Using a Computational Grid for Optimization

JEFF LINDEROTH

ISE Department
COR@L Lab
Lehigh University
jtl3@lehigh.edu

COR@L
COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Department of Industrial & Systems Engineering
University of Wisconsin-Madison
February 20, 2006

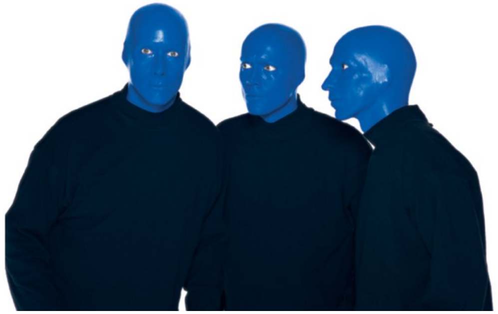
The Perfect Marriage

While my wife really likes this slide, I believe that Optimization and the Computational Grid really form the perfect marriage

Ingredients of the Perfect Marriage

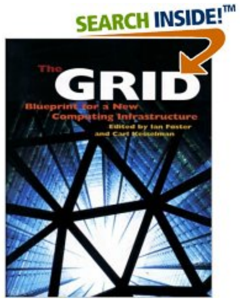
1. Something Old
   - Branch-and-Bound
   - Benders' Decomposition
2. Something New
   - Parallel Tree Search Techniques
   - Nonlinear Relaxations
   - Asynchronous Algorithms
   - Symmetry Handling
3. Something Borrowed
   - Your Computer
   - Cycles on Supercomputer Sites

Something Blue



- You can work until you're blue in the face, and still not solve some of these problems:
  - Quadratic Assignment Problem
  - Multistage Stochastic Programming
  - Symmetric Integer Programs

What's All the Hype?



Computational Grid

A hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities

High End Computational Capabilities

The TeraGrid

- $> 11,000$ Processors
- > 45 TFlops

NCSA — National Center for Supercomputing Applications

1774 CPU, Itanium-2 processors

Problems With Using Supercomputers

A Look At The Job Queue

```
[linderot@tunb ~] bqueues
QUEUE_NAME      PRIO      MAX JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
cap1             60         -    -    -    -   626   114   498     0
cap2             60         -    -    -    -   784   560   224     0
cap3             60         -    -    -    -   160     0   160     0
ind              60         -    -    -    -     0     0     0     0
debug            50         -    -    -    -     0     0     0     0
normal           30         -    -    -    - 10819  9691  1054     0
admin            30         -    -    -    -     0     0     0     0
```

- Nearly $10,000$ pending jobs!
- You can queue your job and wait (literally) days until it will run.
- This hardly seems consistent or pervasive

On a Positive Note

How Many Processors Are Available?

```
[linderot@tunb ~] bhosts | grep 'ok' | wc -l
    231
```

- There are 231 processors that are currently available!
- They are being saved to run the parallel job at the head of the queue
- "Backfill": We could use those processors for our computation, but we have to schedule them for a short time period
- Use the processors as part of a larger computation that can handle processors going away
- The idle cycles are pervasive!

Something Borrowed—Your CPU Cycles

- There are lots of CPU cycles going unused (or wasted) right now.
- Can I use your machine?
- I promise to give it back as soon as you want it
- CPU Cycles are a ubiquitous and nearly endless resource, if only you can harness them
- But how should they be harnessed?

Condor

THE UNIVERSITY of WISCONSIN MADISON

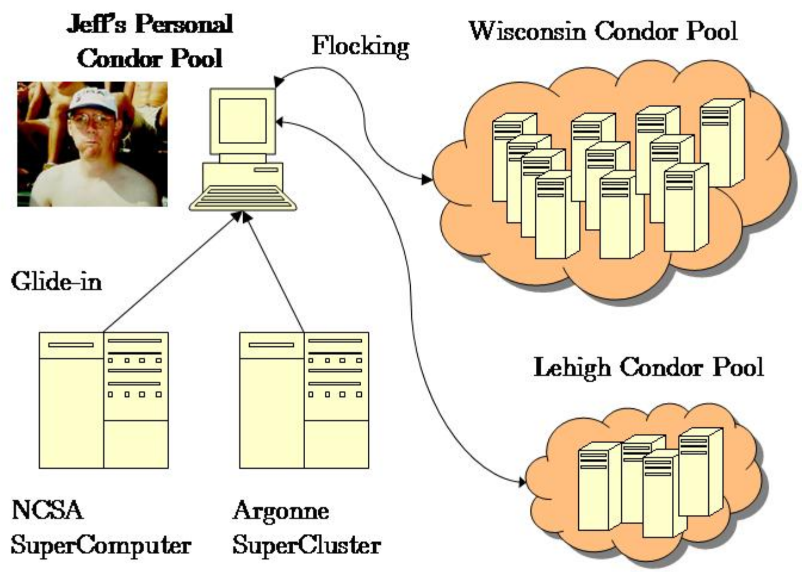
Jaime Frey
Peter Keller
Miron Livny
Erik Paulsen
Rajesh Raman
Marvin Solomon
Todd Tannenbaum
Doug Thain
Derek Wright
LOTS MORE

http://www.cs.wisc.edu/condor

Building Grids with Condor

- Manages collections of "distributively owned" workstations
  - User need not have an account or access to the machine
  - Workstation owner specifies conditions under which jobs are allowed to run—Jobs must vacate when user claims machine!
- How does it do this?
  - Scheduling/Matchmaking
  - Jobs can be checkpointed and migrated
  - Remote system calls provide the originating machines environment
- Flocking: Jobs in one Condor Pool can negotiate to run in other Condor pools
- Glide-in: Nodes can "temporarily" join an existing Condor pool.

