Solution of Large-Scale Quadratic Assignment Problems Using Computational Grids

Kurt M. Anstreicher
Dept. of Management Sciences
University of Iowa

Nathan Brixius
Dept. of Computer Science
University of Iowa

Jean-Pierre Goux
MCS Division, Argonne National Lab
and Northwestern University

Jeff Linderoth
MCS Division, Argonne National Lab

Outline:

1. The Quadratic Assignment Problem
2. A Quadratic Programming Bound
3. Computing QPB
4. The Branch and Bound Algorithm
5. Computational Results
6. Summary

1 The Quadratic Assignment Problem

$$\text{QAP} \qquad \min_{X \in \Pi} \ \mathrm{tr}(AXB + C)X^T,$$

where Π denotes the set of permutation matrices. Typically:

a_{ik}=flow from facility i to facility k,

b_{jl}=distance from location j to location l,

c_{ij}= fixed cost of assigning facility i to location j,

$x_{ij} = 1$ if facility i assigned to location j, 0 otherwise.

Throughout assume A and B are symmetric.

QAP is NP-Hard. In practice instances with $n \geq 25$ are very challenging to solve to optimality.

Solution Methods for QAP:

- Heuristics
- Exact Methods
  - Polyhedral
  - Branch and Bound (B&B)

Bounds for QAP:

- Gilmore-Lawler Bound (GLB)
- Eigenvalue Bounds
- Linear Programming (LP) and Dual LP bounds
- Semidefinite Programming (SDP) Bounds

Key issues:

- Quality/time tradeoff
- Branching information

2 A Quadratic Programming Bound

$$
\begin{array}{lll}
\text{QAP} & \min & \mathbf{vec}(X)^T (B \otimes A)\, \mathbf{vec}(X) + C \bullet X \\
& \text{s.t.} & Xe = X^T e = e \\
& & X^T X = X X^T = I \\
& & X \geq 0.
\end{array}
$$

$$
\begin{array}{lll}
\text{QPB} & \min & \mathbf{vec}(X)^T Q\, \mathbf{vec}(X) + C \bullet X + \gamma \\
& \text{s.t.} & Xe = X^T e = e \\
& & X \geq 0,
\end{array}
$$

where $Q = (B \otimes A) - (I \otimes S) - (T \otimes I)$.

- Q is p.s.d. on nullspace of equality constraints ($Xe = X^T e = e$).
- S, T, and γ are obtained from spectral decompositions of $V^T AV$, $V^T BV$, where columns of V are an orthonormal basis for nullspace of e^T.
- Objectives of QAP and QPB agree for orthonormal X.
- PB $\leq$ QPB $\leq$ QAP, where PB is projected eigenvalue bound of Hadley, Rendl, Wolkowicz.

Some features of QPB compared to PB:

- For root problem, gap between OPT and bound typically reduced by 10-20%.
- Solution of QP provides dual information that is very useful in devising branching strategies.
- Good evidence that bound increases much faster with branching.
- Can trade accuracy of bound computation for speed.

