Highly-scalable branch and bound for maximum monomial agreement

Jonathan Eckstein (Rutgers)
William Hart
Cynthia A. Phillips
Sandia National Laboratories

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Classification: Distinguish 2 Classes

- M vectors v_k, each with N binary features/attributes: x_i for i = 1… N
- Each vector can have a weight w_i
- Each vector is a positive or negative example:

$$\Omega^+ \cup \Omega^- = \{1,\ldots,M\} \text{ and } \Omega^+ \cap \Omega^- = \varnothing$$

Feature→	x_1	x_2	x_3	x_4	class	w_i
v_1	0	0	1	1	+	1.0
v_2	1	0	0	1	-	2.0
v_3	1	0	1	1	-	3.0
v_4	0	1	1	0	+	4.0
v_5	1	0	0	0	+	5.0

Observation ↑

Matrix A

Binary Monomial

- A binary monomial is a conjunction of binary features: $x_1 \wedge \neg x_2 \wedge x_5$
- It is equivalent to a binary function:
  - Let J be the set of literals that appear (uncomplemented)
  - Let C be the set of literals that appear complemented

$$m_{J,C}(x) = \prod_{j \in J} x_j \prod_{c \in C} (1 - x_c)$$

- A binary monomial covers a vector if $m_{J,C}(x) = 1$.
  - The vector agrees with the monomial on each selected feature

$$Cover(J,C) = \left\{ i \in \{1,\ldots,M\} \middle| m_{J,C}(A_i) = 1 \right\}$$

Example: Coverage

- Uncomplemented variables J = {1}
- Complemented variables C = {2}
- Cover(J,C) = {2,3,5}

$x_1 = 1$ and $x_2 = 0$

Feature→	x_1	x_2	x_3	x_4	class	w_i
~~v_1~~	~~0~~	~~0~~	~~1~~	~~1~~	~~+~~	~~1.0~~
v_2	1	0	0	1	-	2.0
v_3	1	0	1	1	-	3.0
~~v_4~~	~~0~~	~~1~~	~~1~~	~~0~~	~~+~~	~~4.0~~
v_5	1	0	0	0	+	5.0

Maximum Monomial Agreement

- Maximize$_{JC}$: $f(J,C) = \left| w\left(Cover(J,C) \cap \Omega^{+}\right) - w\left(Cover(J,C) \cap \Omega^{-}\right) \right|$
  - Weighted difference between covered + and - examples

$x_1 = 1$ and $x_2 = 0$

$$f(\{1\},\{2\}) = 5 - 3 - 2 = 0$$

Feature →	x_1	x_2	x_3	x_4	class	w_i
~~v_1~~	~~0~~	~~0~~	~~1~~	~~1~~	~~+~~	~~1.0~~
v_2	1	0	0	1	-	2.0
v_3	1	0	1	1	-	3.0
~~v_4~~	~~0~~	~~1~~	~~1~~	~~0~~	~~+~~	~~4.0~~
v_5	1	0	0	0	+	5.0

LPBoost

- Use MMA as a weak learner
- Use a linear program to find optimal linear combination of the weak learners (column generation)
  - Optimize for gap/separation
- Dual of the LP gives weights for next MMA round
  - More weight to the harder parts
- We want to solve MMA exactly (Goldberg,Shan, 2007)

$$
\begin{aligned}
&\max \rho - D\sum_{i=1}^{M}\xi_i \\
&s.t. \qquad y_i H_i \lambda + \xi_i \geq \rho \qquad \text{for } i = 1,\ldots,M \\
&\qquad\quad\ \sum_{u=1}^{U}\lambda_u = 1 \\
&\qquad\quad\ \xi_i, \lambda_u \geq 0
\end{aligned}
$$

Sandia National Laboratories

Branch and Bound

Branch and Bound is an **intelligent** (enumerative) **search** procedure for discrete optimization problems.

$$\max_{x \in X} f(x)$$

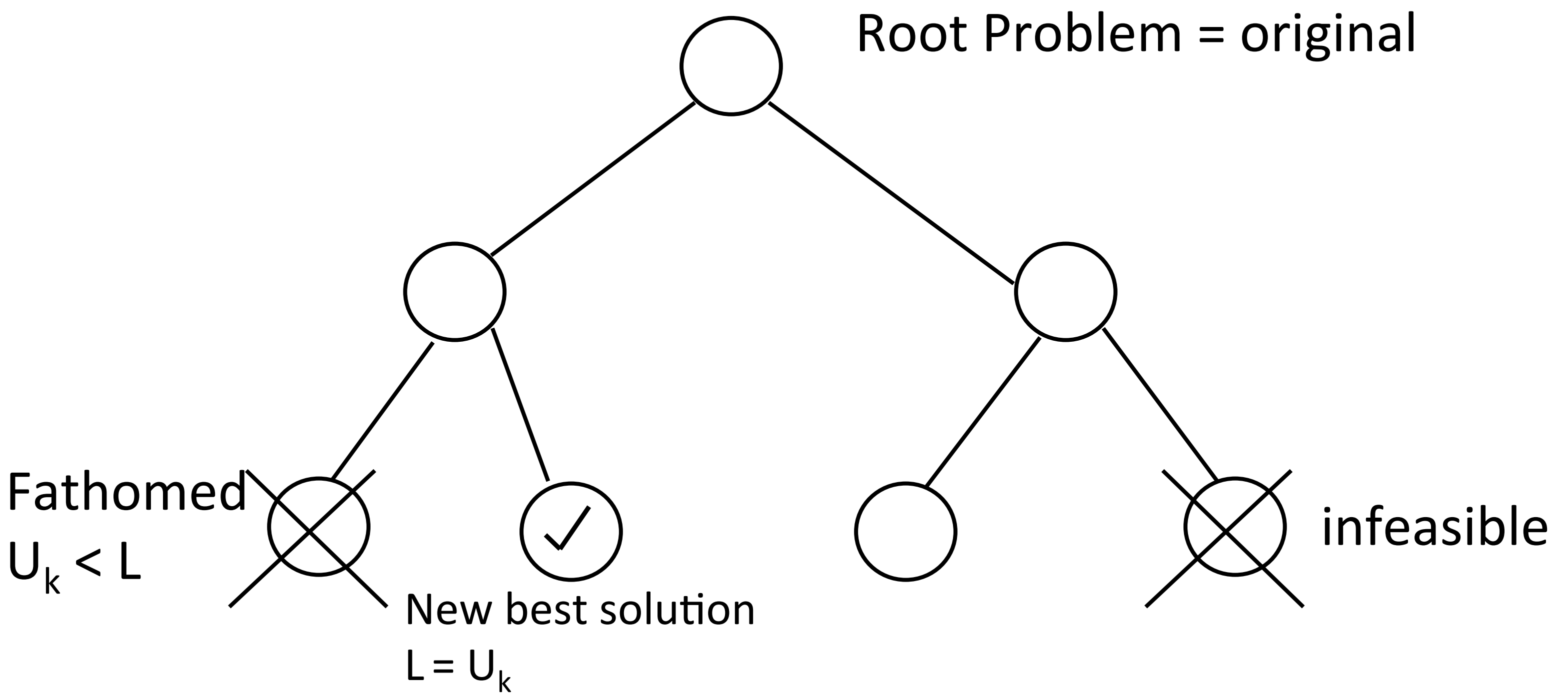
Requires subproblem representation and 3 (problem-specific) procedures:

