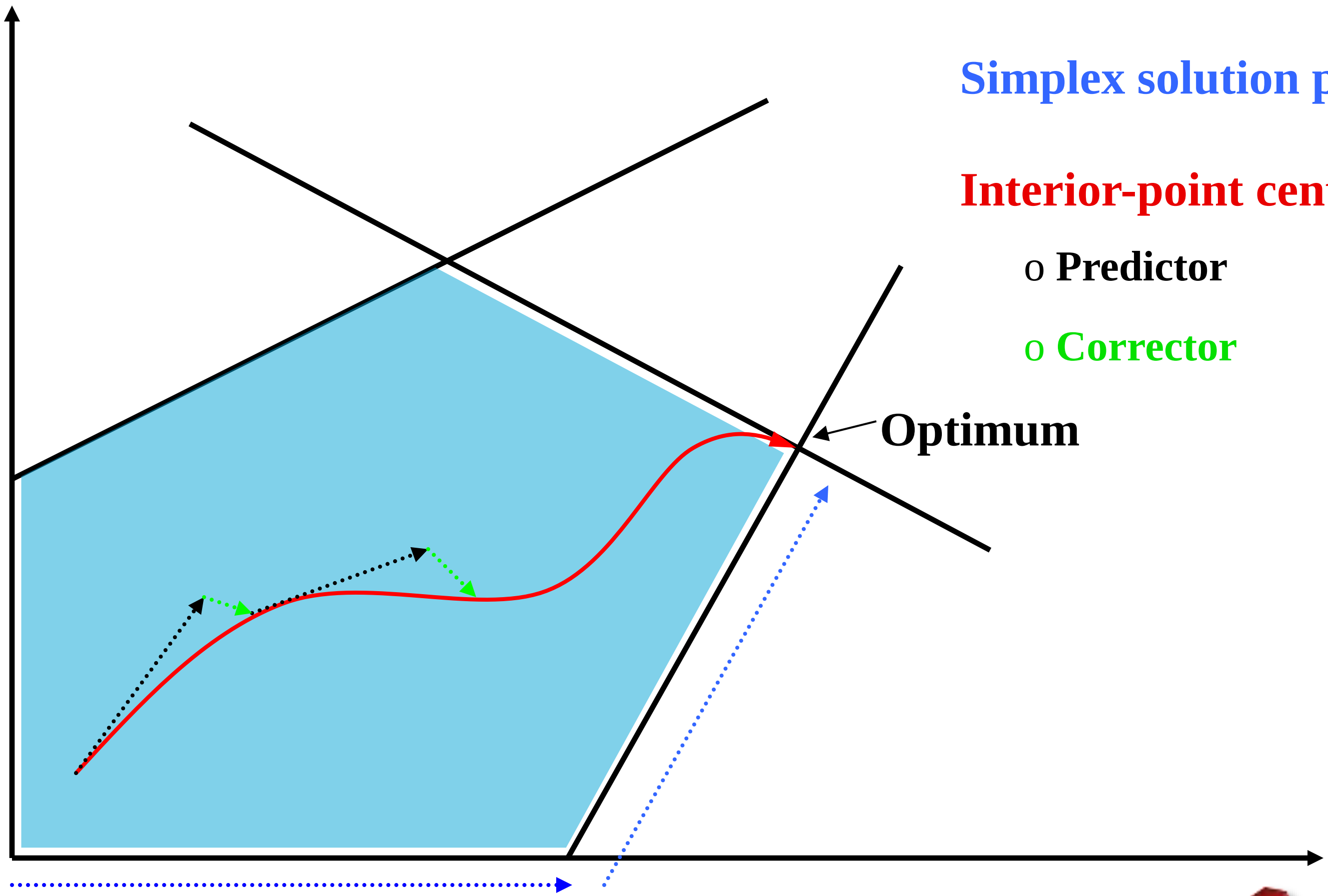
Customer Applications

(Q4 2011–Q3 2012)

- Accounting
- Advertising
- Agriculture
- Airlines
- ATM provisioning
- Compilers
- Defense
- **Electrical power**
- **Energy**
- **Finance**
- Food service
- Forestry
- Gas distribution
- Government
- Internet applications
- **Logistics/supply chain**
- Medical
- Mining
- National research labs
- Online dating
- Portfolio management
- Railways
- Recycling
- Revenue management
- Semiconductor
- Shipping
- Social networking
- Sourcing
- Sports betting
- Sports scheduling
- Statistics
- Steel Manufacturing
- Telecommunications
- Transportation
- Utilities
- **Workforce scheduling**