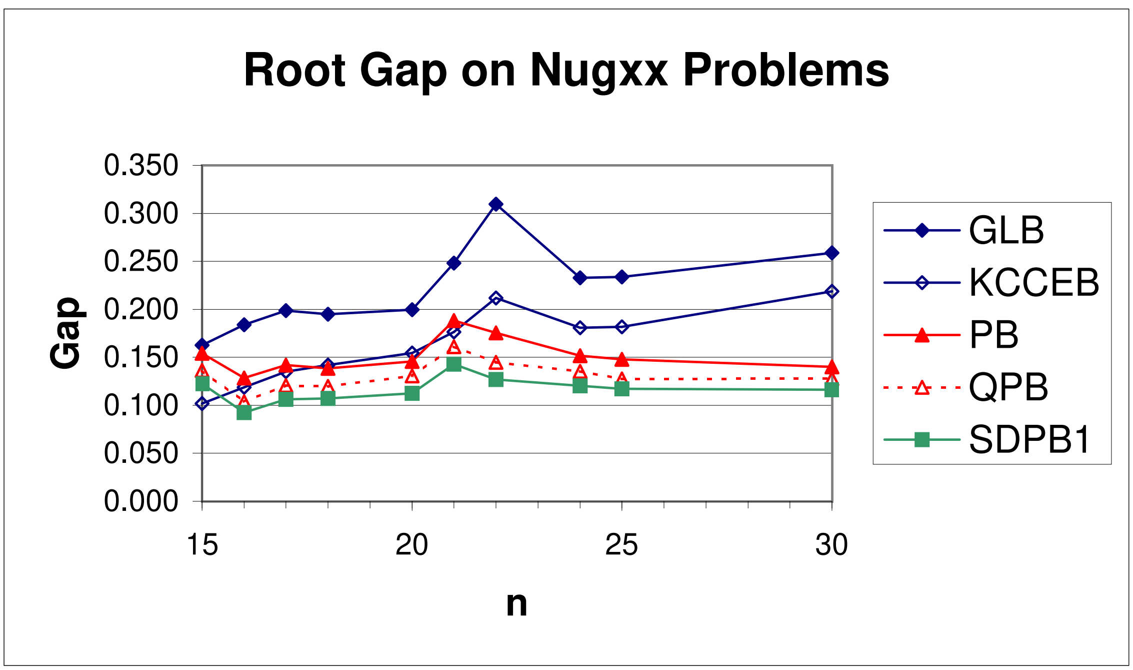

Figure 1: Root gaps for various bounds on nugxx problems

3 Computing QPB

QPB requires solution of a convex QP

- Pivoting methods, Active-set methods, Interior-point methods, ...
- Frank-Wolfe (FW) algorithm (!)

Iterate X_k, gradient matrix $G_k = \nabla f(X_k)$. FW subproblem is then

$$\begin{aligned} \min\ & G_k \bullet X \\ \text{s.t.}\ & Xe = X^T e = e \\ & X \geq 0. \end{aligned}$$

- Low dimension, dense LAP
- Solution X_k^*, dual matrix (reduced costs) U_k.

For any feasible X, have

$$\begin{aligned} f(X) &\geq f(X_k) + G_k \bullet (X - X_k) \\ &= f(X_k) + G_k \bullet (X_k^* - X_k) + G_k \bullet (X - X_k^*) \\ &= z_k + U_k \bullet (X - X_k^*) \\ &= z_k + U_k \bullet X \end{aligned}$$

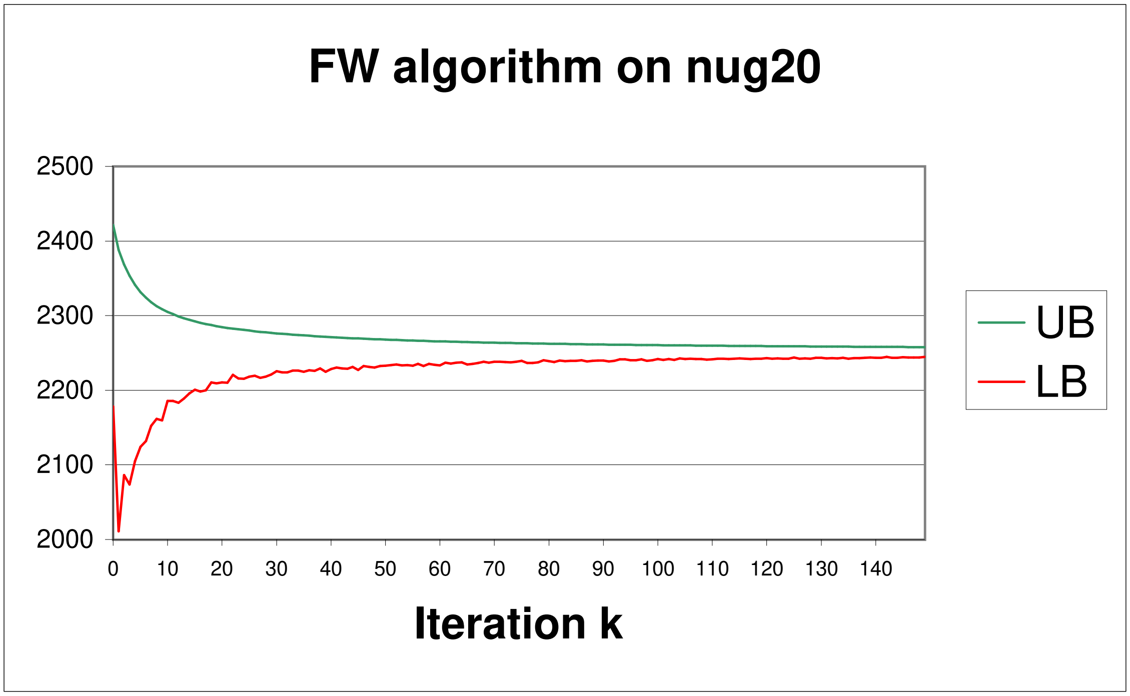

Figure 2: Convergence of Frank-Wolfe algorithm for QPB on nug20

4 The Branch and Bound Algorithm

- Use "polytomic" branching, based on one facility or one location.
- Full symmetry-breaking for grid-based problems.
- Branching rules make use of dual information (U) in attempt to eliminate children, or maximize bounds of children.
- Also consider *strong branching*: compute bound z^{ij} for child problem with $x_{ij} = 1$, use these bounds as basis for final branching decision. Can also use dual info U^{ij} to devise *look-ahead* rule.
- Vary branching rule based on gap, and depth in tree.
- Master-Worker distributed processing implementation. Interesting algorithmic issues include:
  - grainsize/contention reduction
  - clean-up strategy
  - lazy best-first management of master queue

5 Computational Results

First optimal solutions of nug27/28/30, and kra30b, problems.

<u>Solution of nug30</u>:

11,892,208,412 nodes

3.46E8 CPU seconds

653 average workers (1009 peak)

6.9 days wall time

6.9 years equivalent time (HP C3000)

<u>Solution of kra30b</u>:

5,136,036,412 nodes

1.36E8 CPU seconds

417 average workers (780 peak)

3.8 days wall time

2.7 years equivalent time (HP C3000)