- Compute an upper bound b(X)

$$\forall x \in X,\ b(x) \geq f(x) \qquad \forall x \in X$$

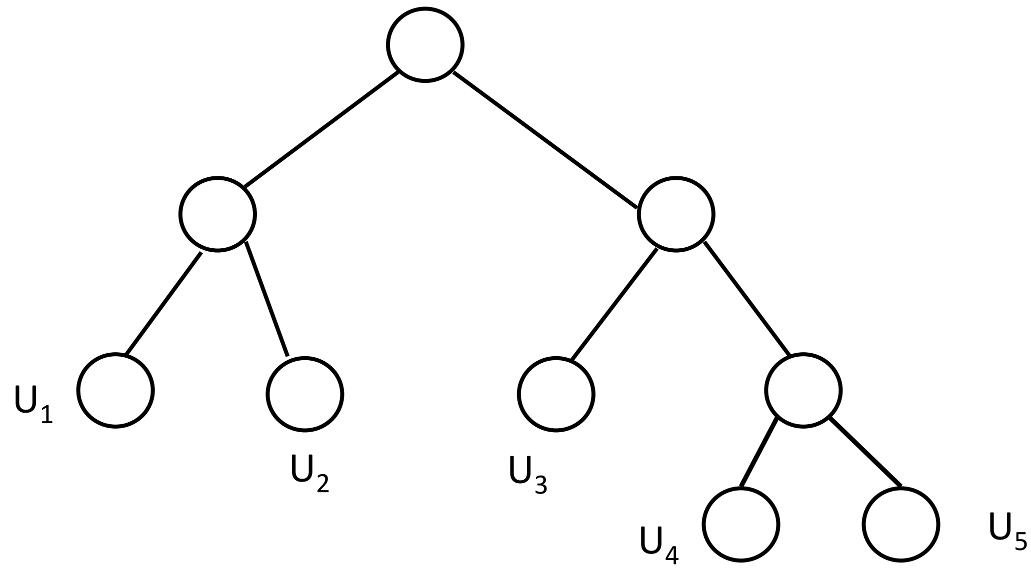
- **Find a candidate solution**
  - Can fail
  - Require that it **recognizes feasibility** if X has only one point
- Split a feasible region (e.g. over parameter/decision space)
  - e.g. Add a constraint

Branch and Bound



- Recursively divide feasible region, prune search when no optimal solution can be in the region.
- Important: need good bounds

Solution Quality



- Global upper bound (maximum over all active problems): $U=\max_k U_k$
- Approximation ratio for current incumbent L is L/U.
- Can stop when L/U is "good enough" (e.g. 95%)
- Running to completion proves optimality

B&B Representation for MMA

- Subproblem (partial solution) = (J,C,E,F)
  - J are features forced into monomial
  - C are features forced in as complemented
  - E are eliminated features: cannot appear
  - F are free features
- Any partition of {1, …, N} is possible
- A feasible solution that respects (J,C,E,F) is just (J,C)
- When F is empty, only one element (leaf)

Serial MMA branch-and-bound elements from Eckstein and Goldberg, "An improved method for maximum monomial agreement," INFORMS J. Computing, 24(2), 2012.

Upper Bound

- Valid: $\max\left\{w\left(Cover(J,C)\cap\Omega^{+}\right),w\left(Cover(J,C)\cap\Omega^{-}\right)\right\}$
- Strengthen by considering excluded features E
- Two vectors inseparable if they agree on all features $i \notin E$
  - Creates q(E) equivalence classes

	x_1	**x_2**	**x_3**	**x_4**
v_1	0	0	1	1
v_2	1	0	0	1
v_3	1	0	1	1
v_4	0	1	1	0
v_5	1	0	0	0

→

E

	x_1	**x_3**	**x_2**	**x_4**
v_1	0	1	0	1
v_4	0	1	1	0
v_2	1	0	0	1
v_5	1	0	0	0
v_3	1	1	0	1

Upper Bound

- V_η^E are vectors in the η^{th} equivalence class
  - All covered or all not covered

$$w_\eta^+(J,C,E) = w\left(V_\eta^E \cap Cover(J,C) \cap \Omega^+\right)$$

$$w_\eta^-(J,C,E) = w\left(V_\eta^E \cap Cover(J,C) \cap \Omega^-\right)$$

- Stronger upper bound:

$$b(J,C,E) = \max\begin{cases} \sum_{\eta=1}^{q(E)} \left(w_\eta^+(J,C,E) - w_\eta^-(J,C,E)\right)_+ \\ \sum_{\eta=1}^{q(E)} \left(w_\eta^-(J,C,E) - w_\eta^+(J,C,E)\right)_+ \end{cases}$$