GUROBI OPTIMIZATION

Linear Programming



LP *Mostly* a Solved Problem

SGM: Schedule Generation Model

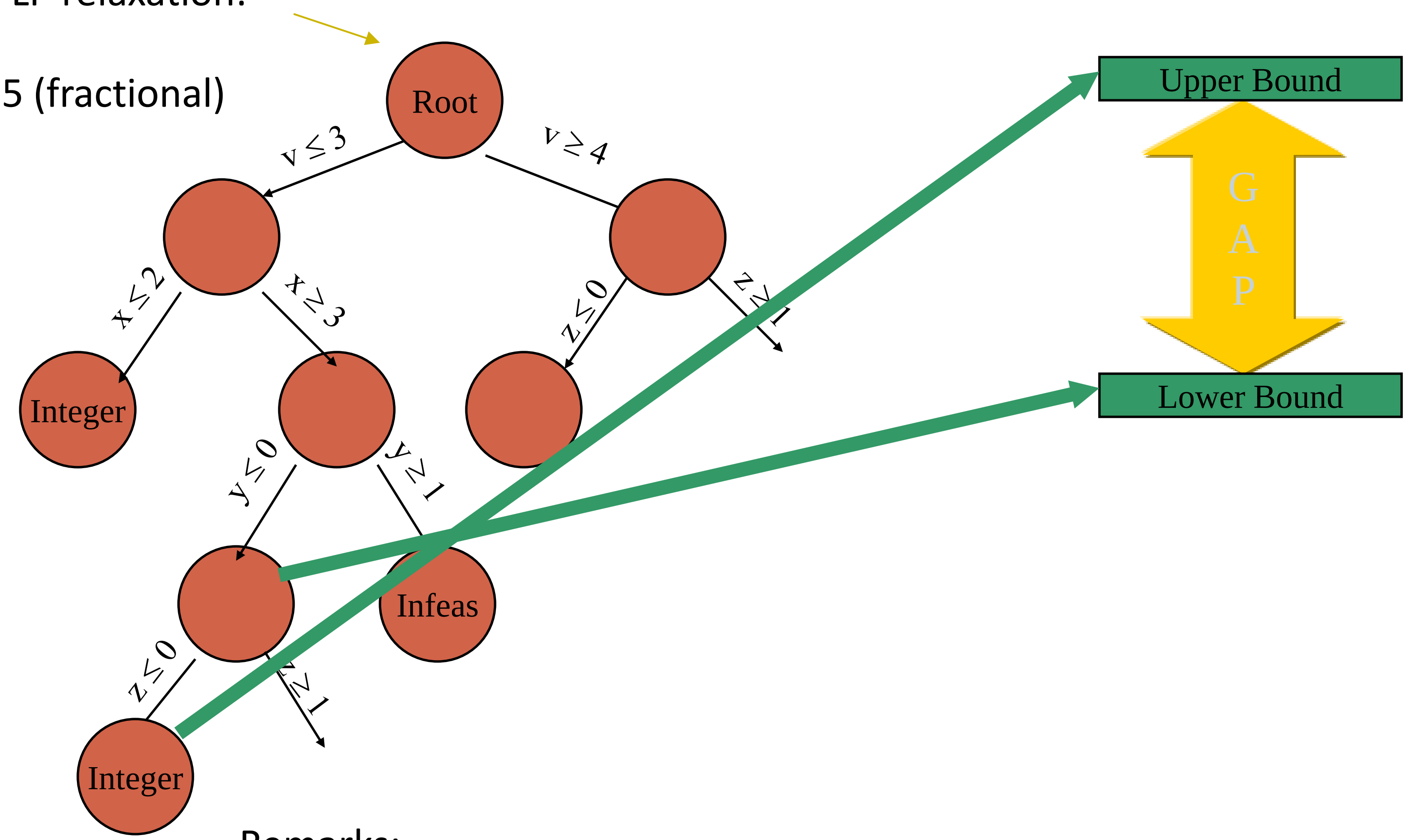
157323 rows, 182812 columns

- LP relaxation at root node:
  - 18 hours
- Branch-and-bound
  - 1710 nodes, first feasible
  - 3.7% gap
  - Time: 92 days!!
- MIP does not appear to be difficult: *LP can be a bottleneck*

MIP solution framework:
LP based Branch-and-Bound

Solve LP relaxation:

v=3.5 (fractional)



Remarks:

(1) GAP = 0 $\Rightarrow$ Proof of optimality

(2) In practice: good quality solution often enough

MIP Definitely Not a Solved Problem

A customer model: 44 constraints, 51 variables, maximization
51 general integer variables (*and no bounds*)

Branch-and-bound:	Initial integer solution	-2186.0
	Initial upper bound	-1379.4

…after 1.4 days, 32,000,000 B&B nodes, 5.5 Gig tree

Integer solution and bound: UNCHANGED

Financial Impact

- Example: NFL
  - Profitability of a $9B company heavily dependent on the solution to one extremely difficult MIP model
- Many other examples

Throw Hardware at the Problem?

- The landscape…
  - Broadly applicable
  - Computationally demanding
  - Solutions have significant financial value
- Plus…
  - "Obvious" sources of parallelism in the algorithms
- Yet…
  - Parallel computing has had a very limited impact in practice

GUROBI OPTIMIZATION

Parallelism in Linear Programming

Simplex Steps

- Maintain a basis B
  - And a basis factorization B=LU
- In each iteration:
  - Choose entering variable
  - Compute direction ($\Delta x = B^{-1} A_{*j}$)
  - Compute step length
  - Update basis and basis factor
- Periodically recompute B=LU

Barrier Steps

- Pre-compute a fill-reducing ordering for $A\,\theta^{-1}\,A'$
- In each iteration:
  - Form $A\,\theta^{-1}\,A'$
  - Factor $A\,\theta^{-1}\,A' = L\,D\,L'$
  - Solve $L\,D\,L'\,x = b$
  - A few Ax and $A'x$ computations
  - A bunch of vector stuff
- Perform a *crossover* to a basic solution

GUROBI OPTIMIZATION

For Any LP/MIP

- Presolve step to reduce the size of the model
  - Remove fixed variables
  - Remove trivially satisfied constraints
  - Use equalities to eliminate variables
  - Etc.

GUROBI OPTIMIZATION

Comparison of Steps

- Iterations
  - Simplex: cheap, thousands–millions
  - Barrier: expensive, several dozen
- Sparse linear algebra
  - Simplex: triangular solves on a very sparse, constantly changing matrix
  - Barrier: Cholesky factorization of a matrix with static structure
- Parallelism
  - Simplex: no general-purpose parallel algorithm
  - Barrier: Cholesky factorization, triangular solves, matrix-vector multiplies, ordering, …

GUROBI OPTIMIZATION

Performance Comparison

- Run a set of 1242 LP test models
  - Public benchmarks and customer models
- Exclude those that are…
  - **Too easy**: solved in less than 0.01 seconds by both methods
  - **Too hard**: not solved in 2 hours by either method
  - Leaves 809 models
- Compute geometric mean of runtime ratios

Performance Comparison

- Results:

  Gurobi 5.6, quad-core i7-3770K processor

  Barrier run on 4 cores, includes crossover

	Wins	GeoMean
Dual simplex	541	1.00
Barrier	483	0.95

- Simplex wins more often, but barrier is 5% faster on average

GUROBI OPTIMIZATION

Exclude Simpler Models

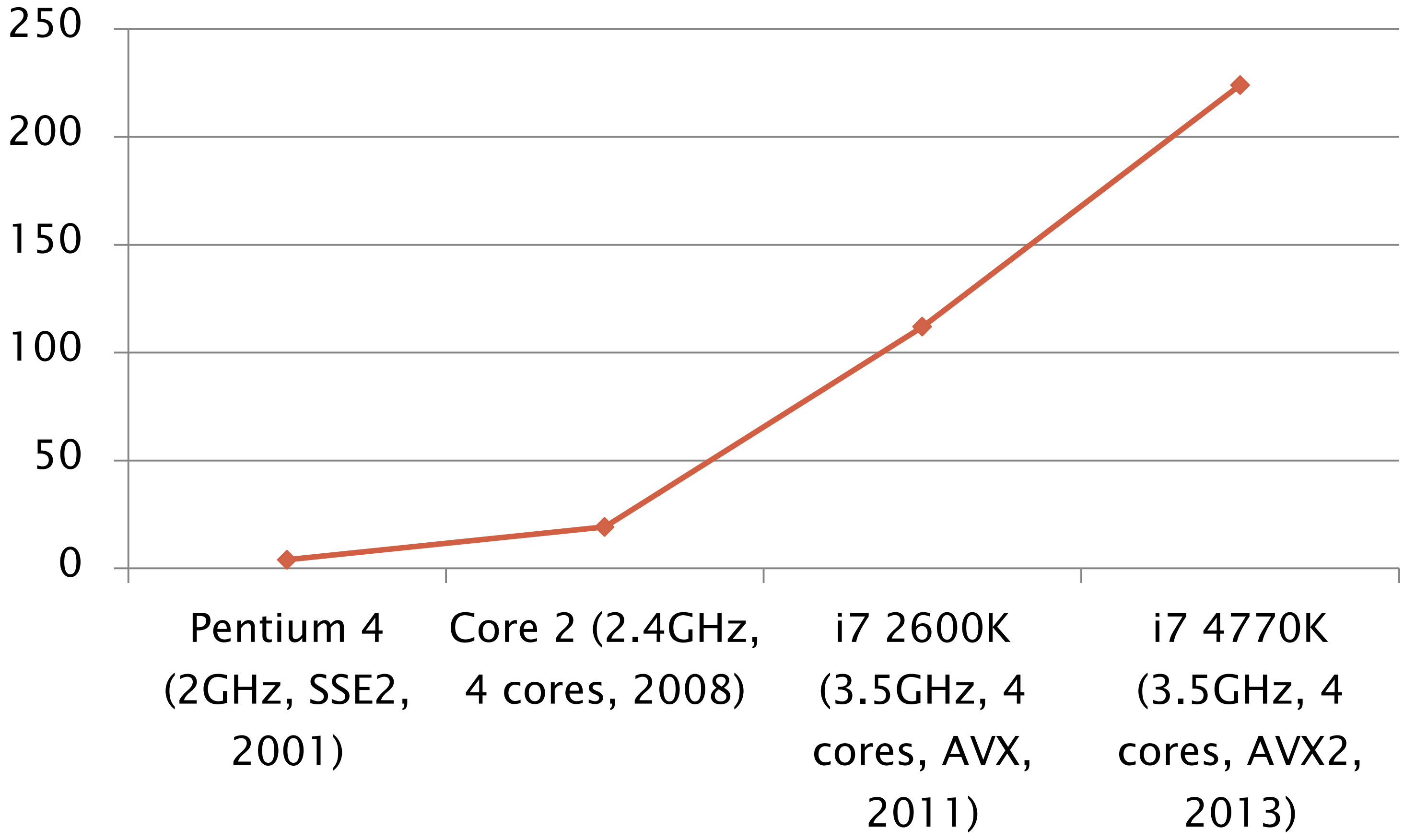
- What if you change the ‘too easy’ threshold…?

	Wins		Bar/Dual
MinTime	Dual	Barrier	GeoMean
>0.01s	541	483	0.95
>0.1s	275	298	0.70
>1s	121	207	0.49

- As models get more difficult, barrier pulls ahead
  - Not on all models, though

Peak Performance

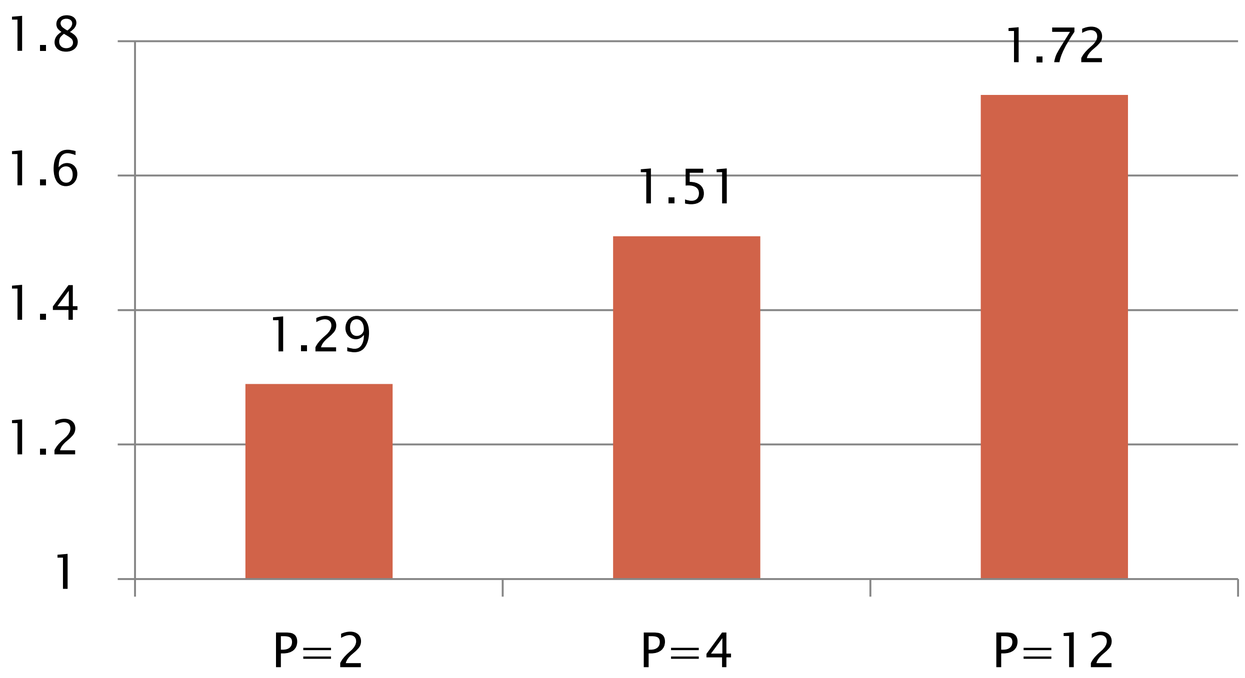
- Peak DP Gflops, from 2001 to today:



GUROBI OPTIMIZATION

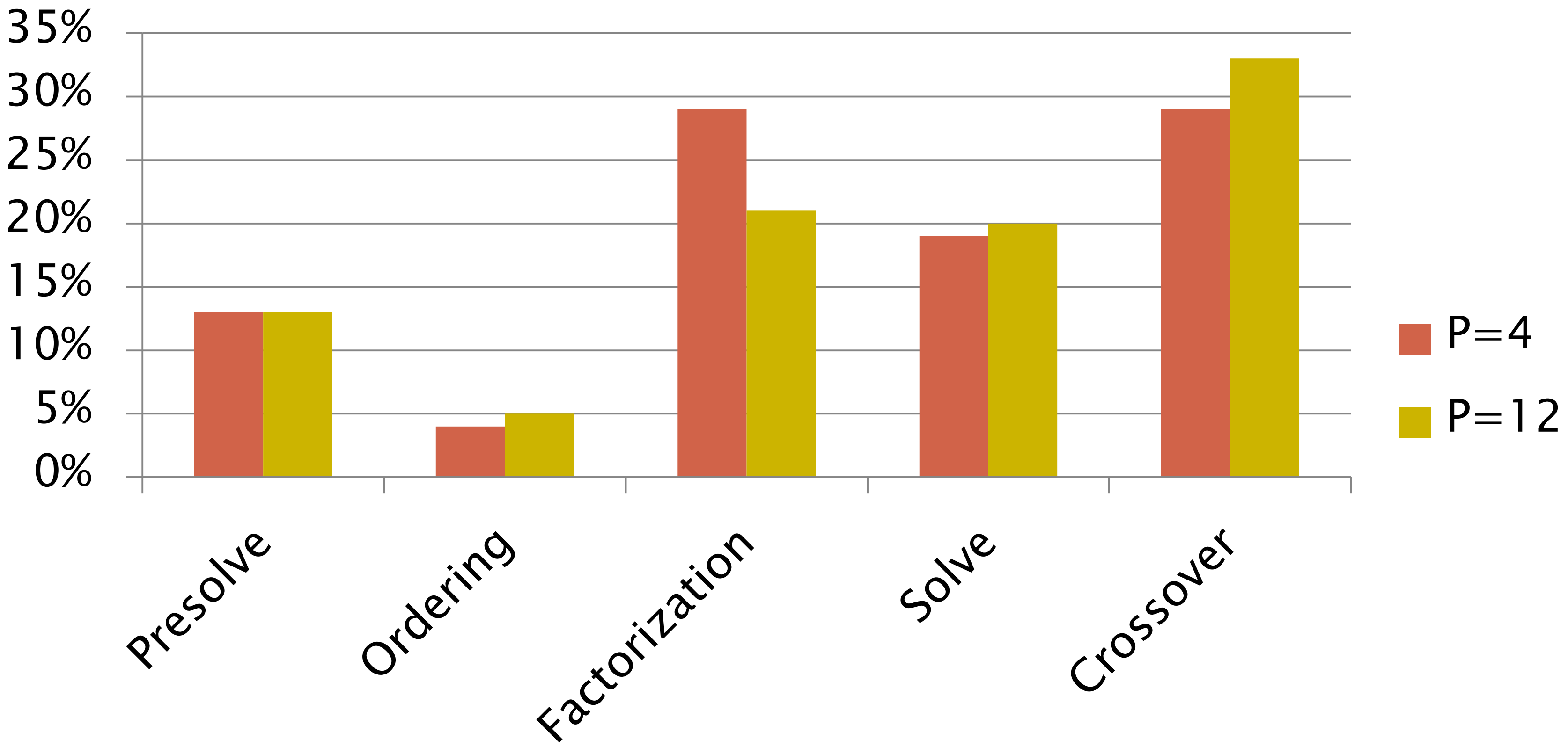
Parallel Barrier Performance

- Parallel speedups
  - Models that take > 1s to solve



Barrier Runtime Breakdown

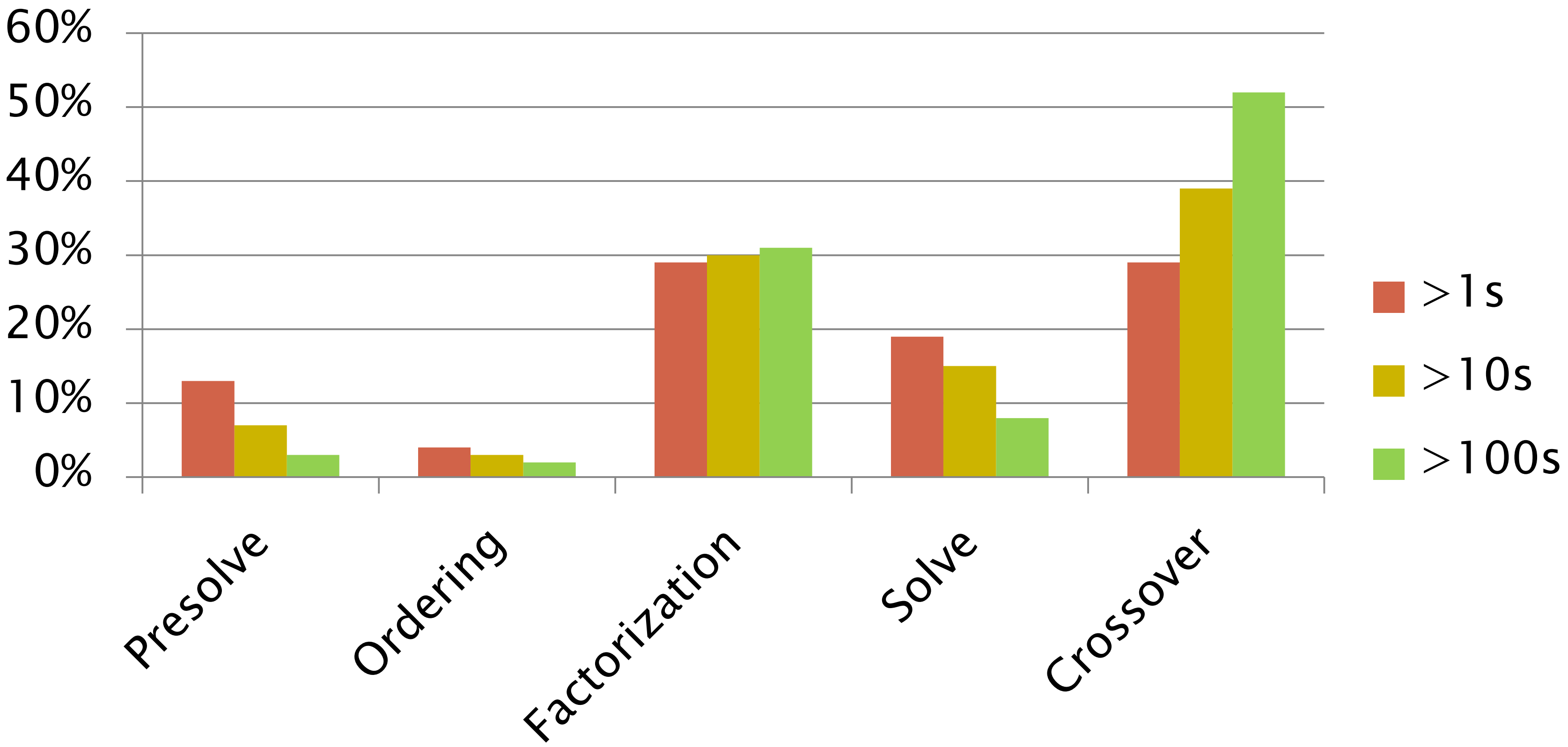
- For models that require more than 1s:



GUROBI OPTIMIZATION

Barrier Runtime Breakdown

- As models get harder (P=4)…



Concurrent Optimization

- Run both algorithms, stop when the first one finishes
- Results:

  Gurobi 5.6, quad-core i7-3770K

  Dual simplex on 1 core, barrier on 3 cores

  Models that take >1s

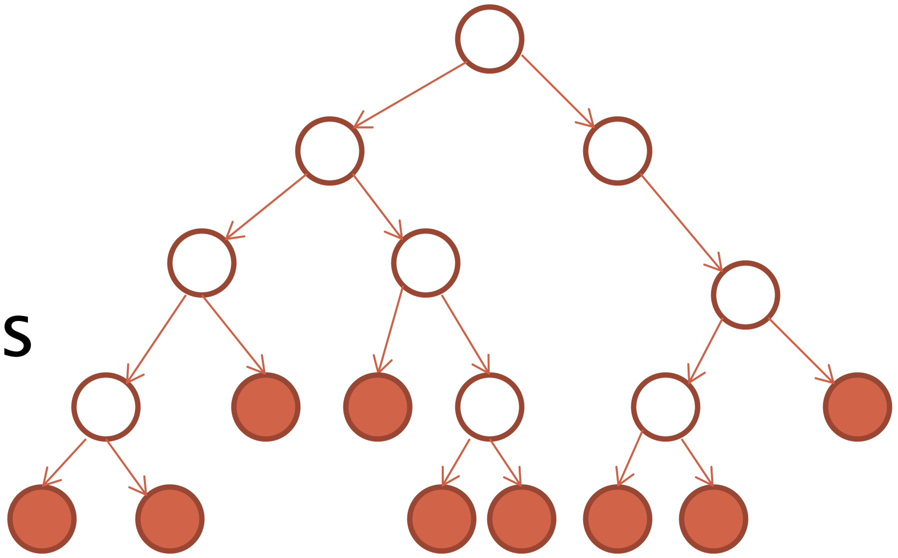
	GeoMean
Dual simplex	1.00
Barrier	0.49
Concurrent	0.38

GUROBI OPTIMIZATION

Parallelism in Mixed-Integer Programming

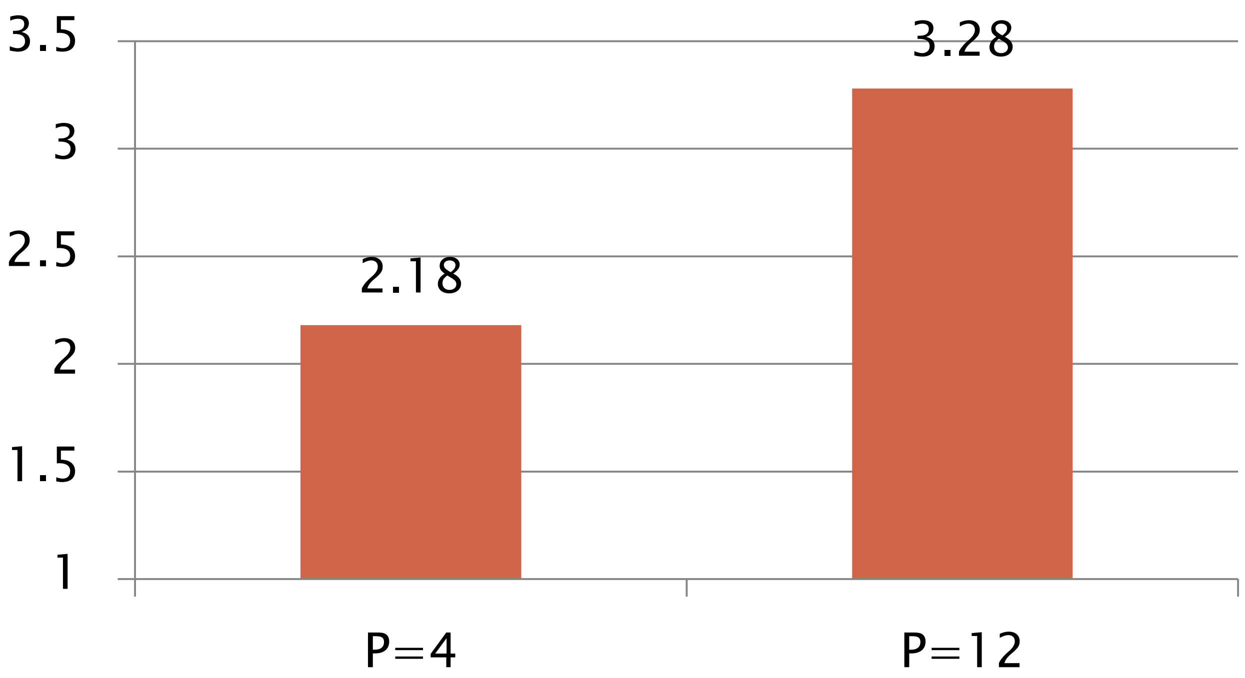
MIP – Embarrassingly Parallel?

- Subtrees in branch-and-bound are independent
- Trivial to distribute them among processors