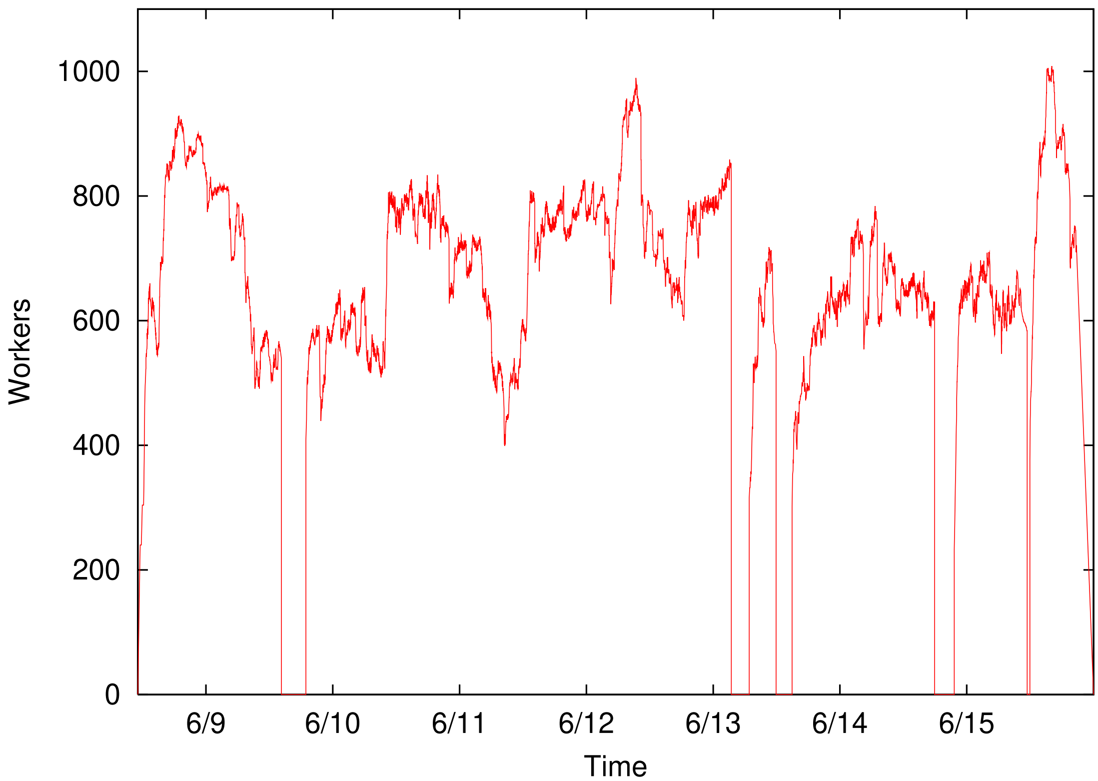

Figure 3: Number of workers on nug30 run

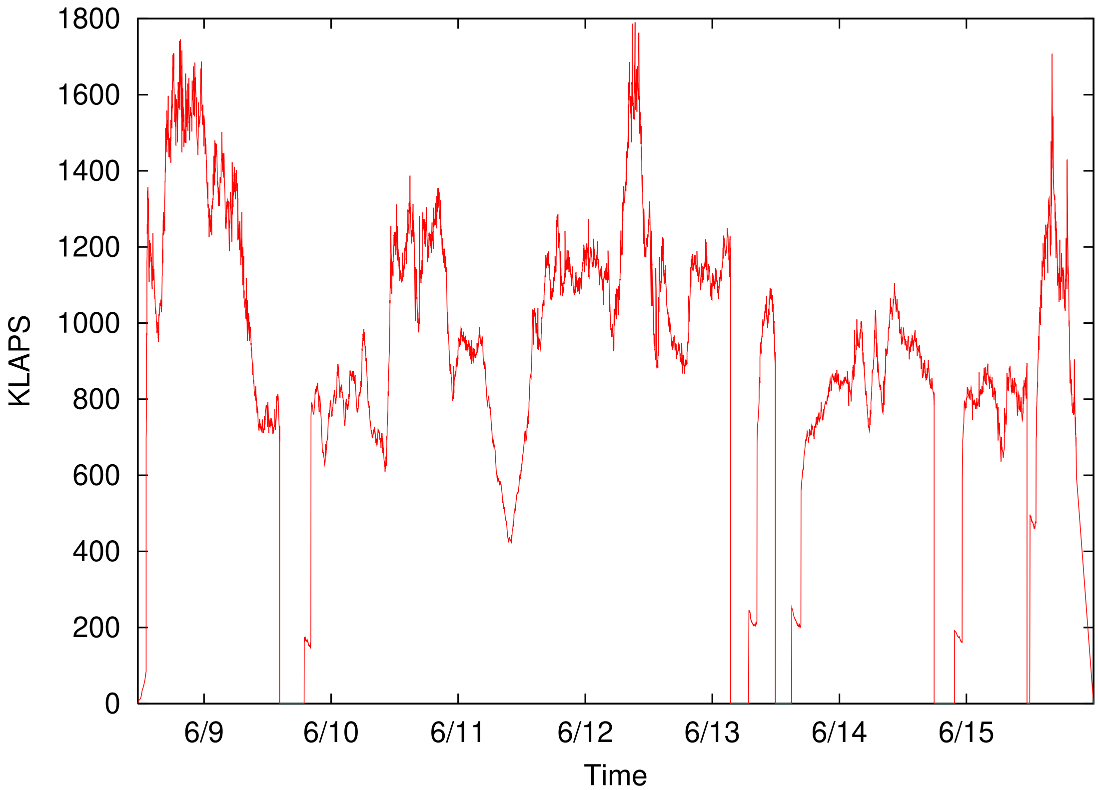

Figure 4: Thousands of LAPs solved per second on nug30 run