Personal Condor



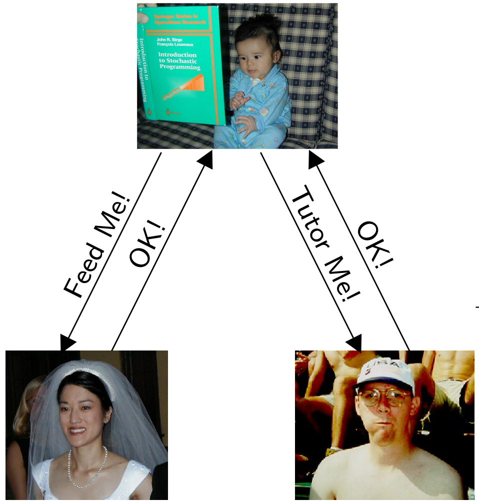
Grid-Enabling Algorithms

- Condor, with flocking and glide-in, gives us the infrastructure from which to build a grid (the spare CPU cycles),
- We still need a mechanism for controlling the algorithm on a computational grid
- No guarantee about how long a processor will be available.
- No guarantee about when new processors will become available

---

- To make parallel algorithms dynamically adjustable and fault-tolerant, we could (should?) use the master-worker paradigm
- What is the master-worker paradigm, you ask?

Master-Worker!



- Master assigns tasks to the workers
- Workers perform tasks, and report results back to master
- Workers do not communicate (except through the master)

- Simple!
- Fault-tolerant
- Dynamic

Making the Marriage Work

MW

- There are three abstraction in the master-worker paradigm: Master, Worker, and Task.
- MW is a software package that encapsulates these abstractions
  - API : C++ abstract classes
  - User writes 10 methods
  - The MWized code will transparently adapt to the dynamic and heterogeneous computing environment
- MW also has abstract layer to resource management and communications packages (an Infrastructure Programming Interface).
  - Condor/PVM, Condor/Socket, Condor/Files, Single processor
- It's Free!: `http://www.cs.wisc.edu/condor/mw`

MW API

- MWMaster
  - `get_userinfo()`
  - `setup_initial_tasks()`
  - `pack_worker_init_data()`
  - `act_on_completed_task()`
- MWTask
  - `pack_work()`, `unpack_work()`
  - `pack_result()`, `unpack_result()`
- MWWorker
  - `unpack_worker_init_data()`
  - `execute_task()`

MW Applications

- MWFATCOP (Chen, Ferris, L) – A branch and cut code for linear integer programming
- MWATR (L, Shapiro, Wright) – A trust-region-enhanced cutting plane code for two-stage linear stochastic programming and statistical verification of solution quality.
- MWKNAP (Glankwamdee, L) – A simple branch-and-bound knapsack solver
- MWQAP (Anstreicher, Brixius, Goux, L) – A branch-and-bound code for solving the quadratic assignment problem
- MWAND (L, Shen) – A nested decomposition-based solver for multistage stochastic linear programming
- MWSYMCOP (L, Margot, Thain) – An LP-based branch-and-bound solver for symmetric integer programs

Grids
Quadratic Assignment Problem
Multistage Stochastic LP
Code Design
Definition
Condor
MW

MWKnap

- Simple (self-contained) branch-and-bound solver included in MW distribution

Knapsack Problem

$$z^* = \max\{c^T x \mid a^T x \leq b, x \in \mathbb{B}^n\}$$

- Use to demonstrate how not to implement branch-and-bound with MW
- MW: Task consists of work and result

Grids
Quadratic Assignment Problem
Multistage Stochastic LP
Code Design
Definition
Condor
MW

Stoopid Ideas

Stoopid Idea #1

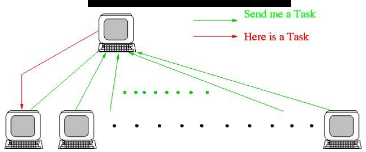
Let the "work" portion of task consist of evaluating one node

- Parallel Efficiency:

$$\eta = \frac{\Sigma(\text{Time workers execute tasks})}{\Sigma(\text{Time workers available})}$$

- $\eta < 1\%$

The Problem!



Grids
Quadratic Assignment Problem
Multistage Stochastic LP
Code Design
Definition
Condor
MW

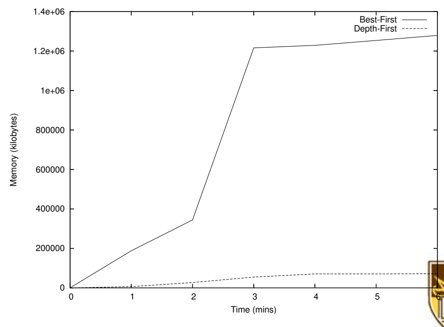
More Stoopid Ideas

Stoopid Idea #2

Let the workers search the trees in a best-first fashion

- The messages you pass back to the master get huge
- The master quickly exhausts all available memory

Memory Consumption



Something New

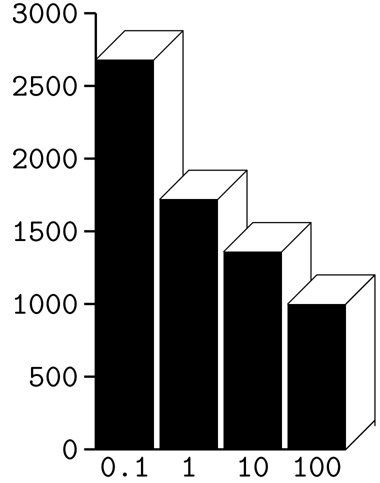
Parallel MW Tree Management Strategy

- Master:
  - Order list best-first
  - Switch to worst-first if the size of the list gets too big
- Worker:
  - Search subtree rooted at task's node in a depth-first fashion for t seconds.
  - Pass back unevaluated nodes in stack to form new tasks.

