Solution of Mixed-Integer Programming Problems on the XT5

Rebecca J. Hartman-Baker[*]
Ingrid K. Busch[+]
Michael R. Hilliard[+]
Richard S. Middleton[+]
Michael S. Schultze[+]

Oak Ridge National Laboratory

CUG 2009
May 7, 2009

[*]National Center for Computational Sciences
[+]Center for Transportation Analysis

Outline

- Motivation
- Problem Formulation
  - Linear Programming
  - Mixed-Integer Linear Programming
- Problem Solution
  - Linear Programming
  - Mixed-Integer Linear Programming
  - Parallelization of MILP solvers
- Existing MILP Solvers
- Results
- Future Work

Motivation

- **Mixed-integer linear programs arise in many applications**
  - Logistics
  - Supply-chain analysis
  - Data mining
- **Problems that restrict some solution variables to integer values**
  - E.g., can’t place 0.6 biorefineries

Linear Programming

- **Optimization problem in which objective function and constraints are linear functions of unknowns**

$$\min \quad c^T x \quad \text{subject to} \quad Ax \le b$$

Mixed-Integer Linear Programming

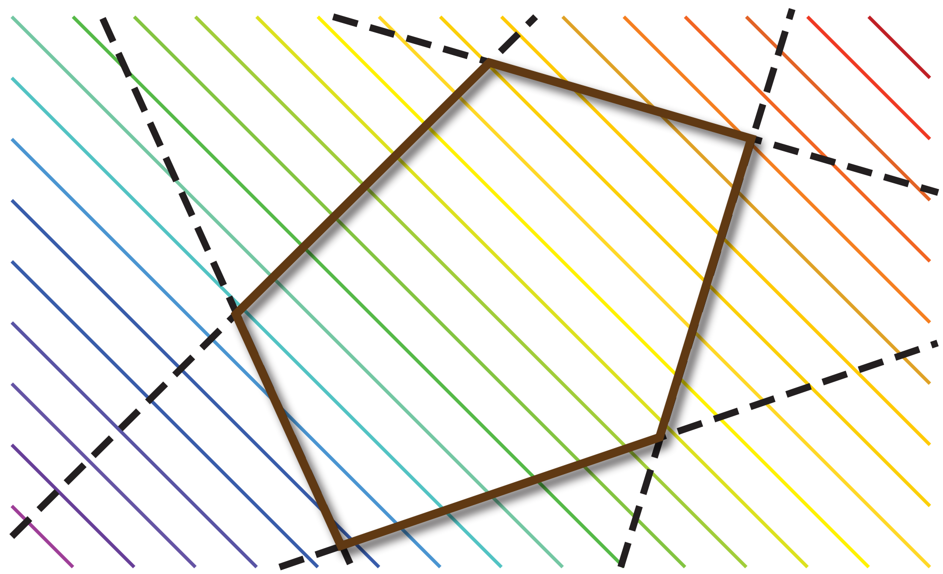
- **Linear program in which some variables restricted to integer values**

$$\min \quad c^T x \quad \text{subject to} \quad Ax \le b, \quad x_j \in Z \; \forall j \in D$$

D = set of indices of integer variables in x

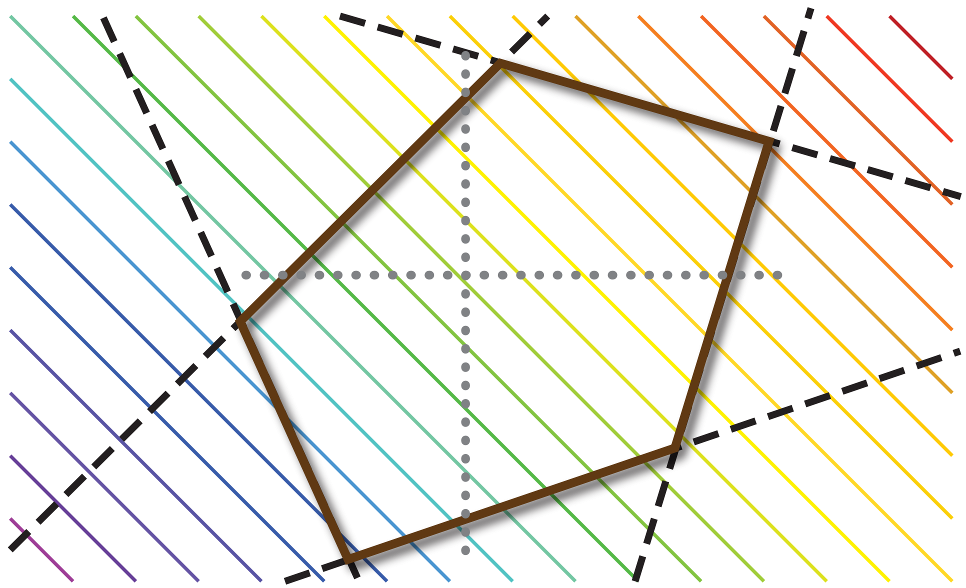
Solving Linear Programs

- Simplex method
  - Iteratively examine all vertices of polytope
  - Worst-case performance (*n* variables, *m* constraints): $\binom{n}{m}$
  - Pathological examples exist but average performance $O(\min\{(m-n)^2, n^2\})$



Solving Mixed-Integer Linear Programs (MILPs)

- MILPs are NP-complete – use heuristics
- Solvers rely on branching concept
- Combine branching with means to compute lower bounds, to develop branch and bound method

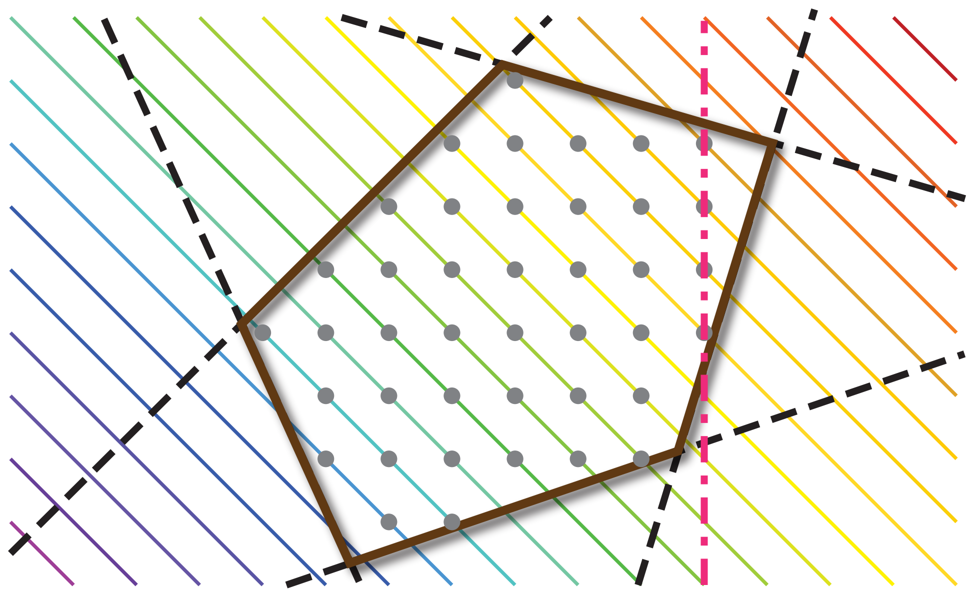


Branch and Bound

- Improve upper and lower bounds by systematic bounding and branching of subproblems
- Use relaxation of MILP to LP as bounding function
- Eliminate nonviable pieces of domain by applying bounds
- Systematically search domain
  - Best-first search (smallest lower bounds first)
  - Depth-first search (deepest branch first)

Branch and Cut

- **Lower bound generated by LP relaxation often too loose for efficient solution**
- **Improve bound by adding valid inequalities to problem as computation proceeds**



Parallelization of MILP Solvers

- Algorithm can be broken into 3 parts
  - Ramp-Up
  - Search and Process
  - Ramp-Down

Ramp-Up Phase

- Break feasible region into enough pieces for all processes to share
- Uninvolved processors are idle
- Can occupy uninvolved processors with auxiliary tasks such as preprocessing and computation of upper bounds, but may not be worth time
- Ideally, shorten ramp-up phase
- No good ramp-up acceleration techniques developed

Search and Process Phase

- Search and process branch nodes, and use and share info (knowledge management)
- Two approaches
  - Centralized control: manager-worker paradigm, not scalable, but clearer global picture of problem
  - Decentralized control: more scalable, local hubs serve as local centralized control, may perform more work
- Search strategy must be considered in terms of global and local scope

Ramp-Down Phase

- Symmetric to ramp-up phase
- Face similar issues
- Generally ignored but can lead to serious scalability issues

Existing MILP Solvers: Commercial

- CPLEX
  - Leading commercial package
  - Shared-memory parallel version available
  - ParaLEX, distributed memory version, developed by researchers, and shows limited promise
- Gurobi
  - Just released (Spring 2009)
  - Parallelizes across multicore processors