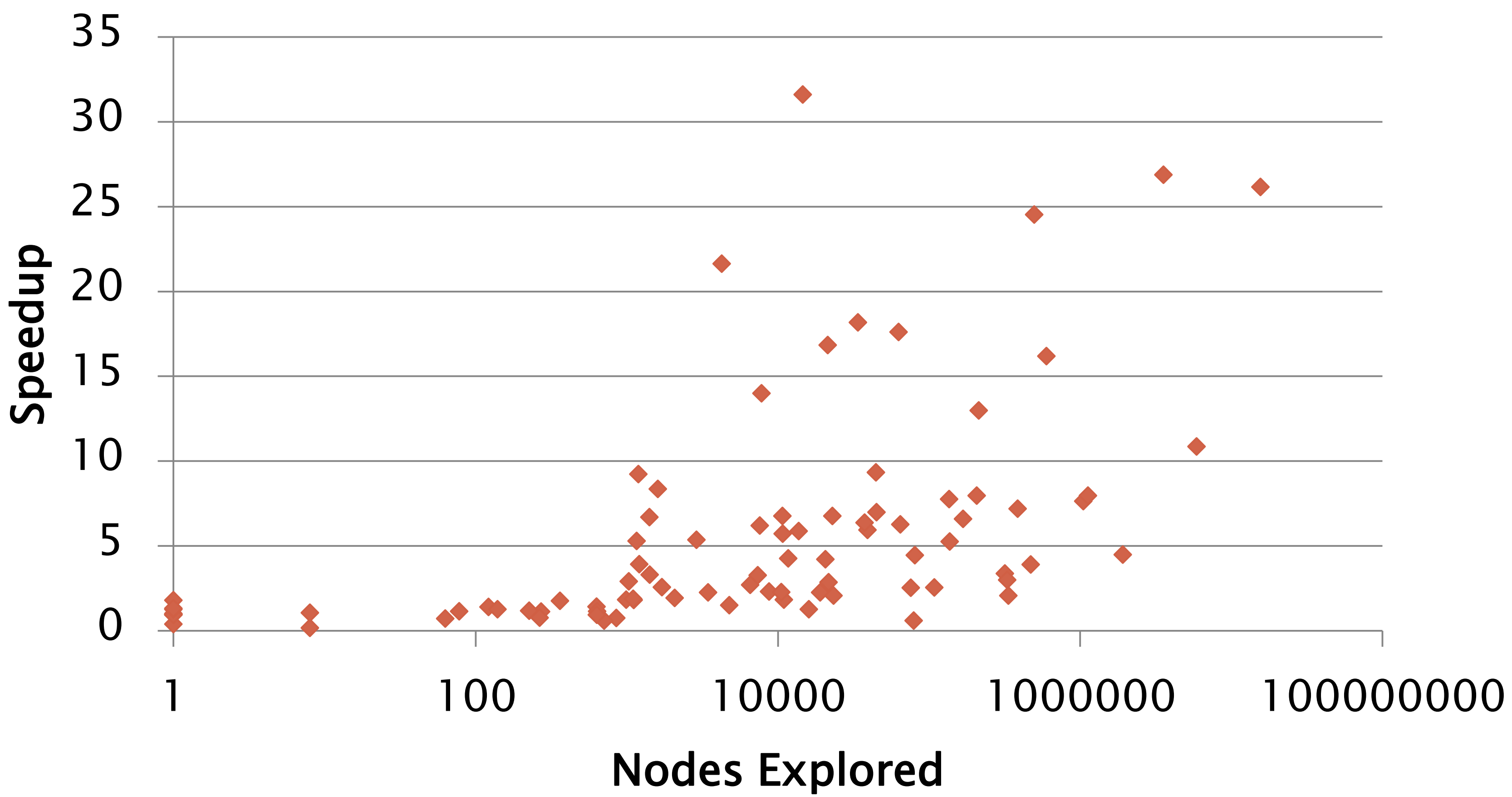


Parallel MIP – Reality

- MIPLIB2010 test set:
  - *Benchmark* subset: 87 models, not too easy, not too hard



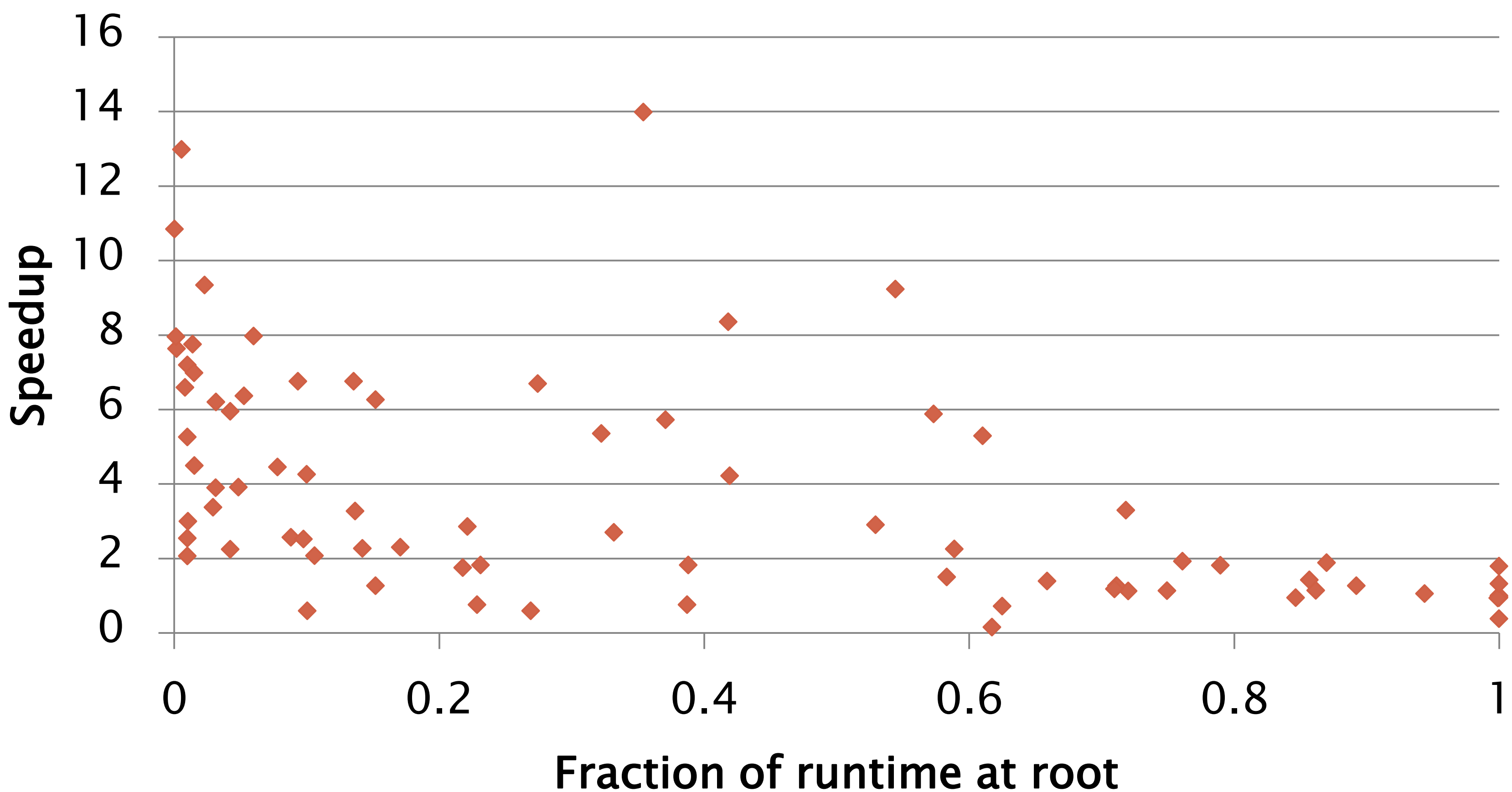
Parallel Speedup By Model (P=12)