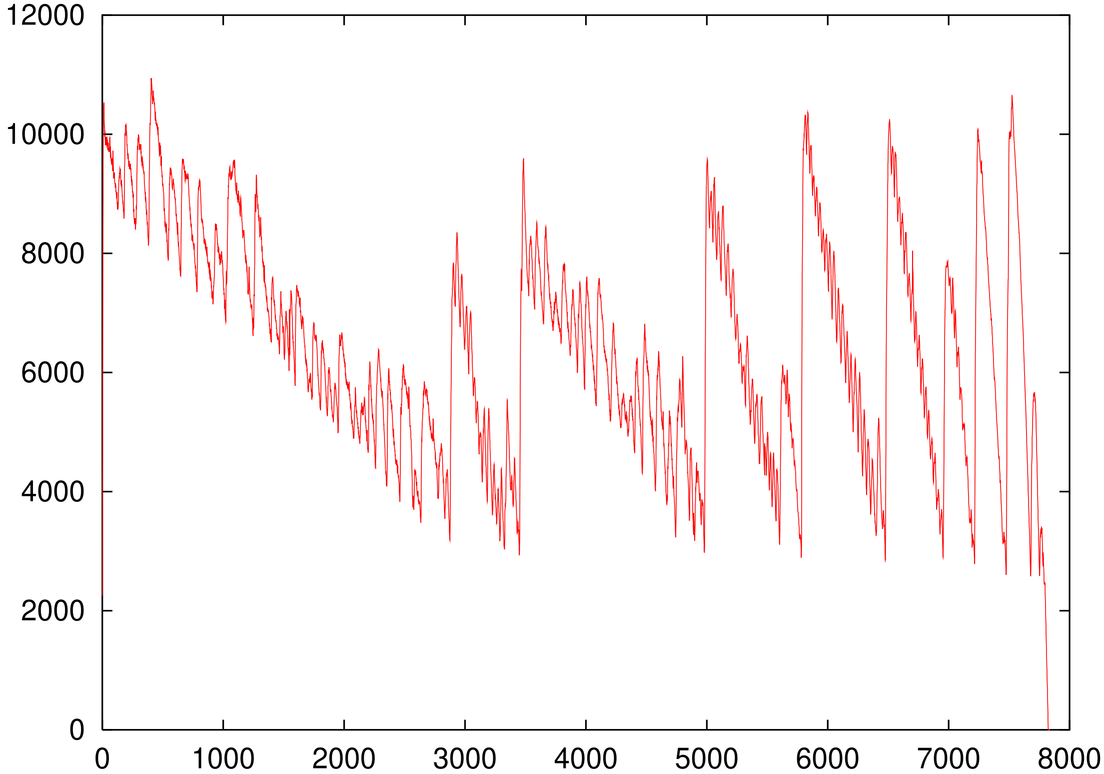

Figure 5: Master queue size vs. number of completed tasks (00), nug30 run

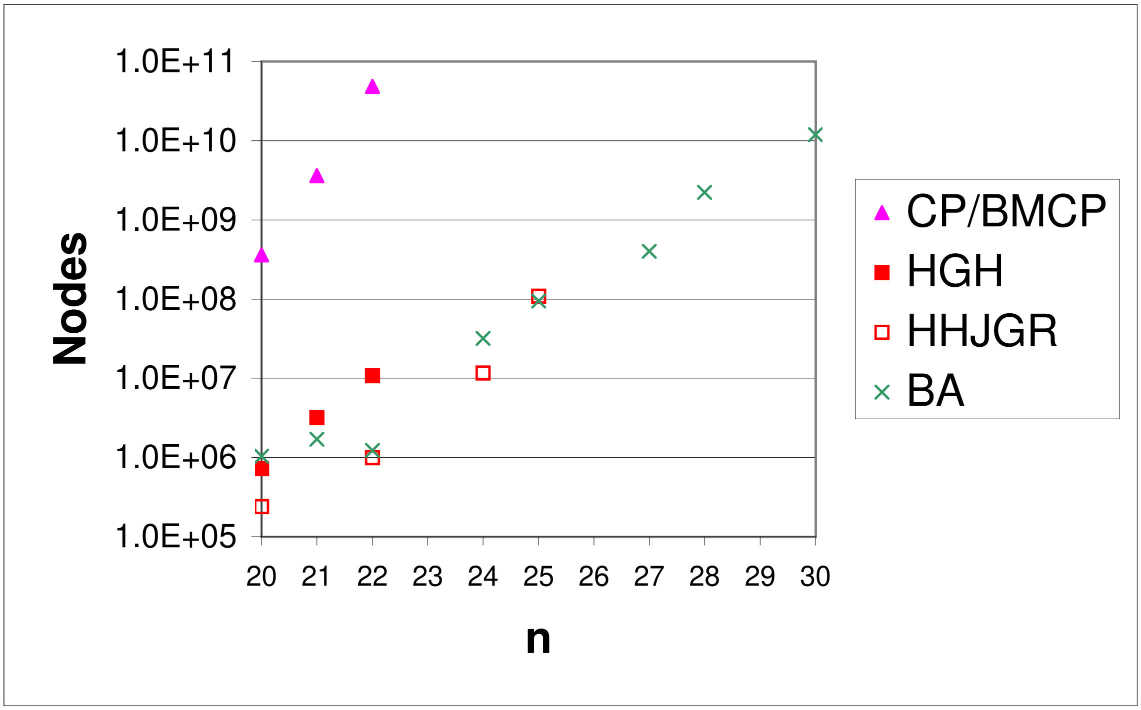

Figure 6: Number of nodes to solve nugxx problems