Existing MILP Solvers: Open Source

- **COIN-OR**
  - Repository of operations research-related open source software
  - Repository includes SYMPHONY and CHiPPS
  - CHiPPS contains BLIS, distributed parallel MILP solver
- **PICO**
  - Distributed parallel package
  - Capable of scaling to thousands of processors

BLIS Parallelization Strategy

- Decentralized approach
- Global list of candidate nodes spread across all processes
- Every process selects nodes from local pool
- Load balancing uses 3-level master/hub/ worker paradigm

PICO Parallelization Strategy

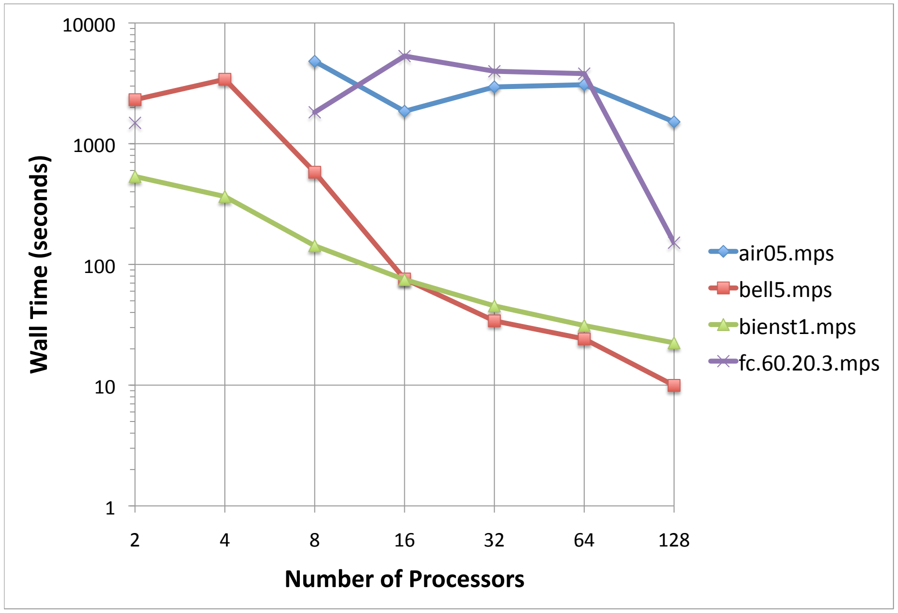
- Hybrid knowledge management scheme, similar to BLIS
- Idle processes given work from overburdened processes by hub processes
- Communication occurs between hub and its workers and between hubs

Results

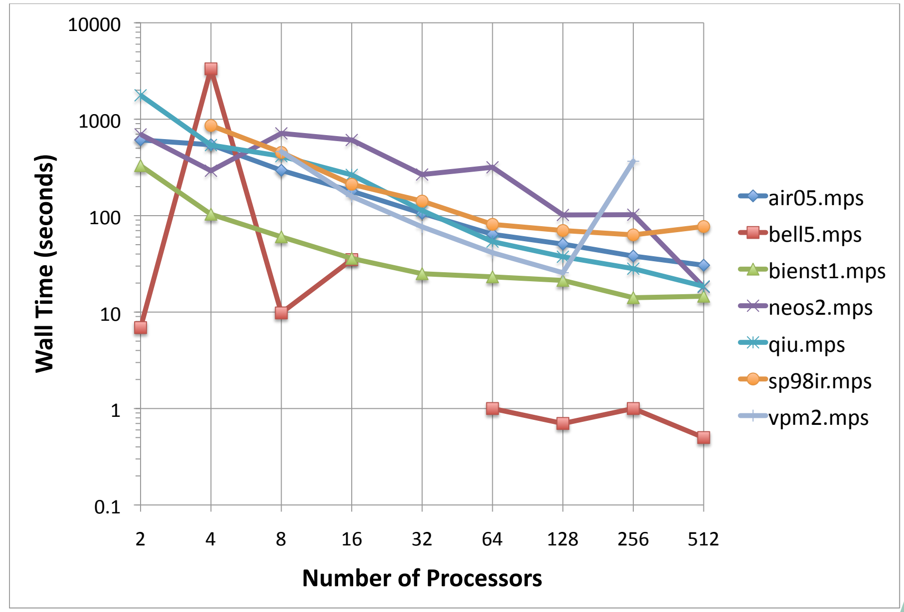
- **Two sets of test problems run on Jaguar Cray XT5 at ORNL**
- **Standard Test Problems**
  - **Problems of varying sizes that came with BLIS distribution**
  - **100-7200 unknowns, 90-1600 constraints**
- **Canadian Cities Problems**
  - **Placement of facilities in Canadian cities**
  - **26,000-2.4 million unknowns and constraints**