E

	x_1	x_3	x_2	x_4
v_1	0	1	0	1
v_4	0	1	1	0
v_2	1	0	0	1
v_5	1	0	0	0
v_3	1	1	0	1

Upper Bound

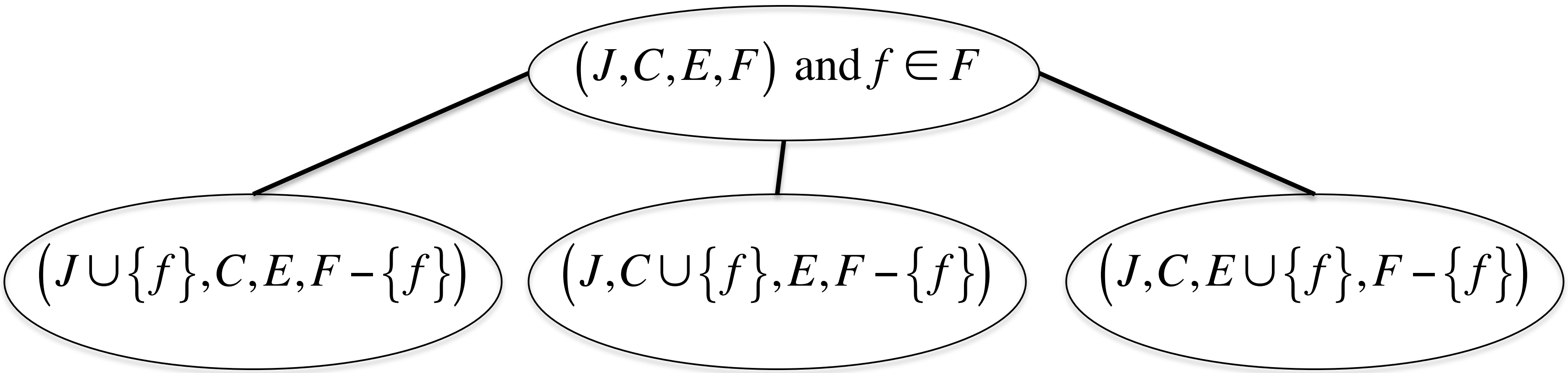
- More convenient form:

$$b(J,C) = \max\left\{w\left(Cover(J,C)\cap\Omega^{+}\right), w\left(Cover(J,C)\cap\Omega^{-}\right)\right\}$$

$$b(J,C,E) = b(J,C) - \sum_{\eta=1}^{q(E)} \min\left\{w_{\eta}^{+}(J,C,E), w_{\eta}^{-}(J,C,E)\right\}$$

- Compute b(J,C) first
  - $|J \cup C|$ set intersections
- If can't fathom, compute second part
  - Compute equivalence classes with radix sort on non-E features

Branching



- Eckstein, Goldberg considered higher branching factor
  – Branching on 1 feature faster, more nodes

Choose branch variable

$$
\begin{tikzcd}
& \left(J,C,E,F\right) \text{ and } f \in F \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & \\
\left(J \cup \{f\}, C, E, F - \{f\}\right) & \left(J, C \cup \{f\}, E, F - \{f\}\right) & \left(J, C, E \cup \{f\}, F - \{f\}\right)
\end{tikzcd}
$$

- Strong branching: for all f
  - Compute all 3 upper bounds, (b_1, b_2, b_3) sorted descending
  - Sort lexicographically, pick smallest. Gives lookahead bound

$$
b\left(J,C,E,F\right) = \min_{f \in F} \left\{ \max \left\{ \begin{array}{l} b\left(J \cup \{f\}, C, E, F - \{f\}\right) \\ b\left(J, C \cup \{f\}, E, F - \{f\}\right) \\ b\left(J, C, E \cup \{f\}, F - \{f\}\right) \end{array} \right. \right.
$$

PEBBL

Parallel Enumeration and Branch-and-Bound Library

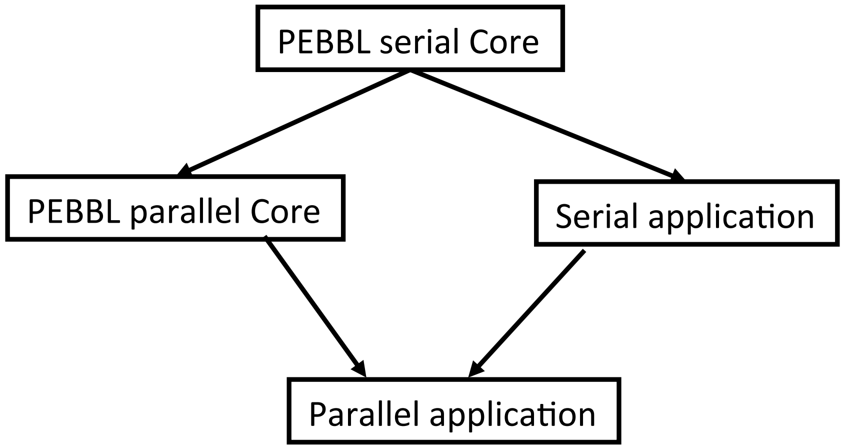
- Distributed memory (MPI), C++

Goals:

- Massively parallel (scalable)
- General parallel Branch & Bound environment
- Parallel search engine cleanly separated from application and platform
- Portable
- Flexible
- Integrate approximation techniques

There are other parallel B&B frameworks: PUBB, Bob, PPBB-Lib, Symphony, BCP, CHiPPS/ALPS, FTH-B&B, and codes for MIP

Pebbl's Parallelism (Almost) Free



User must

- Define serial application (debug in serial)
- Describe how to pack/unpack data (using a generic packing tool)

C++ inheritance gives parallel management

User may add threads to

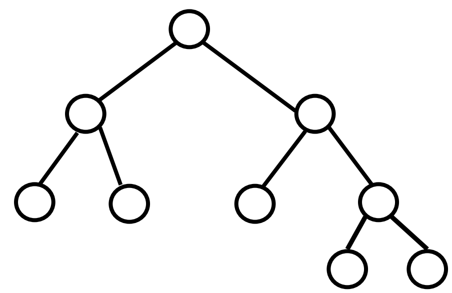
- Share global data
- Exploit problem-specific parallelism
- Add parallel heuristics

PEBBL Features for Efficient Parallel B&B

- Efficient processor use during ramp-up (beginning)
- Integration of heuristics to generate good solutions early
- Worker/hub hierarchy
- Efficient work storage/distribution
- Control of task granularity
- Load balancing
- Non-preemptive proportional-share “thread” scheduler
- Correct termination
- Early output
- Checkpointing