- $\boxed{\text{In Initial Implementation, } \eta = 0.41}$
- Since there is very little synchronization required in the branch and bound algorithm, this number is shockingly low!

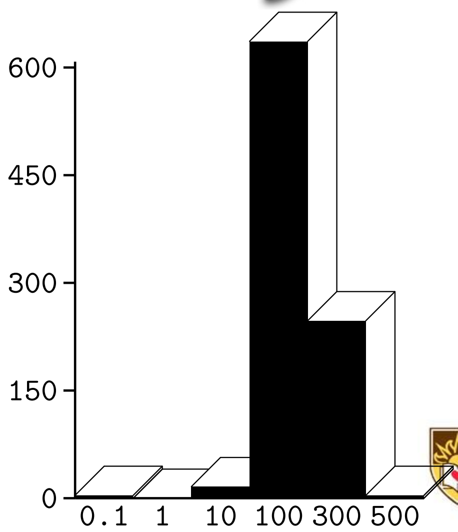
Deducing the Problem

- We may *want* the workers to examine a subtree for t seconds, but that doesn't mean that there are t seconds of work!
- A histogram of task times. (For $t = 100$)



Elementary, Dear Watson

- Make sure that workers only pass back nodes that will have enough "meat"
- Allow additional time for workers to pop up the DFS stack, finishing off remaining easy nodes.
- η improved to 0.9



QAP Collaborators

Kurt Anstreicher
University of Iowa

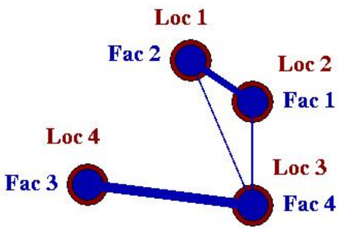
Nate Brixius
Micro$oft

Jean-Pierre Goux
Argonne, Northwestern, and Artelys

The Quadratic Assignment Problem

Mathematical Formulation

$$\min_{\pi \in \Pi} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)} + \sum_{i=1}^{n} c_{i\pi(i)}$$



- Assign facilities to locations
- QAP is NP-Super-Hard
- Branch and Bound is the method of choice, but very few tight, computable, bounds exist.

Odds and Ends

Something New

Kurt and Nate gave a tight nonlinear relaxation

$$\min_{X} \mathbf{vec}(X)^T ((B \otimes A) - (I \otimes S) - (T \otimes I)) \mathbf{vec} X + C \bullet X$$

$$\text{s.t. } Xe = X^T e = e, X \geq 0$$

- Convex quadratic relaxation
- Something Old! Used the Frank-Wolfe algorithm to solve the relaxation
- Something Old and New! Adapt "strong branching" to this context
- Engineer the algorithm for the Grid

Our Computational Grid

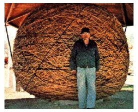
Number	Type	Location
414	Intel/Linux	Argonne
96	SGI/Irix	Argonne
1024	SGI/Irix	NCSA
16	Intel/Linux	NCSA
45	SGI/Irix	NCSA
246	Intel/Linux	Wisconsin
146	Intel/Solaris	Wisconsin
133	Sun/Solaris	Wisconsin
190	Intel/Linux	Georgia Tech
94	Intel/Solaris	Georgia Tech
54	Intel/Linux	Italy (INFN)
25	Intel/Linux	New Mexico
5	Intel/Linux	Columbia U.
10	Sun/Solaris	Columbia U.
12	Sun/Solaris	Northwestern
2510		

The Holy Grail

- (NUG30) ($n = 30$) had been the "holy-grail" of computational QAP research for > 30 years
- In 2000[1], Anstreicher, Brixius, Goux, & Linderoth set out to solve this problem
- Using an old idea of Knuth, we estimated the CPU time required to solve NUG30 to be 5-10 *years* on a fast workstation

[1] Something Old!

NUG30 is solved!



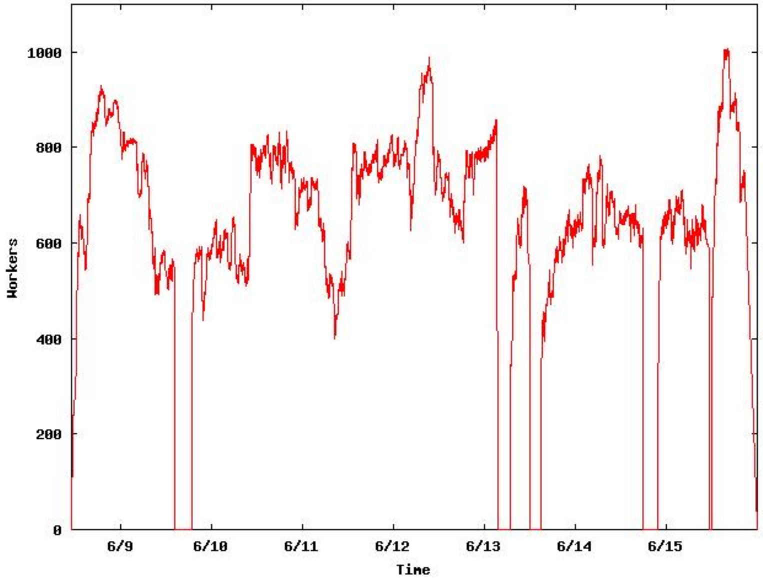
14, 5, 28, 24, 1, 3, 16, 15, 10, 9, 21, 2, 4, 29, 25, 22, 13, 26, 17, 30, 6, 20, 19, 8, 18, 7, 27, 12, 11, 23