A Bit of Noise Mixed In

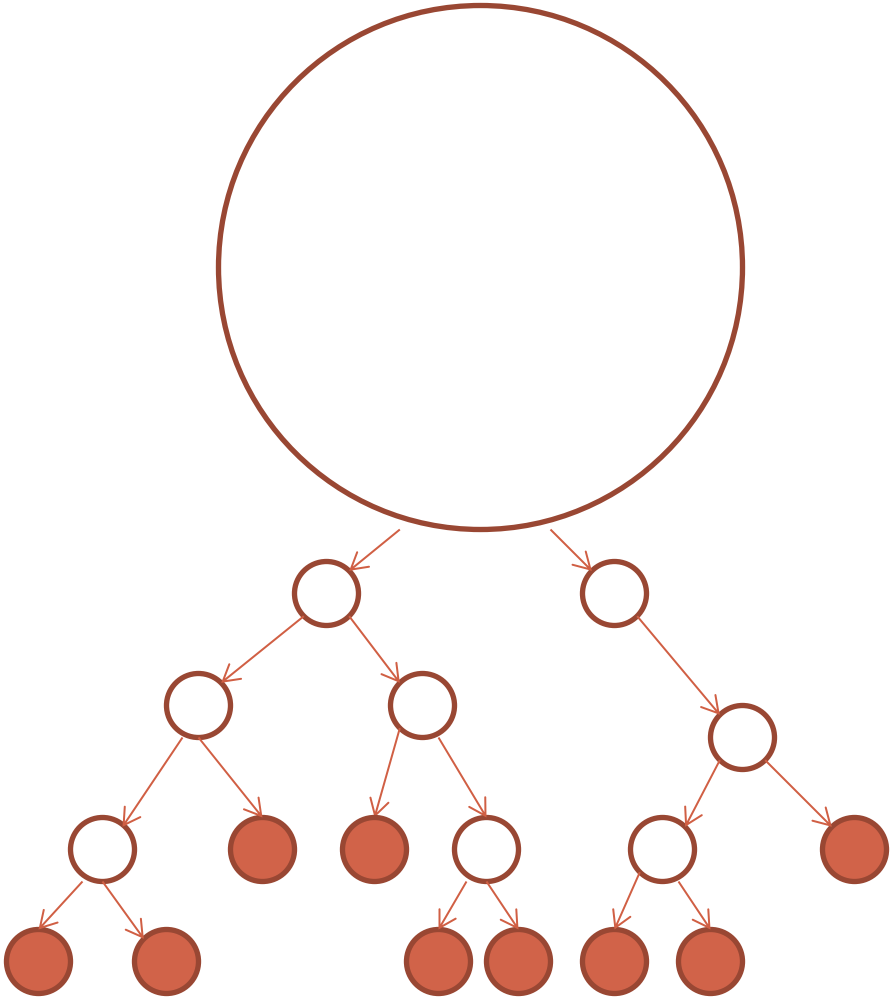
- Random noise plays a big role
- Example – model *60WA01*:
  - Default settings: 509s
  - *Seed=2*: 23s
- 22X speedup from changing the random number seed

Parallel Speedup By Model (P=12)



GUROBI OPTIMIZATION

More Accurate Picture of Search Tree



Root Computations

- What happens at the root node?
  - Presolve
  - Root relaxation solution
  - Cutting planes
  - Heuristics
  - Symmetry detection
  - Initial branch variable selection
  - …
- Basic motivation
  - Better to discover something at the root than rediscover it at every node

GUROBI OPTIMIZATION

Example – Cutting Planes

- Identify constraints that cut off continuous solutions but don’t cut off integer solutions
  - Simple example: clique cut (binary variables)
    - $x + y <= 1$, $y + z <= 1$, $x + z <= 1$
    - Feasible relaxation solution: $x=y=z=0.5$
    - Implied: $x + y + z <= 1$
- Add redundant constraints to the model to tighten the relaxation
  - 13 different cutting plane types in Gurobi

GUROBI OPTIMIZATION

Example – Symmetry

- Identify symmetry in the model
  - Given a MIP
    - min {c'x | Ax <= b}
  - Find all *automorphisms*:
    - Row permutation α
    - Column permutation β
    - $(\beta, \alpha)(A) = A$, $\alpha(c) = c$, $\beta(b) = b$
- During search, prune subtrees that are isomorphic to already explored subtrees

GUROBI OPTIMIZATION

MIP Speedup 2009–Present

- Test environment
  - Internal test set (~6000 models)
  - Solvable by at least one version
  - At least one version takes > 100 seconds
  - Geometric means speedup
  - P=4*
- Version-to-version improvements
  - Gurobi 1.0 -> 2.0: 2.4X
  - Gurobi 2.0 -> 3.0: 2.2X (5.1X)
  - Gurobi 3.0 -> 4.0: 1.3X (6.6X)
  - Gurobi 4.0 -> 5.0: 2.0X (12.8X)
  - Gurobi 5.0 -> 5.5: 1.3X (16.4X)
  - Gurobi 5.5 -> 5.6: 1.3X (20.9X)**

*p=4 vs. p=1 for V5.1 – 1.9X
**Approximately 2x per year

GUROBI OPTIMIZATION

The Nature of the Improvements

- MIP improvements generally reduce the number of nodes explored
  - Speed of processing branch-and-bound nodes hasn't changed much over the years
  - Improvements often increase the time spent at the root node
- Consequence
  - Better MIP algorithms -> fewer opportunities for parallelism

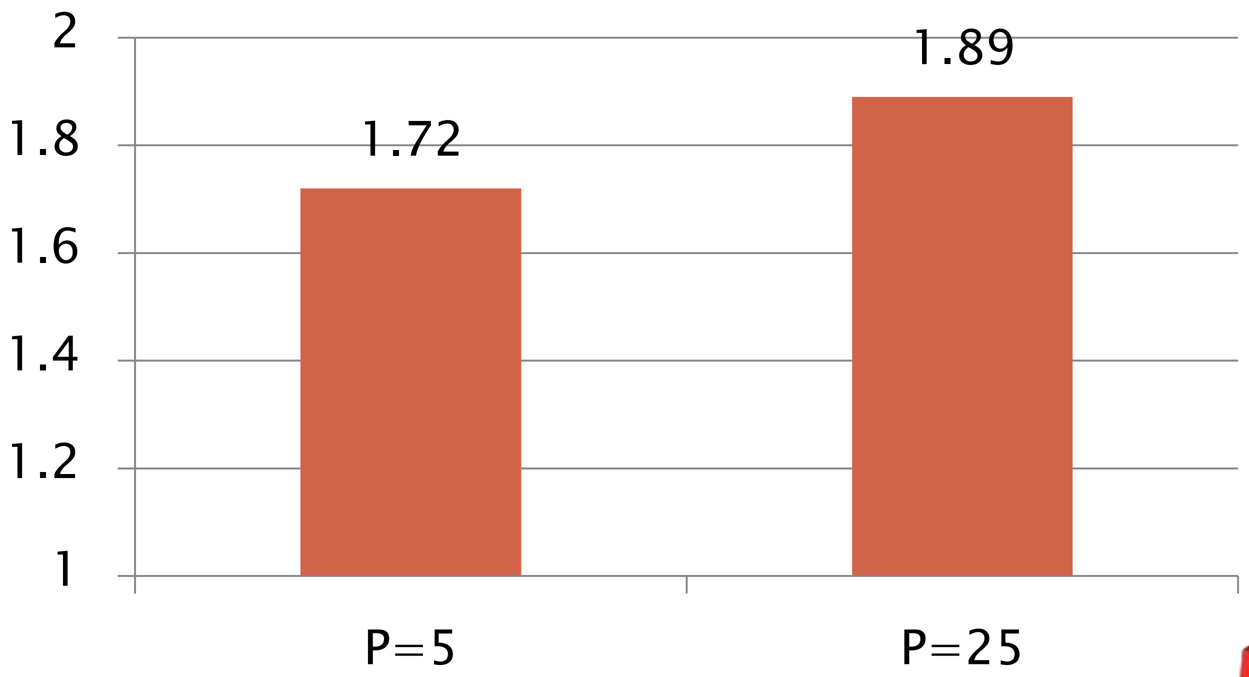
Concurrent MIP

- Same idea as for LP:
  - Apply different algorithms on different processors
  - First one that finishes wins
- For MIP:
  - Consider different strategies rather than different algorithms
    - More/less aggressive cuts
    - More/less aggressive heuristics
    - Different branch variable selection
    - More/less aggressive presolve
  - Most effective strategy we've found so far…
    - Different random number seeds