Results: Standard Test Problems

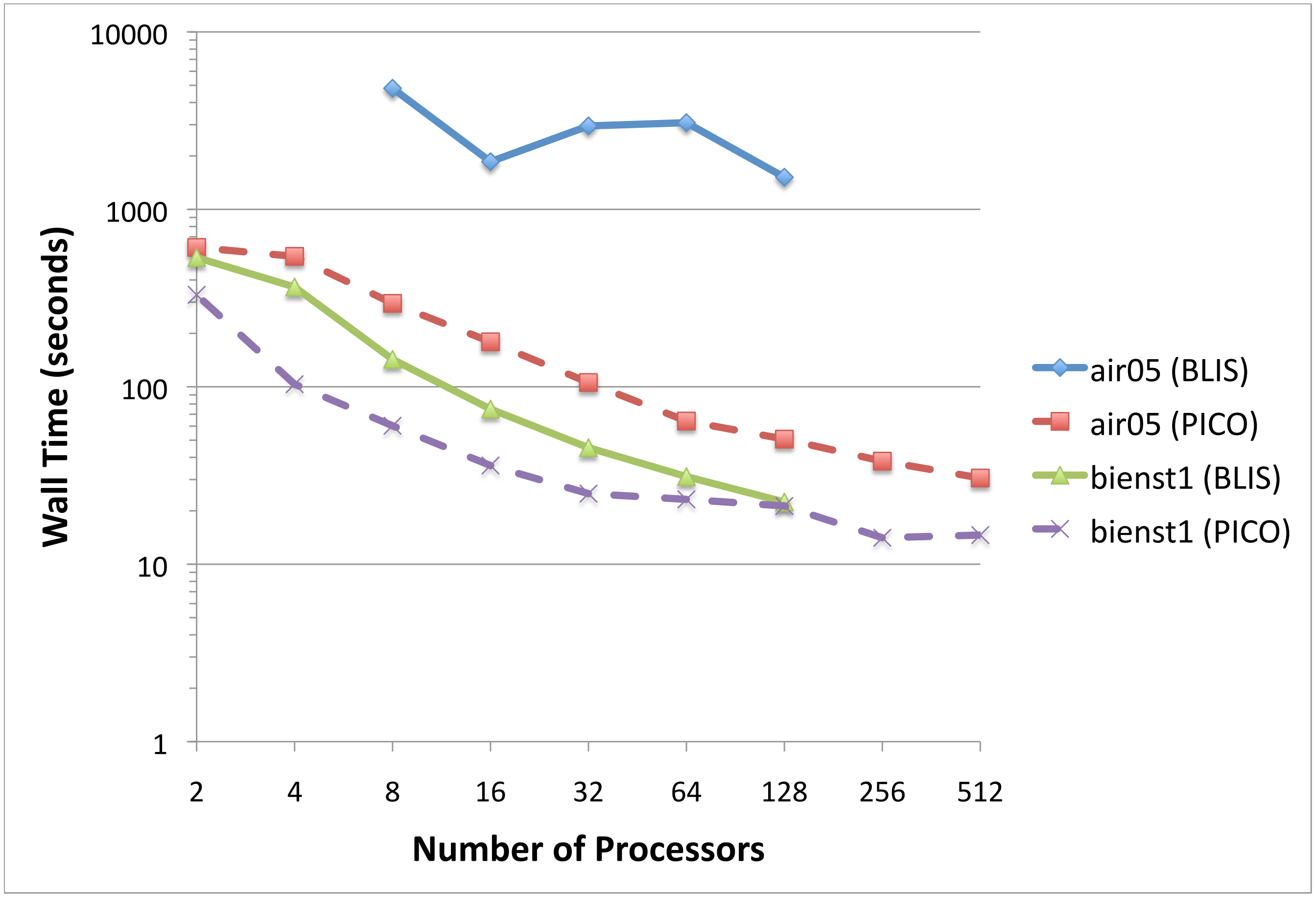
BLIS



PICO

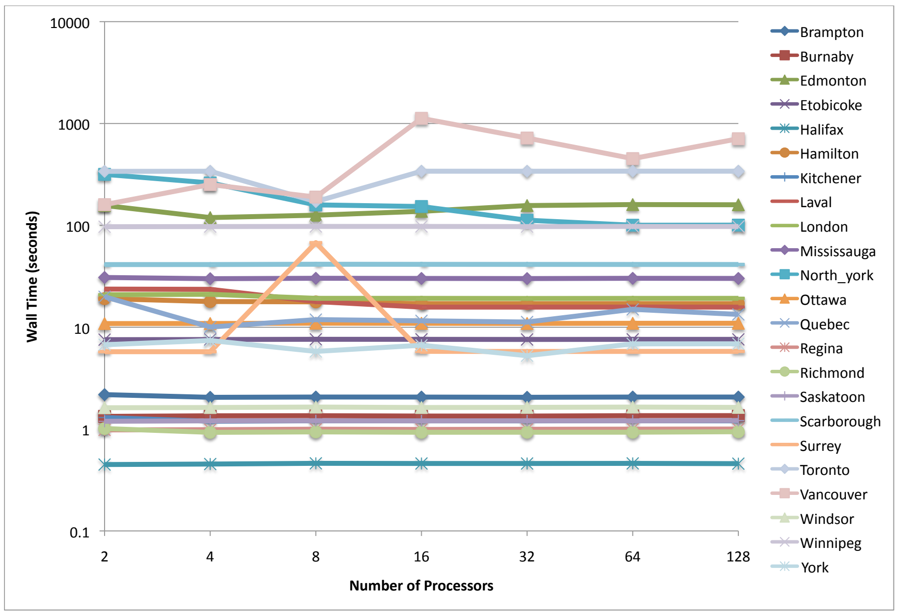


Results: Standard Test Problems

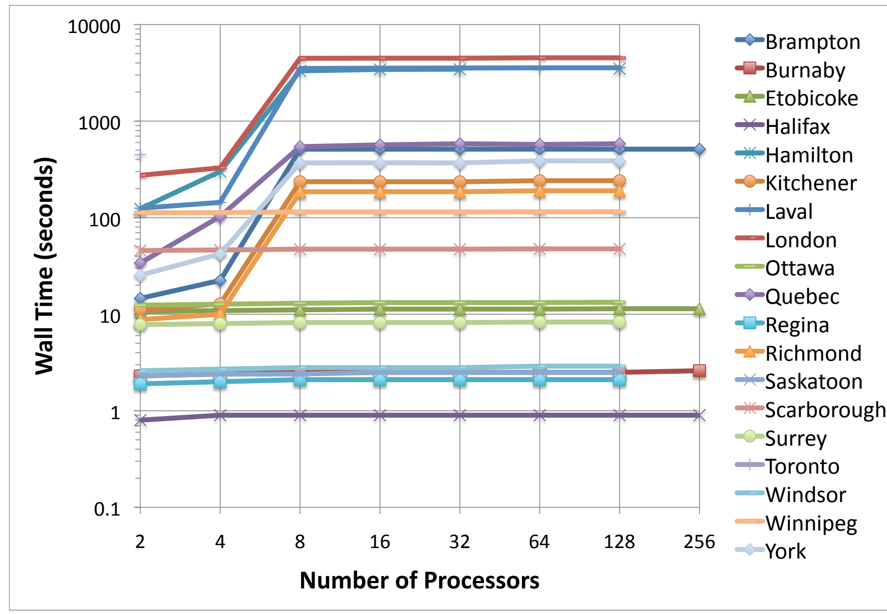


Results: Canadian Cities

BLIS



PICO



Results: Analysis

- Some scalability for standard test problems
- No scaling for Canadian cities problems
  - Majority of compute time spent in ramp-up phase
  - No room to scale!
- Both PICO and BLIS had trouble solving some problems
  - Segfaults, OOM, hangs
  - Insufficiently robust for petascale
- In principle, parallel MILP solvers show promise; in practice, need improvement

Future Work

- **More code development needed**
- **One idea: use existing parallel framework to parallelize MILP solvers**
  - **MADNESS (Multiscale Adaptive Numerical Environment for Scientific Simulation) good candidate**
  - **Use MADNESS to distribute branches across processes, paired with well-established branching and bounding/cutting methods**

Acknowledgments

- This research was sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U.S. Department of Energy.
- This research used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Resources

- **BLIS**

  Available for download from http://www.coin-or.org/

- **PICO**

  Jonathan Eckstein, Cynthia A. Phillips and William E. Hart, "PICO: An object-oriented framework for parallel branch-and-bound," in *Proc Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications*, Elsevier Scientific Series on Studies in Computational Mathematics, pp. 219 -- 265, 2001.

Resources

- **Books and articles about MILP**

  Steven G. Nash and Ariela Sofer, *Linear and Non-Linear Programming*, New York: McGraw-Hill, 1996.

  T. Ralphs, ``Parallel Branch and Cut,'' in *Parallel Combinatorial Optimization*, Wiley, 2006.

Questions?