NUG30 Computation

Wall Clock Time:	6:22:04:31
Avg. # Machines:	653
CPU Time:	$\approx$ 11 years
Nodes:	11,892,208,412
LAPs:	574,254,156,532
Parallel Efficiency:	92%

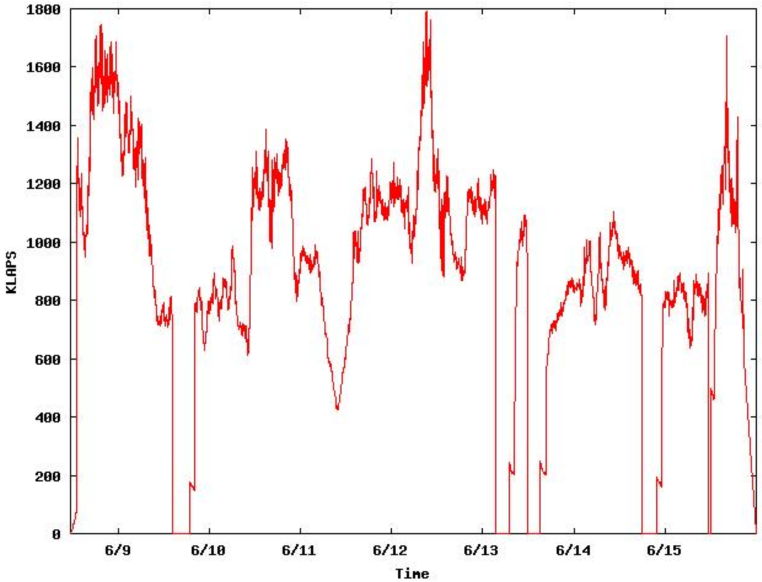
Grids
Quadratic Assignment Problem
Multistage Stochastic LP
Code Design
Background
Practical Branch-and-Bound

Workers



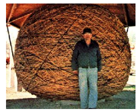
Grids
Quadratic Assignment Problem
Multistage Stochastic LP
Code Design
Background
Practical Branch-and-Bound

KLAPS



Grids
Quadratic Assignment Problem
Multistage Stochastic LP
Code Design
Background
Practical Branch-and-Bound

Solution of More QAP Instances



Even More Wasted CPU Time

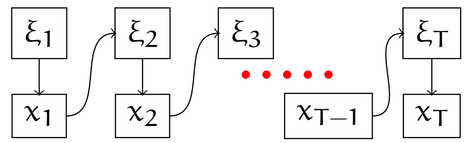
	KRA30B	KRA32	THO30
Wall Clock Time (Days)	3.79	12.3	17.2
Avg. # Machines	462	576	661
Max. # Machines	780	1079	1307
CPU Time (Years)	4.32	15.2	24.7
Nodes	5.14×10^9	16.7×10^9	34.3×10^9
LAPs	188×10^9	681×10^9	1.13×10^{12}
Parallel Efficiency:	92%	87%	89%

Multistage Stochastic LP

Jierui Shen
COR@L Lab
Lehigh University

Grids
Quadratic Assignment Problem
Multistage Stochastic LP
Code Design
Synchronicity
Data Management
Implementation Details
ShowOff

Multistage Decision Making

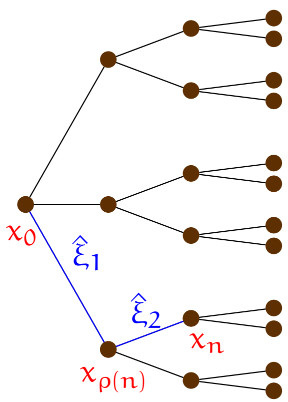


- Random vectors $\xi_1 \in \mathbb{R}^{n_1}, \xi_2 \in \mathbb{R}^{n_2}, \ldots, \xi_T \in \mathbb{R}^{n_T}$
- Make sequence of decisions $x_1 \in X_1, x_2 \in X_2, \ldots, x_T \in X_T$.

- Risk Neutral: We always aim to optimize the expected value of our current decision x_t
- Linear: Assume X_t are polyhedra
- Discrete: Assume ξ_t are drawn from a discrete distribution.