GUROBI OPTIMIZATION

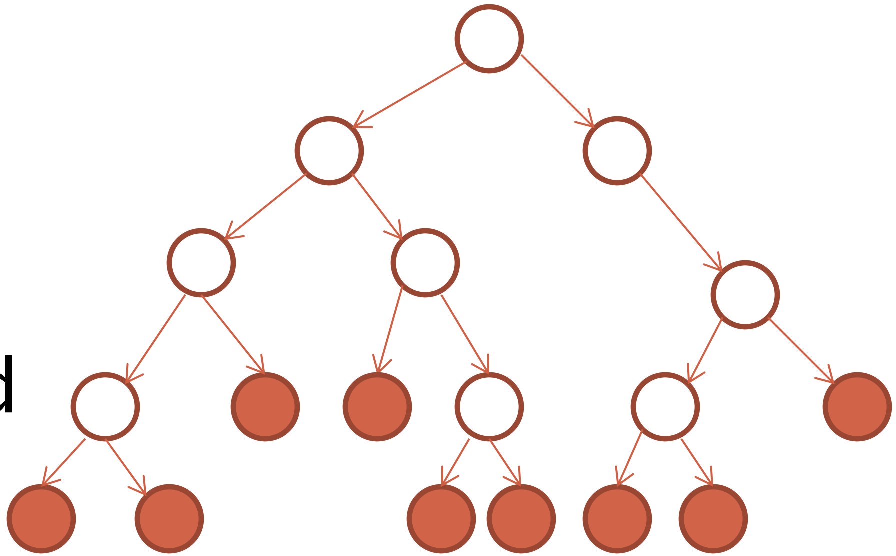
Concurrent MIP

- MIPLIB2010 test set:
  - Models that require >100s
  - Different random number seeds on each instance



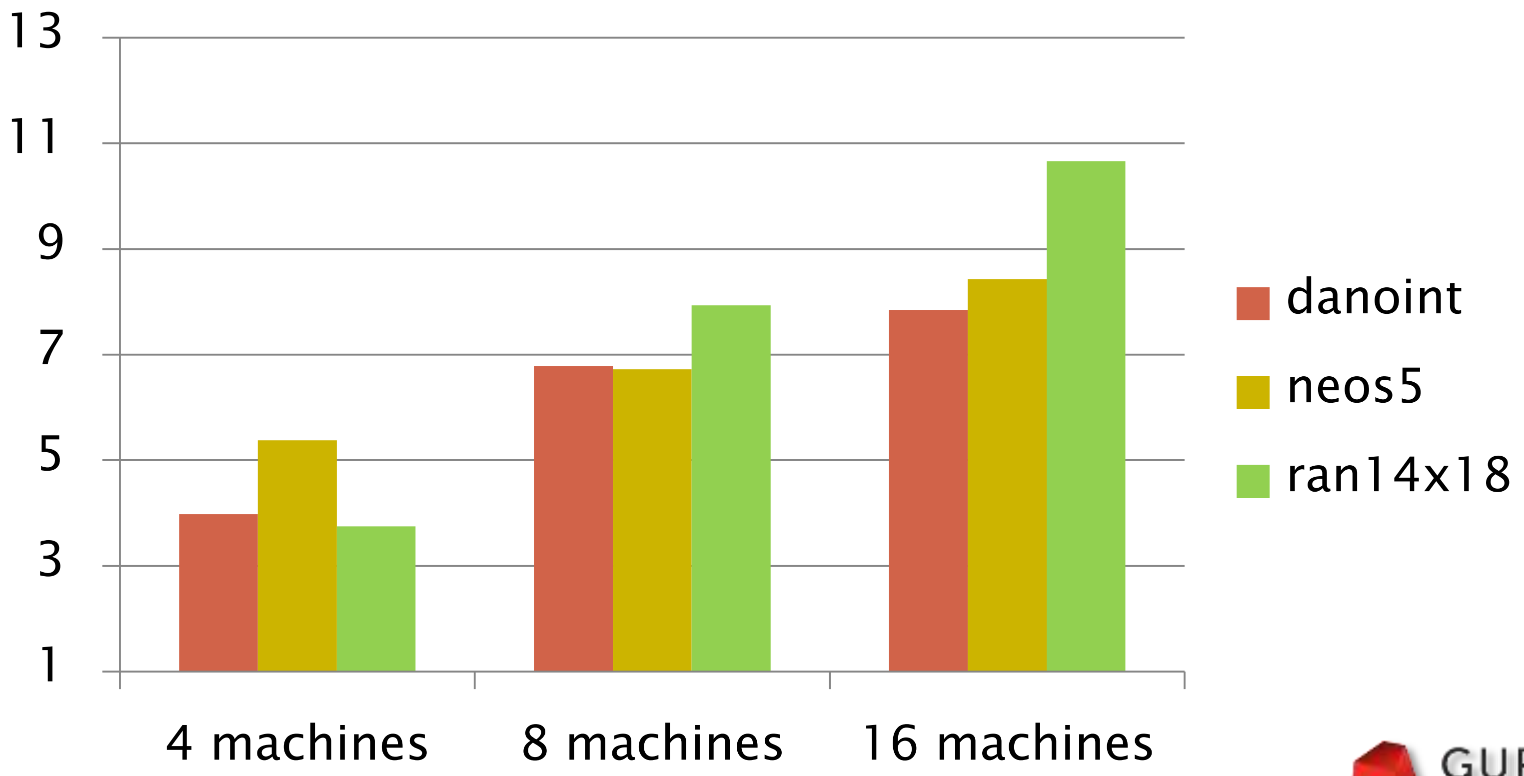
Distributed MIP

- Not all is lost
- Still plenty of models with large search trees
- Simple distributed scheme sometimes works well



Distributed MIP

- Parallel speedups, versus a single machine



Conclusions

- Significant demand for performance
  - The data is there
  - The money is there
- Despite “obvious” sources of parallelism, parallel computing continues to play only a modest role