Tree Elaboration Strategies in Branch-and-Bound Algorithms for Solving the Quadratic Assignment Problem

Peter M. Hahn, University of Pennsylvania
William L. Hightower, Elon College
Terri Anne Johnson, Elon College
Monique Guignard-Spielberg, University of Pennsylvania
Catherine Roucairol, University of Versailles

- The Quadratic Assignment problem:

  Locate N facilities among N fixed locations, where we know for each pair of facilities i,k a flow of commodities f(i,k) and for each pair of locations j,n a corresponding distance d(j,n).

  Transportation cost between facilities is f(i,k)·d(j,n),.given that i-th facility is assigned to j-th location and k-th facility to n-th location.

  The objective is to find an assignment minimizing the sum of all transportation costs.

- Find an assignment **U** to minimize:

$$R(\mathbf{U}) = \sum_{ijkn} f_{ik}\ d_{jn}\ u_{ij}\ u_{kn}$$

Given N^4 costs $C_{ijkn} \geq 0$ (i, j, k, n = 1, 2,...,N) determine

$$\mathbf{U} = [u_{ab}] \tag{1}$$

called an "assignment", so as to minimize a cost function,

$$R(\mathbf{U}) = \sum_{ijkn} C_{ijkn} \; u_{ij} \; u_{kn} \tag{2}$$

subject to the following constraints on **U**:

$$u_{ij} = 0,1 \quad (i,j=1,2,...,N), \tag{3}$$

$$\sum_{i=1}^{N} u_{ij} = 1 \quad (j = 1,2, \quad ,N), \tag{4}$$

$$\sum_{j=1}^{N} u_{ij} = 1 \quad (i = 1,2, \quad ,N) \tag{5}$$

- Lawler introduced the concept of an N^2 by N^2 solution (or assignment) matrix V that is a Kronecker product of the N x N assignment matrix U with itself.

That is,

$$\mathbf{V} = \mathbf{U} \mathbf{x} \mathbf{U} = \begin{pmatrix} u_{11}\mathbf{U} & u_{12}\mathbf{U} & & u_{1N}\mathbf{U} \\ u_{21}\mathbf{U} & u_{22}\mathbf{U} & & u_{2N}\mathbf{U} \\ \vdots & \vdots & \ddots & \vdots \\ u_{N1}\mathbf{U} & u_{N2}\mathbf{U} & & u_{NN}\mathbf{U} \end{pmatrix} = [v_{ijkn}] \qquad (6)$$

where $v_{ijkn} = u_{ij} u_{kn} = u_{kn} u_{ij} = v_{knij}$ $(i \neq k \text{ and } j \neq n)$ (7)

- (7) says if an element v_{ijkn} is equal to 1 then it has a "complementary element" v_{knij} that is also equal to 1

Example (N=3)

$$\mathbf{V} = \left(\begin{array}{ccc|ccc|ccc} \{0\} & 0 & 0 & 0 & \{0\} & 0 & 0 & 0 & \{1\} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \{1\} & 0 & 0 & 0 & \{0\} & 0 & 0 & 0 & \{0\} \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \{0\} & 0 & 0 & 0 & \{1\} & 0 & 0 & 0 & \{0\} \end{array}\right) \qquad \mathbf{U} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Certain elements of **V** are always zero, specifically

$$v_{ibkb} = 0 \text{ if } i \neq k \qquad (8)$$

$$v_{bjbn} = 0 \text{ if } j \neq n \qquad (9)$$

Historical perspective

- Our branch-and-bound algorithm is based on a dual (ascent) procedure bound (DP) for solving a relaxed linearization of the QAP. (See H&G, Vol. 46, No 6 of Operations Research.)

- DP lower bound calculations result in a reformulation that attempts to solve the QAP from the dual perspective - an essential part of our algorithm. (See Hahn and Grant.)

- Depth-first branching strategy takes advantage of DP bound reformulation. (Will be described.)

- The DP branch-and-bound algorithm solved problem instances up to and including the Nugent 24 an order of magnitude faster than the only competing algorithm with 3 orders of magnitude fewer nodes evaluated. (See Hahn, