Solving the World LARGEST Quadratic Assignment Problem on Metacomputers

Jean-Pierre Goux
(Northwestern / Argonne.)

Collaborators
Kurt Anstreicher (Iowa)
Nate Brixius (Iowa)
Jeff Linderoth (Argonne)

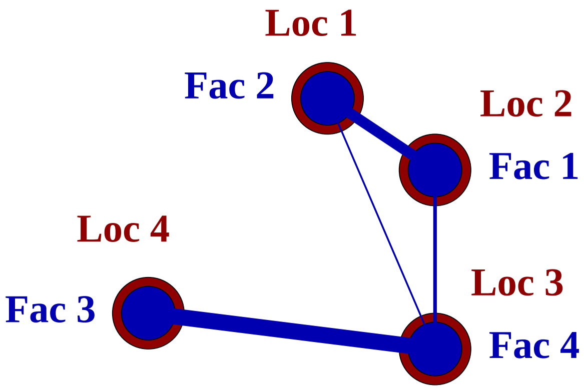
Optimization Seminars
Northwestern University

Feb 1. 2000

Outline

- The Quadratic Assignment Problem
- Branch-and-Bound
  - QAP bounds
  - The QPB bound
  - Branching Strategies
  - Knuth's estimator
- Parallel Implementation
  - Condor & MW
  - Parallel B&B with MW
  - Normalized Statistics
- Numerical Results
  - Grainsize determination
  - Parallel Efficiency
  - NUG25 numerical results
- Conclusion and Future Work

Quadratic Assignment Problem

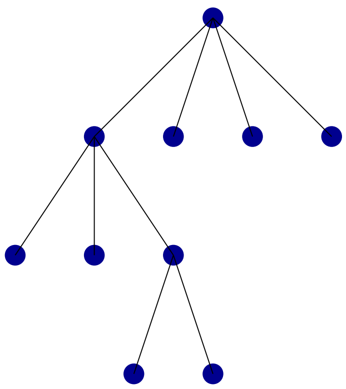


$$\min_{\pi \in \Pi} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)} + \sum_{i=1}^{n} c_{i\pi(i)}$$

$\mathcal{NP}$ hard problem, complexity $n!$

Enumeration : NUG25 : 150,000,000 years.

$\leadsto$ Branch-and-Bound (66 days).



QAP bounds

Trade-off Quality / Computational Effort.

- Semi-definite Programming bounds (SDP)
- Gilmore-Lawler bound (GLB)
- Projected Eigenvalue Bound (PB)
- Quadratic Programming Bound (QPB)

Gilmore-Lawler Bound

$$l_{ij} = \langle a_i^{'}, b_j^{'} \rangle_{-} + a_{ii} b_{jj} + c_{ij}$$

GLB := LAP with cost matrix L.

Projected Eigenvalue Bound

$$QAP(A,B,C) = \min_{x \in X} tr(AXB + C)X^\top$$

$$\min_{x \in \mathcal{O}} trAXBX^\top + LAP(C)$$

$$\min_{x \in \mathcal{O}} trAXBX^\top = \langle \lambda(A), \lambda(B) \rangle_-$$

$$PB(A,B,C) = \langle \lambda(\hat{A}), \lambda(\hat{B}) \rangle_- + LAP(D) - \frac{1}{n^2} s(A)s(B)$$

PB grows slowly with branching.

Convex QP Bound

Anstreicher – Brixius (1999)

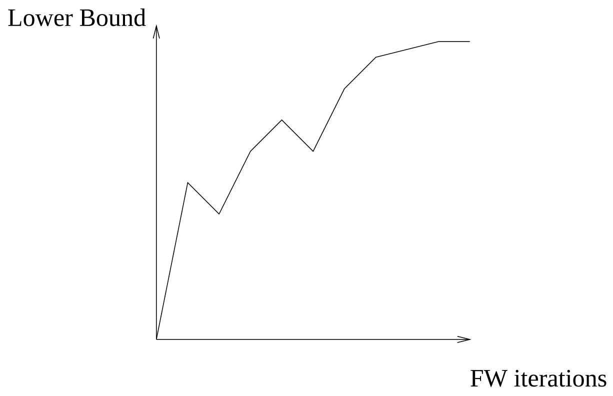
$$tr(AXB+C)X^\top =$$

$$\langle \lambda(\hat{A}), \lambda(\hat{B}) \rangle_- + DX + vect(\hat{X})^\top \hat{Q} vect(\hat{X}) - \frac{1}{n^2} s(A)s(B)$$

$$QP(A,B,C) = \min_{X \in \mathcal{E}, X \geq 0} vec(X)^\top Q vec(X) + CX$$

$$QPB(A,B,C) = \langle \lambda(\hat{A}), \lambda(\hat{B}) \rangle_- + QP(A,B,C)$$

QP solved approximately with Frank-Wolfe
FW iterations control accuracy.