PEBBL Ramp-up

- Tree starts with one node. What to do with 10,000 processors?
- Serialize tree growth
  - All processors work in parallel on a single node
- Parallelize
  - Preprocessing
  - Tough root bounds
  - Incumbent Heuristics
  - Splitting decisions (MMA)
    - Strong-branching for variable selection



PEBBL Ramp-up

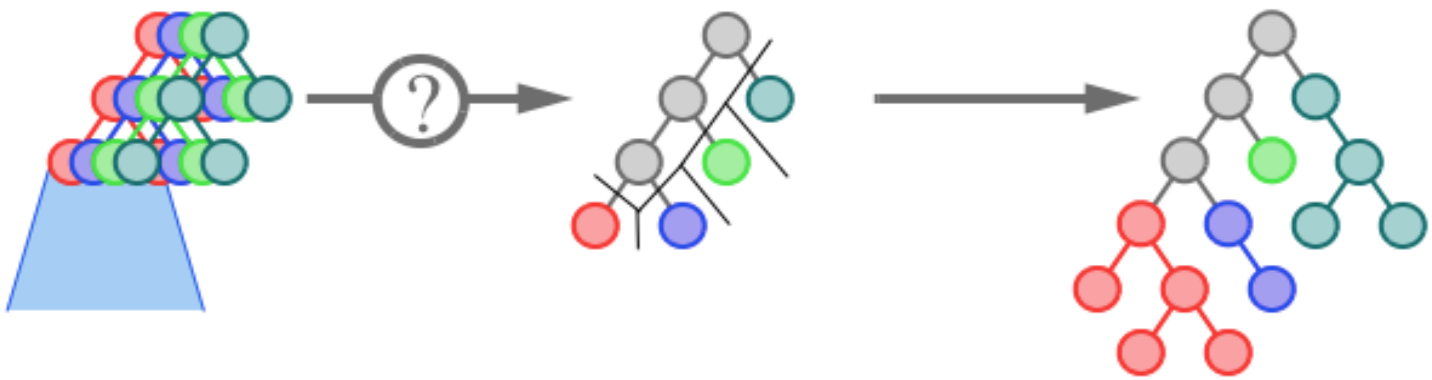
- Strong branching for variable selection
  - Divide free variables evenly
  - Processors compute bound triples for their free variables
  - All-reduce on best triples to determine branch var
  - All-reduce to compute lookahead bound

$$b(J,C,E,F) = \min_{f\in F}\left\{\max\left\{\begin{array}{l} b(J\cup\{f\},C,E,F-\{f\}) \\ b(J,C\cup\{f\},E,F-\{f\}) \\ b(J,C,E\cup\{f\},F-\{f\}) \end{array}\right.\right.$$

- Note: last element most computation: recompute equivalence classes

Crossing over

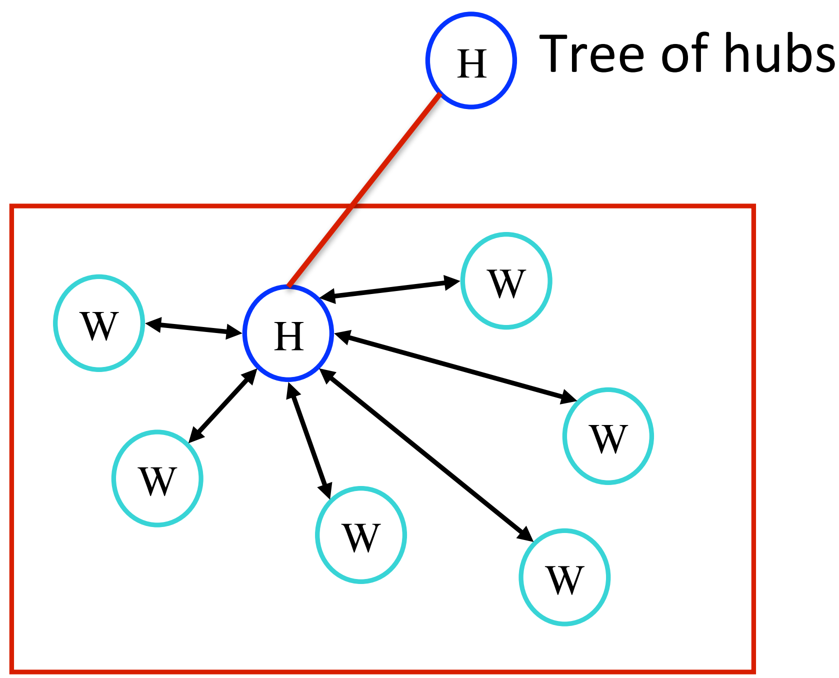
- Switch from parallel operations on one node to processing independent subproblems (serially)



- Work division by processor ID/rank
- Generally Crossover to parallel with perfect load balance
  - When there are enough subproblems to keep the processors busy
  - When single subproblems cannot effectively use parallelism
- For MMA: crossover when #open problems = N, the # of features

Hubs and Workers

- Control communication
  - Processor utilization
  - Approximation of serial order
- Subproblem pools at both the hubs and workers
- Hubs keep only *tokens*
  - Subproblem identifier
  - Bound
  - Location (processor, address)

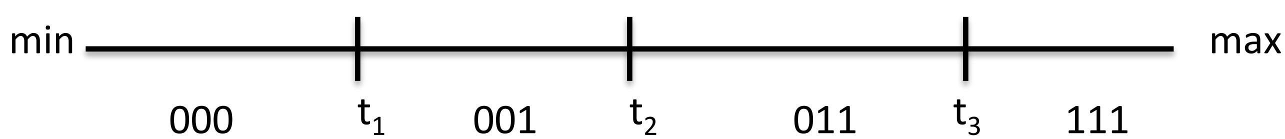


Load Balancing

- Hub pullback
- Random scattering
- Rendezvous
  - Hubs determine load (function of quantity and quality)
  - Use binary tree of hubs
  - Determine what processors need more work or better work
  - Exchange work

Experiments

- UC Irvine machine learning repository
  - Hungarian heart disease dataset ($M = 294$, $N = 72$)
  - Spam dataset ($M= 4601$, $N = 75$)
  - Multiple MMA instances based on boost iteration
    - Later iterations are harder
- Dropped observations with missing features
- Binarization of real features (Boros, Hammer, Ibaraki, Kogan)
  - Feature (i,j) is 1 iff $x_i \geq t_j$
  - Cannot map an element of Ω^+ and Ω^- to the same vector