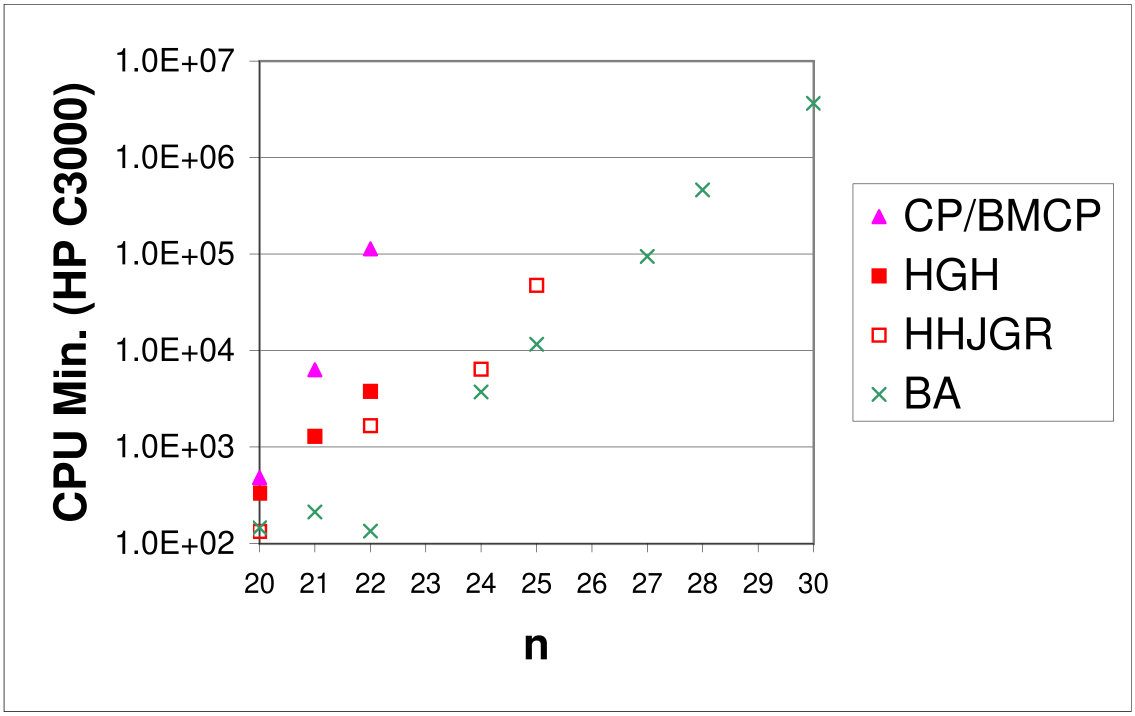

Figure 7: Normalized time to solve nugxx problems

6 Summary

- Consider a nonlinear bound for a nonlinear problem.
- “Newer” isn’t always “better.”
- Metacomputing rules!