Branching Strategies

(ROW)	choose location, fix facilities
(COLUMN)	choose facility, fix locations
(MIXED)	choose location or facility (50 % better)

No pseudo-costs but QP dual information U

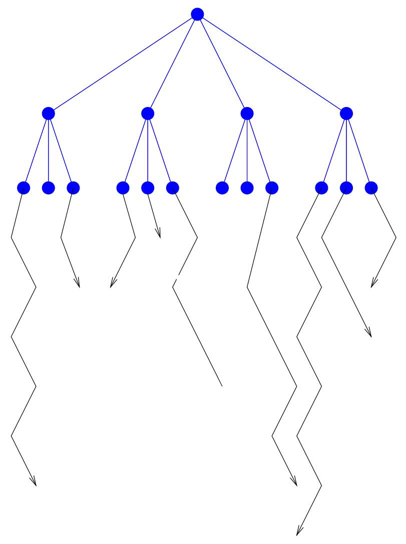
- Max sum (rows / columns)
- Max entries $>$ upperbound (rows / columns)
- Look-ahead (bounds computed few rows / columns)
- Look-ahead and U' information

- Branching rules $\pm$ expensive
- Branching rules evolves with depth
- Branching rules evolves with duality gap

complex B&B $\leadsto$ tuning tool

Knuth's Estimator

- Tree size estimator
- CPU time estimator
- Help decide whether problem solvable or not
- Help tune algorithm parameters (branching/bounding)

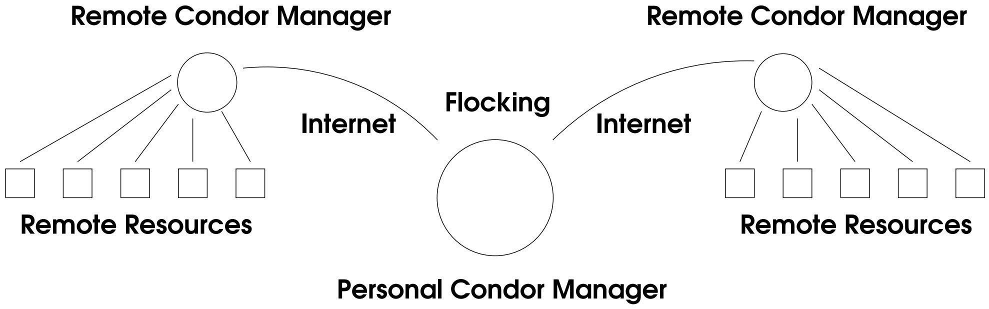


- Child nodes randomly chosen
- Perfect fit for "high" nodes
- Under estimate for "low" nodes (new)
- Biased-estimator in progress (biased by duality gap)

The Condor System

www.cs.wisc.edu/condor

- Scavenger system, hunter of Idle workstations
- Resource Manager of individually owned machines
- Owner of workstation defines usage policy
- PVM Support
- Flocking, Glide-in, Personal Condor