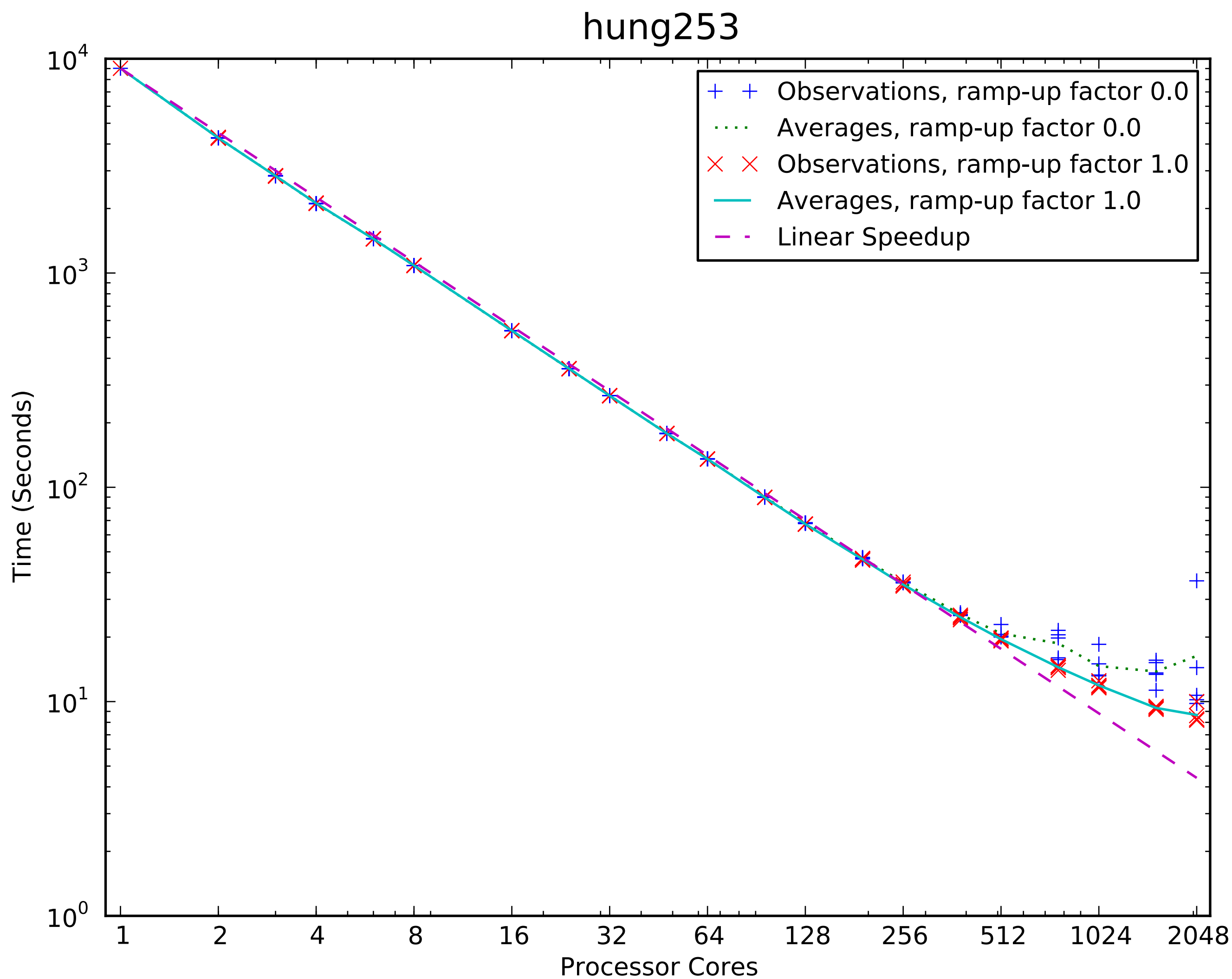


Red Sky

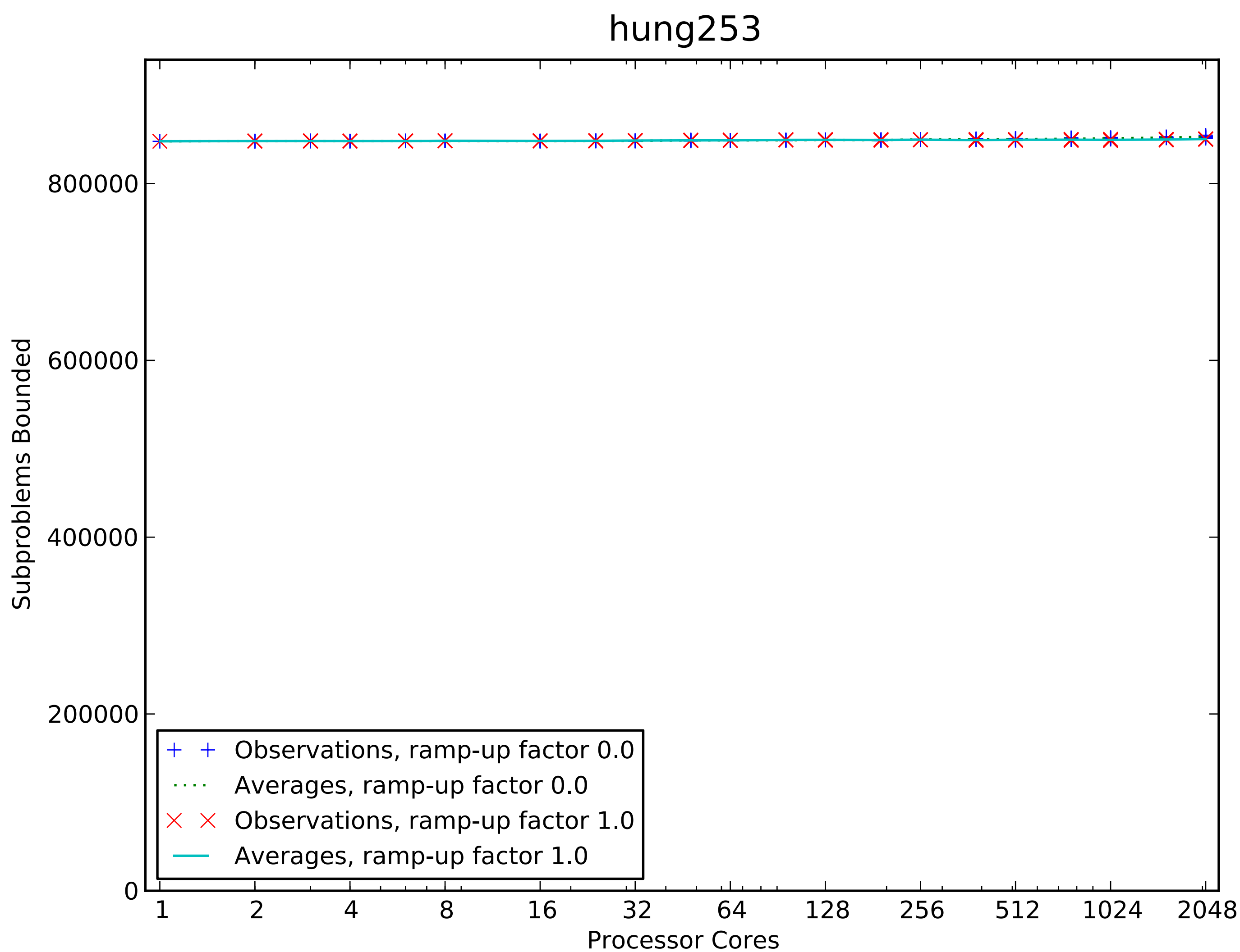
- Node: two quad-core Intel Xeon X5570 procs, 48GB shared RAM
- Full system: 22,528 cores, 132TB RAM
- General partition: 17,152 cores, 100.5TB RAM
  - Queue wait times OK for 1000s of processors
- Network: Infiniband, 3D torroidal (one dim small), 10GB/s
- Red Hat Linux 5, Intel 11.1 C++ compiler (O2), Open MPI 1.4.3