Grids
Quadratic Assignment Problem
Multistage Stochastic LP
Code Design
Synchronicity
Data Management
Implementation Details
ShowOff

Scenario Tree



- N: Set of nodes in the tree
- $\rho(n)$: Unique predecessor of node n in the tree
- $\mathcal{S}(n)$: Set of successor nodes of n
- q_n: Probability that the sequence of events leading to node n occurs
- x_n: Decision taken at node n

Grids
Quadratic Assignment Problem
Multistage Stochastic LP
Code Design
Synchronicity
Data Management
Implementation Details
ShowOff

Multistage Stochastic Programming

- Solving a multistage stochastic LP equivalent to solving a giant LP known as the Deterministic Equivalent

Deterministic Equivalent

$$z_{SP} = \min \left\{ \sum_{n \in N} q_n c_n^T x_n \;\middle|\; T_n x_{\rho(n)} + W_n x_n = h_n \;\forall n \in N \right\}$$

Something Old: Nested Benders' Decomposition

Value Function of node n

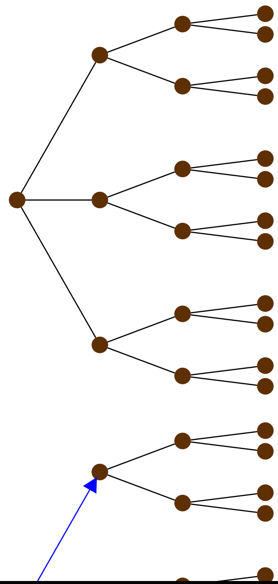
$$\mathcal{Q}_n(x_{\rho(n)}) \stackrel{\text{def}}{=} \min_{x_n} \left\{ c_n^T x_n + \sum_{m \in \mathcal{S}(n)} \hat{q}_{mn} \mathcal{Q}_m(x_n) | W_n x_n = h_n - T_n x_{\rho(n)} \right\}$$

- Nested Benders' Decomposition works by building under-approximations to each node's value function:

$$\mathcal{Q}_n(x_{\rho(n)}) \geq \min_{x_n} \left\{ c_n^T x_n + M_n^k(x_n) \;\middle|\; W_n x_n = h_n - T_n x_{\rho(n)} \right\} \qquad ((\mathrm{MLP}_n))$$

- Solution of child node's MLP gives "cuts", improving $M_n^k(\cdot)$

Action Pictures



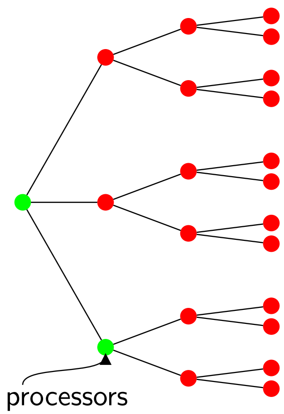
MWImplementation

Grid Programmers Do it in Parallel!

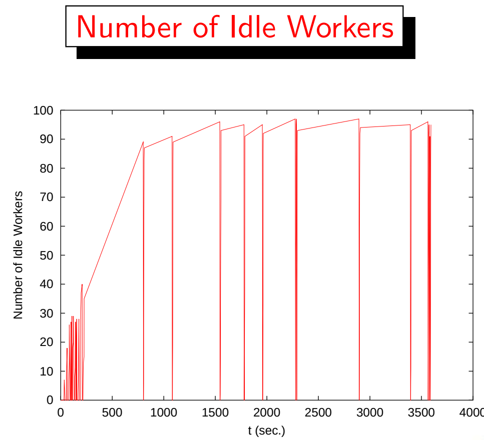
Nodes in nonoverlapping subtrees may be evaluated independently

- MWTask—Work
  - Collection of nodes (going the same direction) from the same stage
  - The $x_{\rho(n)}$ from these nodes
- MWTask—Result
  - (Forward): x_n
  - (Backwards): Cut(s) $F_{n[j]}^k, f_{n[j]}^k$
- `act_on_completed_task()` is responsible for updating node state and deciding which nodes to evaluate next

Synchronicity is Bad!



All processors waiting for this node to finish!



MW Implementation—Asynchronous

- It is *not* necessary to wait for all children ($\mathcal{S}_n$) to report in order to start a new evaluation of M_n^k
- Something Old: Ruszczyński ('93) also showed how to do nested decomposition in an asynchronous fashion and ensure convergence
- Something New: We have "reinvented-reinterpreted" some of these results.
- Each node has state({Forward, Backward}, ({ R, Y, G) })
  - Red: There's nothing useful I can do with this node
  - Yellow: Node is ready to be evaluated
  - Green: Node is being evaluated, and waiting for results.
- In `act_on_completed_task()`, node states are examined.
- In this method we relax assumption that new tasks can only be created if all child tasks gave been reported

A Grid Challenge: Cut Management

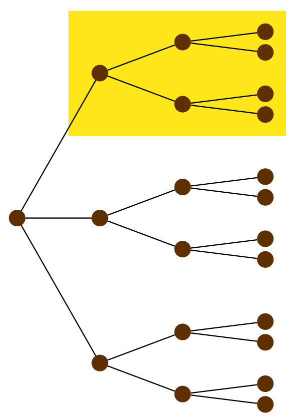
- We may require *lots* of memory to store the cuts
  - Ex.: 27,000 nodes in period T-1, each node contains 20 cuts, $x_n \in \mathbb{R}^{100} \Rightarrow \geq$ 400MB just to store cuts
- Grid: Since we don't have guarantees about worker processors, we cannot store cuts on the workers.
- Grid: All cuts (must) be stored on the master processor
  - Leads to memory overload of master
  - Leads to increased "service time" of the master for worker requests. (contention)
- Grid: $\Rightarrow$ We *must* do what we can to compress and reduce the number of cuts
  - Don't record duplicates
  - Aggregate nodes

Duplicate Cuts

- Random variables "time-independent"
  - You can "share" cuts among nodes at a stage
  - You need only keep one copy.
- Random variables "time-dependent"
  - It may be the case that the cost-to-go functions are *very* similar.
  - Example: Four stage problem, Nodes/stage (1, 32, 1024, 32768)