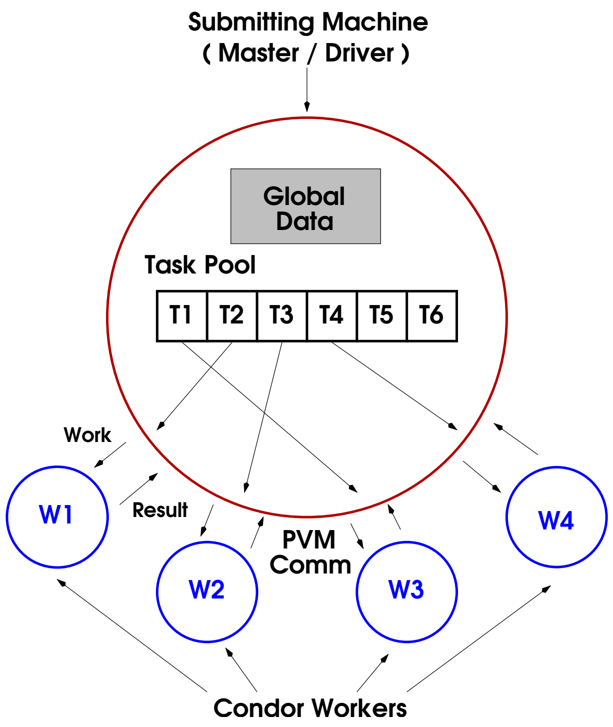
MW Development Framework

Master-Worker Metacomputing

Yoder - Linderoth - Goux

www.cs.wisc.edu/condor/mw

- C++ classes : MWDriver, MWWorker, MWTask
- Fault-tolerance, task scheduling, resource ranagement & discovery, interprocess communication



MW Abstraction

Driver

- `get_userinfo()`
- `setup_initial_tasks()`
- `pack_worker_init_data()`
- `act_on_completed_task()`

Task

- `pack_work()`, `unpack_work()`
- `pack_results()`, `unpack_results()`

Worker

- `unpack_init_data()`
- `execute_task()`

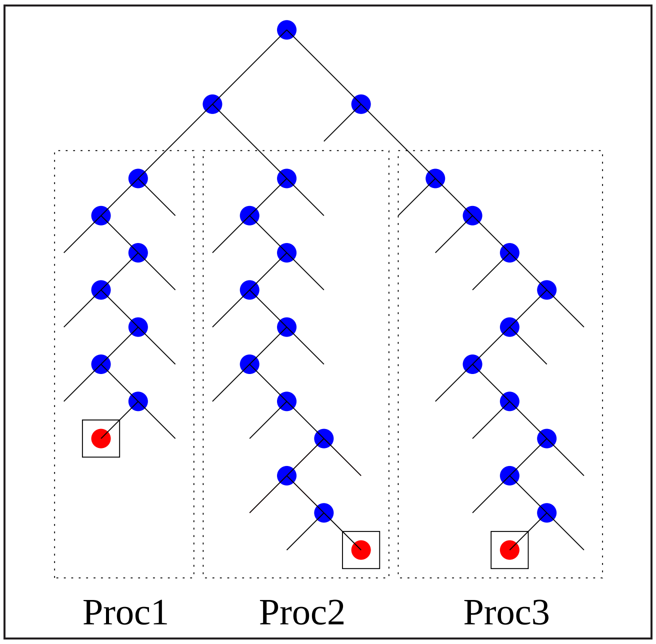
Parallel Strategy

1 Worker $\Longleftrightarrow$ 1 QAP bound

$\leadsto$ grainsize too small

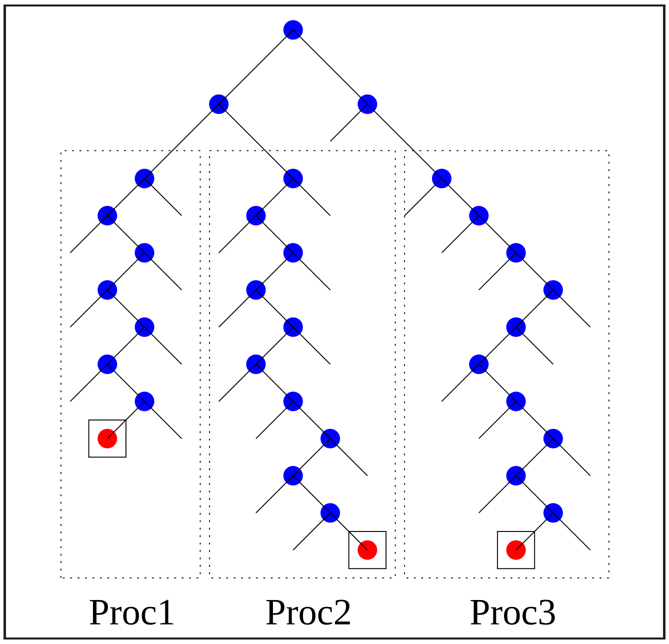
$\leadsto$

1 Worker $\Longleftrightarrow$ 1 subtree



distributed task pools...