- Because subproblem bounding is slow, 128 workers/core

Value of ramp up (no enumeration)

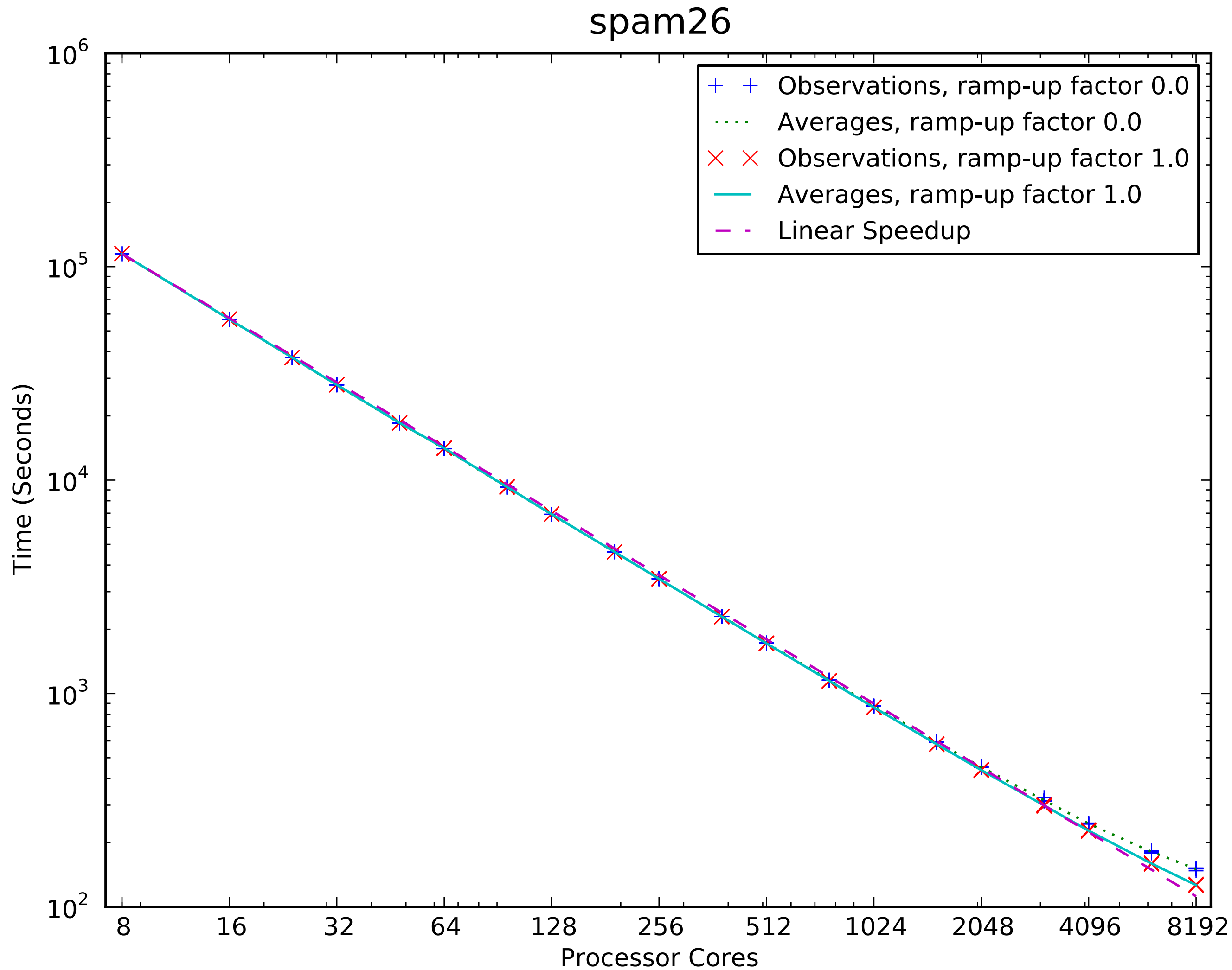


Number of tree nodes

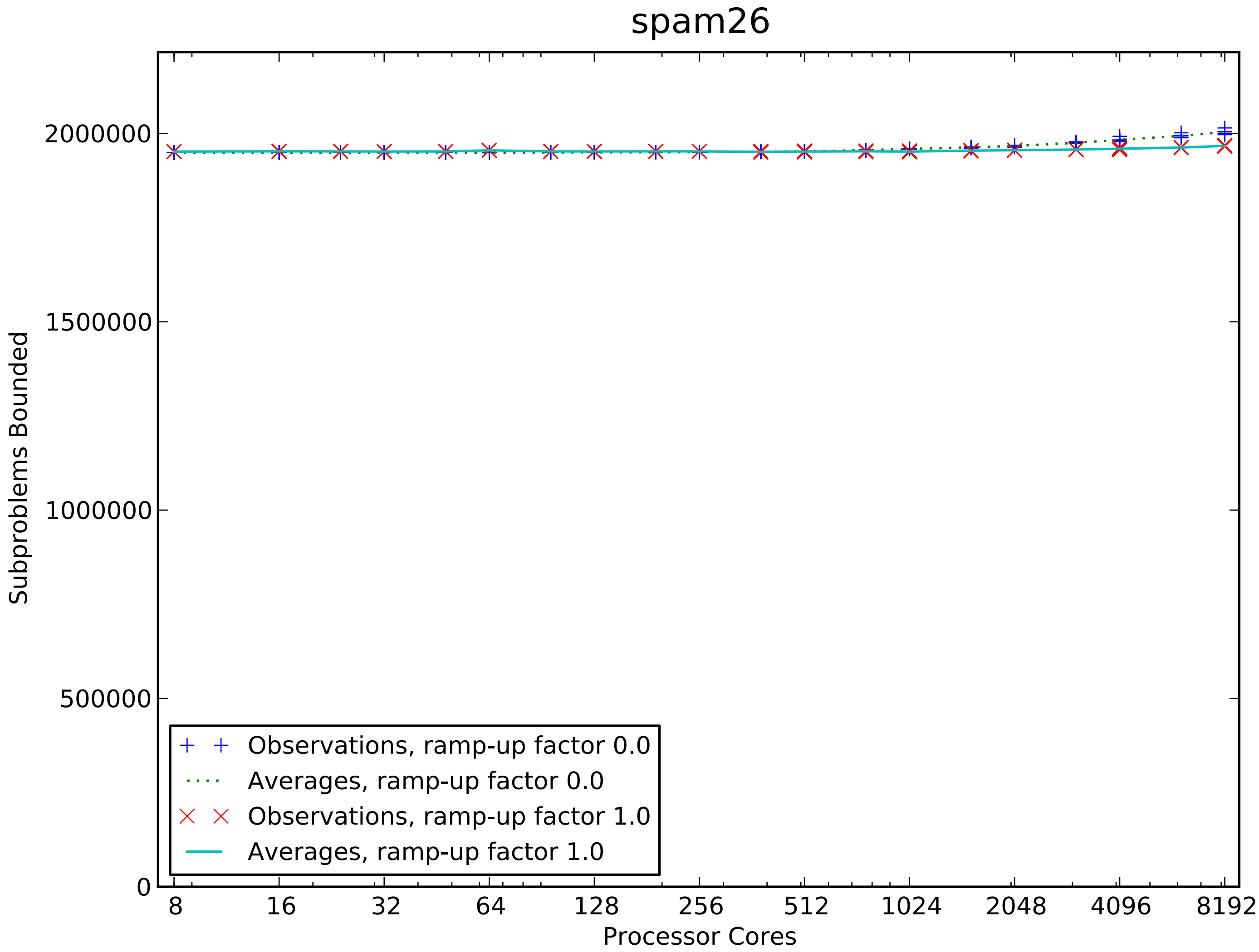


Sandia National Laboratories

Spam, value of ramp up



Spam, tree nodes



Comments: Ramp up

- Using initial synchronous ramp up improves scalability (e.g. 2x processors), reduces tree inflation.
- Speed up departure point from linear depends on problem difficulty and tree size.
  - Tree inflation is the main contributor to sub-linear speedup
- Solution times down to 1-3 minutes