Period	Cuts	Duplicates
1	51	16
2	5055	5009
3	11264	11256

Cut Management—Aggregation

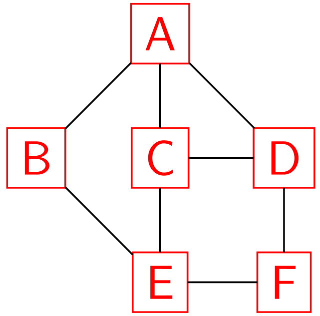


- Form the deterministic equivalent of a group of nodes, and treat this as one larger "supernode"
- Node subproblems get larger
- Fewer cuts

MWAND

- MW Asychronous Nested Decomposition
  - Magic WAND? :-)
- Uses the COIN Osi Interface to build MLP_n
- Uses the COIN Clp (simplex) solver to solve MLP_n
- Does not use the COIN-Smi to manipulate stochastic program
- SUTIL: Stochastic Programming Utility Library (Czyzyk & L)
  - Reads SMPS
  - Samples instances
  - Creates deterministic equivalents
  - It's Free! `http://coral.ie.lehigh.edu/sutil`

Telecommunication Network Design



- Set of stages T
- Set $\mathcal{J}$ of links
- Sets I_t of demands
- Random demand $d_t(\xi) \in \mathbb{R}^{|I_t|}$
- Budget each period
- Install capacity on links each period to minimize the total expected unserved demand

Some (Limited) Computational Results

- $T = 5$, Last three periods aggregated.
- Right now, we are using a "baby grid"
- Helpful Hint: Don't unleash your code onto a big grid unless you are reasonable sure it is working well.

Location	Number
Wisconsin	785
NCSA	1280
Argonne/U of C	288

Computational Results

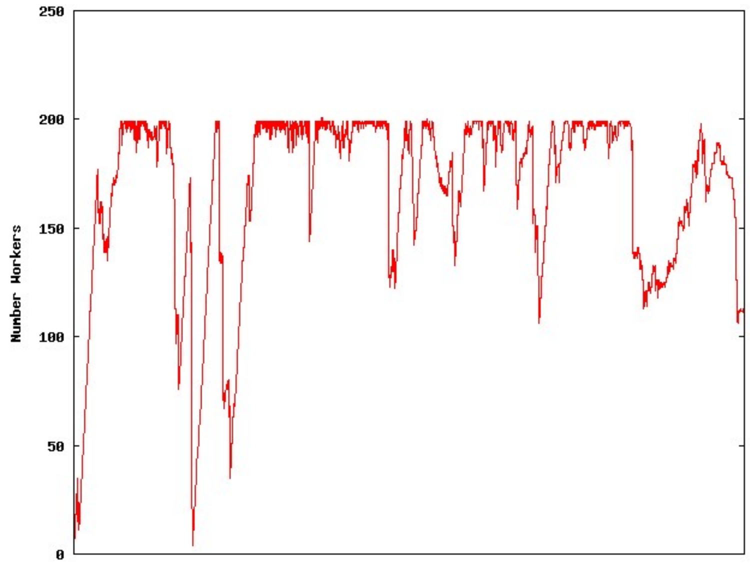
- K: Realizations/Period
- N: Number of scenarios
- DE: Size of deterministic equivalent

K	N	DE Size
30	0.81M	18M * 31M
50	6.25M	140M * 236M
60	12.9M	290M * 488M

- It: Number of iterations (Times MLP_0 was solved)

K	It	Avg Workers	Wall Time	CPU Time	η
30	9	62	2:34:21	6:15:15:10	67
50	7	75	1:12:49:27	85:20:24:15	77
60	11	162	3:16:51:00	431:12:15:37	73

Workers in Solving ssn5-60



Solving Symmetric IPs

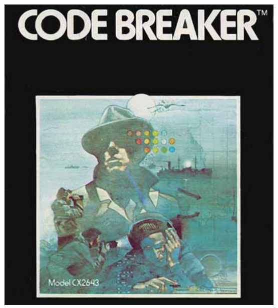
Carnegie Mellon

François Margot
Carnegie Mellon

The University of Wisconsin Madison

Greg Thain
University of Wisconsin-Madison

Code Design



- $W(v, \alpha)$: Set of all "words" of length v from alphabet $\{0, 1, \dots \alpha - 1\}$.
- $|W(\alpha, v)| = \alpha^v$
- We will abbreviate $W(v, \alpha) = W$
- A code is a subset $C \subseteq W$
- Hamming distance: $a \in W, b \in W$, $\text{dist}(a, b) = |\{i \mid a_i \neq b_i\}|$

Code Applications

Error Correcting Code

- Find $C \subset W$ such that $a \in C, b \in C \Rightarrow \text{dist}(a, b) \geq 2d + 1$
- Maximize $|C|$
- Application: Words in C submit over a "noisy" channel on which at most d letters are changed can be "self-corrected."

Covering Code

- Find a code $C \subset W$ such that every word $w \in W$ is at most a distance d away from at least one word in C
- $(\text{dist}(w, C) \leq d \ \forall w \in W)$
- Minimize $|C|$
- Application: Something far more practical

Are You Ready for Some Football!