QAPWorker



Go Depth-first until

- Subtree fathomed (Task Pool empty)
- Solution found
- Worker_CPU > max_CPU (100s)
- Plug sequential B&B

Initial Data

- Matrices A, B, C
- Set of branching/bounding parameters
- Symmetry breaking information

$$\boxed{\textbf{QAPTask}}$$

Work

- Assigned facilities / locations
- max_CPU
- Newest Upper bound

Result

- Integer Feasible Solution (if any)
- Remaining Task Pool (if any)
- Cumulated Statistics

QAPDriver

- Manage main TaskPool.
- Store current best permutation.
- Store current upperbound / lowerbound.
- Merge and Store statistics information.
- Maintain workers information.
- Maintain normalized statistics.
- Provide next node : Deepest (Memory).
- Check stopping criteria.

QAPLIB Challenge

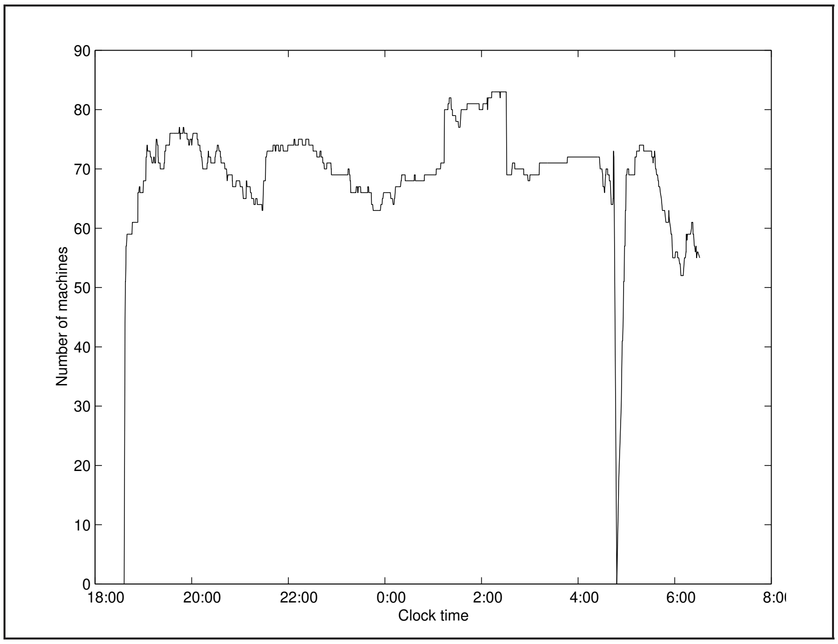
http://www.imm.dtu.dk/~sk/qaplib/

Hahn & Co (sequential)
Marzetta & Co (parallel)

NUG22	Seq.	Ultra 360Mhz	56 Hours	Hahn
NUG24	Seq.	Ultra 360Mhz	9 days	Hahn
NUG25	Seq.	Ultra 360Mhz	66 days	Hahn
NUG22	Par.	48..96 Cenju-3	9 days	Marzetta(1)
NUG25	Par.	64..128 Paragon	30 days	Marzetta(1)

NUG22 with MW-QPB (run 1)

solved in 11 hours.



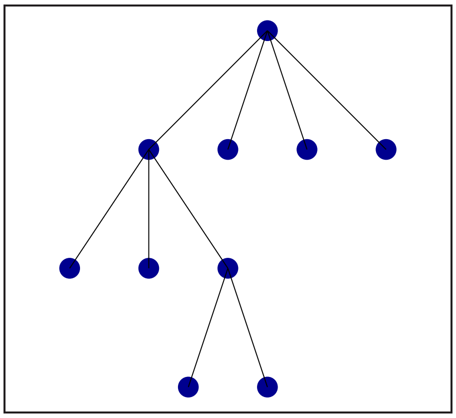
- Machines : 116 Intel/Solaris, 39 Sun/Solaris
- RAM : from 32 to 512 MB
- CPU : from 50 to 446 MIPS
- 25 days cumulated CPU time (not normalized)
- 80 % efficiency (and potentially less)

$\leadsto$ OK but not satisfied !!!!

Parallel Efficiency

- Master-Worker limitations ?
- average grainsize 8 seconds (OK...)
- analysis of tasks grainsize distribution
- 36% of the tasks have a grainsize smaller that 0.05s !
- 56% of the tasks have a grainsize smaller that 0.5 !
- 87% of the tasks $<$ 0.5s rooted at nodes deeper than 6.
- most of the tasks of depth > 6 are very small ($\neq$ MIP, MINLP)
- Where are these small/deep tasks coming ?
- $\rightsquigarrow$ Minimize returned number of these tasks.