  - Spam26: 3 min on 6144 cores, 27 hours on 8 cores
- For MMA no significant efficiency drop from 1 processor and going to multiple hubs

Parallel Enumeration

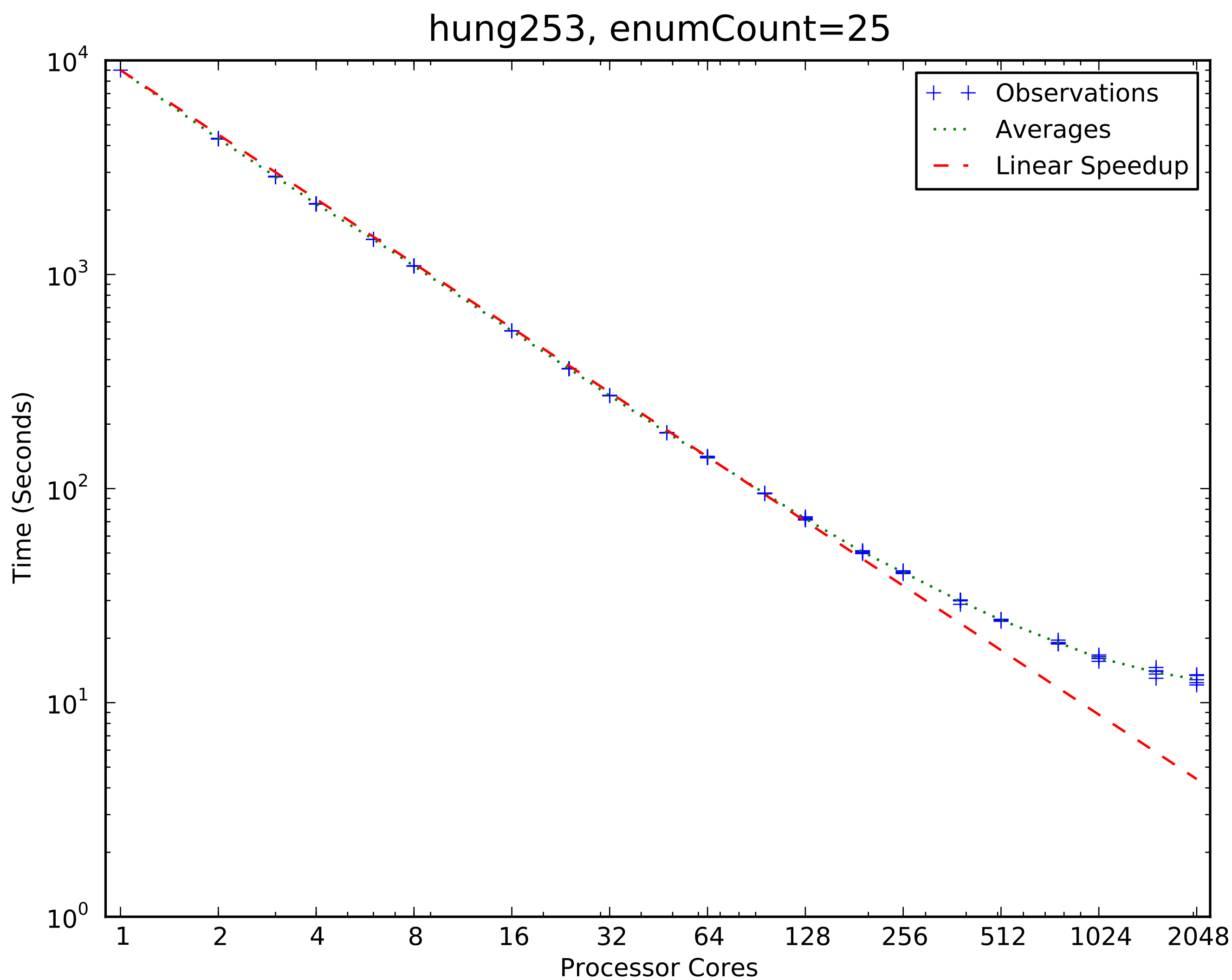
- Fundamental in PEBBL: best k, absolute tolerance, relative tolerance, objective threshold
- Requires: branch-through on “leaves” and duplicate detection
- Hash solution to find owning processor
- For all but best-k
  - independent solution repositories
  - parallel merge sort at end
- For k-best need to periodically compute cut off objective value