- Predict the outcome of v soccer matches
- $\alpha = 3$
  - 0: Team A wins
  - 1: Team B wins
  - 2: Draw
- You win if you miss at most $d = 1$ games

The Football Pool Problem

What is the minimum number of tickets you must buy to assure yourself a win?

Grids
Quadratic Assignment Problem
Multistage Stochastic LP
Code Design
Background
Dealing with Symmetry
Improving Lower Bounds with Fixings
Solving It

Solutions for $v = 3$

Answer #1

1	2	3
0	0	0
1	1	0
1	0	1
0	1	1
2	2	2

Answer #2

1	2	3
2	0	0
1	1	0
1	0	1
2	1	1
0	2	2

- These solutions are isomorphic.
  - For first component: $2 \leftrightarrow 0$
- There are LOTS of isomorphic solutions:

1. "Rename" W,L,D for any subset of the matches: $(\alpha!)^v$
2. Reorder the matches: $v!$

- There are $(\alpha!)^v v! = 1296$ equivalent solutions for $v = 3$

Grids
Quadratic Assignment Problem
Multistage Stochastic LP
Code Design
Background
Dealing with Symmetry
Improving Lower Bounds with Fixings
Solving It

How Many Must I Buy?

Known Optimal Values

v	1	2	3	4	5
$\lvert C_v^* \rvert$	1	3	5	9	27

The Football Pool Problem

What is $\lvert C_6^* \rvert$?

- Despite significant effort on this problem for > 40 years, it is only known that

$$65 \leq C_6^* \leq 73$$

Grids
Quadratic Assignment Problem
Multistage Stochastic LP
Code Design
Background
Dealing with Symmetry
Improving Lower Bounds with Fixings
Solving It

But It's Trivial!

- For each $j \in W$, let $x_j = 1$ iff we word j is in code C
- Let $A \in \{0,1\}^{|W| \times |W|}$ with $a_{ij} = 1$ iff word $i \in W$ is distance $\leq d = 1$ from word $j \in W$

IP Formulation

$$\min e^T x$$

$$\begin{aligned} \text{s.t.} \quad Ax &\geq e \\ x &\in \{0,1\}^{|W|} \end{aligned}$$