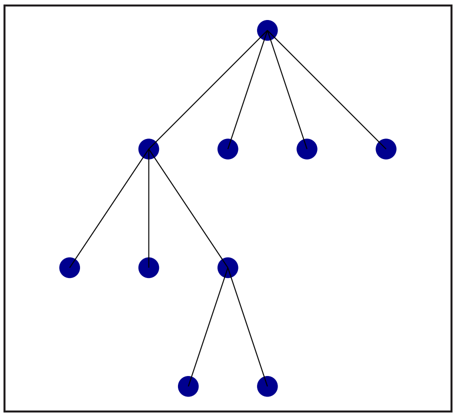
Finish Up strategy



- Workers work for max_cpu (100) seconds
- if (done) OK
- if (!done)
  - allow an extra max_cpu seconds to finish nodes deeper than `max_returned_depth`.
  - allow an extra max_cpu seconds to finish nodes deeper than `max_returned_depth + 1`.

$\rightsquigarrow$ fast tasks reduced from 56% to 26%.

Child re-ordering strategy



- still too much small tasks even for high depth in tree.
- we want for each level to do small/short tasks first.
- how ???
- Before : child introduced by alphetical order. (arbitrary).
- Now : order child based on bound information !!!!!
- $\leadsto$ no more small tasks !!!!
- average grainsize around 50-60 seconds, good distribution
- $\leadsto$ efficiency up to 95% !!!
- both heuristics work well together, more resources OK

MW Normalized Statistics

Metacomputer := dynamic / heterogeneous
Impossible to cross-compare runs
Impossible to evaluate speed-up
$\rightsquigarrow$ normalized/independent statistics

$p(i)$	**Machine Relative Performance**
$\mathcal{P}_1$	**Average Worker Performance**
$\mathcal{P}_2$	**Normalized Average Worker Performance**
$\mathcal{N}_p$	**Average number present workers**
$\mathcal{N}_{ns}$	**Average number non susp workers**
$\mathcal{N}_a$	**Average number active workers**
$\mathcal{P}$	**Equivalent Pool Performance**
η	**Parallel Performance**
$\mathcal{T}$	**Equivalent Run Time**

$\mathcal{W}$	T	$\mathcal{T}$	$\lvert\mathcal{I}\rvert$	$\mathcal{N}_{ns}$	$\mathcal{N}_a$	$\mathcal{P}_2$	$\mathcal{P}$	η
1780	37671	5739	49	23.5	21.2	0.15	3.2	0.90
886	25998	5777	52	33.8	29.3	0.22	6.5	0.86
886	26324	5693	74	36.3	29.7	0.22	6.4	0.81
1191	32438	5676	84	35.8	27.3	0.17	4.76	0.76
945	32283	5789	84	40.2	34.1	0.18	6.12	0.85
1033	32569	5625	81	37.8	31.5	0.17	5.44	0.83
959	26730	5756	54	30.3	27.9	0.22	6.00	0.92
889	21589	5803	67	27.3	24.3	0.27	6.52	0.88
707	15529	5975	59	26.1	21.9	0.38	8.44	0.84
667.9	8814.3	6019	43	15.5	13.2	0.68	9.01	0.85
505.7	8813.2	6059.1	43	19.1	17.4	0.69	12.0	0.91
489	8844	6054.4	49	20.3	18.1	0.68	12.4	0.88

MW-QAP

Quadratic Assignment Problem Solver

NUG25 World Record : 6.7 hours (01/20/2000)

Machines	Linux / Solaris
Average # Machines	93
Equivalent Pool Performance	42
Parallel Efficiency	97%
Nodes	80,430,341
Total time	27 days
Normalized time	12.7 days

NUG30 attempt (01/26/2000)

Machines	Linux / Solaris
Nodes Explored	1,101,427,278
Wall Clock time	5.5 days
Total CPU time	456 days
(under) Estimated Nodes	6,000,000,000
(under) Estimated Time	1 month

Conclusions

- Metacomputing works
- Metacomputing tools exist
- Metacomputing suited for (some) optimization
- Combination algorithm/platform/parallel impl
- Large problems allow to :
  - improve algorithms
  - understand algorithm scalability
  - improve tools
- Cycles would have been wasted anyway !!

Future Work

- Improvement Estimator
- Improvement Bound/Branching
- Improvement Algorithm
- Scale number of resources