Enumeration Experiments

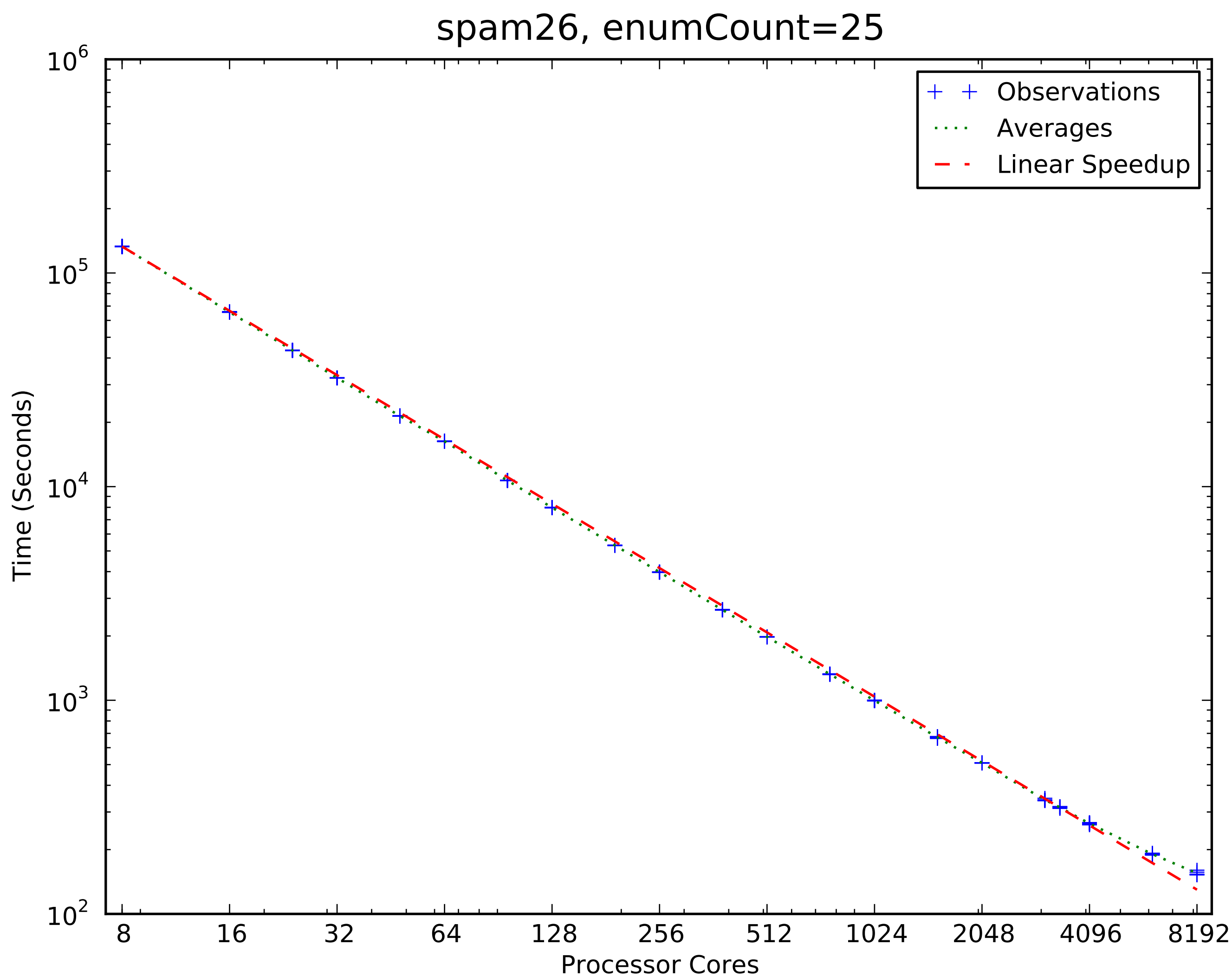
- Why Enumeration for MMA?
  - MMA is the weak learner for LP-Boost
  - Add multiple columns in column generation
    - In this case, add the best 25 MMA solutions

- Hungarian Heart
  - Tree size about same
  - More communication
- Spam
  - Larger tree with enumeration
  - Harder subproblems than Hungarian heart (more observations)

Results: Enumeration



Results: Enumeration



Open-Source Code Available

- Software freely available (BSD license)
  - PEBBL plus knapsack and MMA examples
- http://software.sandia.gov/acro
- ACRO = A Common Repository for Optimizers

Thank you!