Parallelism in Linear and Mixed Integer Programming

Ed Rothberg
Problem Statement – LP

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

\[ \text{Minimize} \quad c^T x \]

\[ \text{Subject to} \quad Ax = b \]

\[ l \leq x \leq u \]
Problem Statement – MIP

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

\[
\text{Minimize} \quad c^T x \\
\text{Subject to} \quad Ax = b \\
\quad \quad \quad \quad l \leq x \leq u \\
\text{some or all } x_j \text{ integer}
\]
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

© 2013 Gurobi Optimization
Linear Programming

Simplex solution path

Interior-point central path
- Predictor
- Corrector

Optimum
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

© 2013 Gurobi Optimization
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
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
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.
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, …
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Wins</th>
<th>GeoMean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual simplex</td>
<td>541</td>
<td>1.00</td>
</tr>
<tr>
<td>Barrier</td>
<td>483</td>
<td>0.95</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>MinTime</th>
<th>Wins</th>
<th>Bar/Dual</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;0.01s</td>
<td>541</td>
<td>483</td>
</tr>
<tr>
<td>&gt;0.1s</td>
<td>275</td>
<td>298</td>
</tr>
<tr>
<td>&gt;1s</td>
<td>121</td>
<td>207</td>
</tr>
</tbody>
</table>

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

- Peak DP Gflops, from 2001 to today:

  - Pentium 4 (2GHz, SSE2, 2001)
  - Core 2 (2.4GHz, 4 cores, 2008)
  - i7 2600K (3.5GHz, 4 cores, AVX, 2011)
  - i7 4770K (3.5GHz, 4 cores, AVX2, 2013)
Parallel Barrier Performance

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

![Bar Chart](chart.png)

- P=2: 1.29
- P=4: 1.51
- P=12: 1.72
Barrier Runtime Breakdown

- For models that require more than 1s:
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

<table>
<thead>
<tr>
<th>Method</th>
<th>GeoMean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual simplex</td>
<td>1.00</td>
</tr>
<tr>
<td>Barrier</td>
<td>0.49</td>
</tr>
<tr>
<td>Concurrent</td>
<td>0.38</td>
</tr>
</tbody>
</table>
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)

![Graph showing parallel speedup by model with fraction of runtime at root on the x-axis and speedup on the y-axis.](image-url)
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
Example – Cutting Planes

- Identify constraints that cut off continuous solutions but don’t cut off integer solutions
  - Simple example: clique cut (binary variables)
    - \( x + y \leq 1, y + z \leq 1, x + z \leq 1 \)
    - Feasible relaxation solution: \( x=y=z=0.5 \)
    - Implied: \( x + y + z \leq 1 \)

- Add redundant constraints to the model to tighten the relaxation
  - 13 different cutting plane types in Gurobi
Example – Symmetry

- Identify symmetry in the model
  - Given a MIP
    - min \{c’x \mid Ax \leq b\}
  - Find all \textit{automorphisms}:
    - Row permutation \(\alpha\)
    - Column permutation \(\beta\)
    - \((\beta, \alpha)(A) = A, \alpha(c) = c, \beta(b) = b\)

- During search, prune subtrees that are isomorphic to already explored subtrees
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
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
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

![Graph showing speedup comparison between 4, 8, and 16 machines across different machines and datasets.](image)
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