CPLEX Can Solve Every IP

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

      0     0       56.0769   729                     56.0769     2200
*     0+    0                   0      243.0000       56.0769     2200   76.92%
*     0+    0                   0      110.0000       56.0769     2200   49.02%
                    56.5164   729      110.0000    Fract:  56     2542   48.62%
*     0+    0                   0      107.0000       56.5164     2542   47.18%
                    56.5279   729      107.0000     Fract:  6     2673   47.17%
*     0+    0                   0       94.0000       56.5279     2673   39.86%
*     0+    0                   0       93.0000       56.5279     2673   39.22%
Elapsed time =  90.03 sec. (tree size =  0.00 MB)
*    50+   50                   0       91.0000       56.5285    12242   37.88%
Elapsed time = 6841.16 sec. (tree size = 14.12 MB)
  31100 30002       60.1690   544       87.0000       57.1864  5467339   34.27%
  31200 30102       77.7888   216       87.0000       57.1864  5499451   34.27%
* 31200+28950                   0       86.0000       57.1864  5499451   33.50%
  31300 29044       58.9809   611       86.0000       57.1870  5511005   33.50%
Elapsed time = 9500.15 sec. (tree size = 18.70 MB)
  42700 39098       78.3242   197       85.0000       57.2845  7623200   32.61%
* 42740+36552                   0       83.0000       57.2845  7626440   30.98%
Elapsed time = 117349.90 sec. (tree size = 202.88 MB)
Nodefile size = 74.98 MB (61.52 MB after compression)
 465100 434311        66.8425   410       80.0000       58.0439 92473005   27.45%
```


What's the Problem!?: Symmetry

- π : Permutation of $\{1, 2, \dots n\}$
- $\pi(x) = \pi(x_1, x_2, \dots x_n) = (x_{\pi(1)}, x_{\pi(2)}, \dots x_{\pi(n)})$
- π is a symmetry of an IP if
  1. x feasible $\Leftrightarrow \pi(x)$ feasible
  2. $c^T x = c^T \pi(x)$
- $G(IP)$: Set of all symmetries of IP
- For covering design, $|G(IP)| = v!(\alpha)^v$
- $6! \times 3^6 = 524880$

Isomorphism Pruning

- For some permutation $g \in G(IP)$ and set of indices $S \subset \{1, 2, \dots, n\}$, let

$$g(S) = \{g(i) \mid i \in S\}$$

- At a node a of the branch-and-bound tree
  - $F_1^a = \{i \mid x_i \text{ fixed to 1 at } a\}$
  - $F_0^a = \{i \mid x_i \text{ fixed to 0 at } a\}$
- Nodes a and b are isomorphic if

$$\exists g \in G(IP) \text{ with } g(F_1^a) = F_1^b, g(F_0^a) = F_0^b$$

- You may prune one of a or b. [Bazaraa, Kirca 83]

Something Old: Minimum Index Branching

[Butler, Ivanov, Lam, Margot, McKay, Read, Stinson, ...]

- The set $\{g(S) \mid g \in G(IP)\}$ is an equivalence class of all equivalent "relabelings" of $S \subseteq \{1, 2, \dots, n\}$
- Choose one representative for each potential set of variable fixings.
- For example, in Minimum Index Branching, A set S is a representative of its equivalance class if

$$S = \text{lexmin}\{g(S) \mid g \in G(IP)\}$$

- Isomorphism Pruning:
  - If F_1^a is not a representative, then prune node a.

Results (All Thanks to François)

The Good

- For $d = 1, v = 5, \alpha = 3$, Isomorphism Pruning can establish $|C_5^*| = 27$ in 1409 nodes, 82 seconds.
- CPLEX (v9.1) does not solve the problem in more than 4 hours.

The Bad and Ugly

- For $d = 1, v = 6, \alpha = 3$, Isomorphism Pruning gets nowhere..
- $|C_6^*| \geq 61$ after *long* running time.

Subcodes

- Partition W into words that start with each letter
- $W(6,3) = W_0 \cup W_1 \cup W_2$
- $w \in W_0$ covers 11 words in W_0
- $w \in W_1$ covers 1 words in W_0
- $w \in W_2$ covers 1 words in W_0
- An optimal code has
  - $C_0^* \subset W_0$, $|C_0^*| \stackrel{\text{def}}{=} y_0$
  - $C_1^* \subset W_1$, $|C_1^*| \stackrel{\text{def}}{=} y_1$
  - $C_2^* \subset W_2$, $|C_2^*| \stackrel{\text{def}}{=} y_2$

- So if a code of size $|C^*| = M$ exists, then it must satisfy

Covering System

$$\begin{aligned} 11y_0 + y_1 + y_2 &\geq 243 \\ y_0 + 11y_1 + y_2 &\geq 243 \\ y_0 + y_1 + 11y_2 &\geq 243 \\ y_0 + y_1 + y_2 &= M \end{aligned}$$

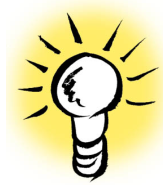
Repeat! (Something "Old")

- This same idea applies to fixing $m > 1$ component

Nice Idea! (Östergård and Weakley)

- Pick an $M = |C^*|$
- Enumerate all non-isomorphic solutions to covering system for $m = 1, 2, \ldots 6$
- If for some m, there are no solutions, then $M + 1$ is a valid lower bound on $|C^*|$
- Östergård and Weakley (2000): Able to show $M = 62$ is optimal code length for $d = 1, v = 9, \alpha = 2$
- Östergård and Wassermann (2002): Able to show $M \geq 65$ for $d = 1, v = 6, \alpha = 3$. Required over 1 CPU year!

Extending the Idea



Something New!

Combine the subcode fixing with IP.

- For some (small) m, and optimal code size M, enumerate all non-isomorphic solutions to the covering system
- This gives a list of possible y values, e.g. for $m = 1$ you get a list of triples (y_0, y_1, y_2)
- For each member of the list, solve the "Sequence IP"

Sequence IP (M, y_0, y_1, y_2)

$$
\begin{aligned}
&\min e^T x \\
\text{s.t.} \quad & Ax \geq e \\
&\sum_{i \in W_0} x_i = y_0 \\
&\sum_{i \in W_1} x_i = y_1 \\
&\sum_{i \in W_2} x_i = y_2 \\
&e^T x \leq M \\
&x \in \{0,1\}^{|W|}
\end{aligned}
$$

Improving the Lower Bound on C_6^*

If you find no solution, then $M+1$ is a valid lower bound

How to Solve It: SYMCOP

- There were even more "pre-pre" processing tricks that were used to reduce the number of sequence IPs to solve
- Final total: 346 sequence IPs
- François and Jeff implemented these ideas using the old MW-FATCOP framework for MILP.
- Many of the engineering/tuning ideas from the knapsack and QAP experience were used in this implementation
- Greg has made numerous improvements in MW's robustness and has been great at scavenging cycles in an effort to solve this problem
- Our mission: establish $C_6^* \geq 70$

Doh!

- I really,really, really wanted to announce that $C_6^* \geq 70$ today.
- Sadly, only 284 of the 346 sequence IPs have been completed.
- Our Personal Condor master machine is lying in pieces
- But it has been working hard!

Not For a Lack of Trying!

Statistics so far...

Wall Time	30.7 days
CPU Time	36.24 years
Avg Workers	455.8
Max Workers	1253
Total Nodes	6.51×10^8
Total LP Pivots	4.46×10^{11}
Parallel Performance	95.6%

Conclusions

- Optimization and the Grid really are "The Perfect Marriage"
- As my wife Helen so often tells me, even the most perfect marriage requires a little bit of work
- You may need to "tune" your optimization algorithm to run nicely on a Grid platform
  - Exploit dynamic grain size
  - Re-examine search strategy
  - Make algorithm run asynchronously
- The boost in computing power that you get by making your code/algorithm Grid-enabled is worth the effort!

It's Not All "On The Grid"

Recent Applications

- Stochastic Network Interdiction
- Stochastic Vehicle Routing

Recent Work with Companies

- Agere Systems: Product Portfolio Management
- Air Products and Chemicals: Rescheduling for Bulk-Gas Production/Distribution
- Air Products and BOC Gases: Product Swap Contract Valuation
- FCC: Implementation for Combinatorial Auction

But Wait, There's More

Other Current Research Areas

- Computational Integer Programming
  - Georgia Teach & UW-Madison
  - SAS Institute
- Mixed Integer Nonlinear Programming
  - Argonne National Lab
- Quasi-Monte Carlo Sampling for Stochastic Programming
  - Northwestern University and UW-Madison

The End

- MW: `http://www.cs.wisc.edu/condor/mw`
- COR@L: `http://coral.ie.lehigh.edu`
- `mailto:jtl3@lehigh.edu`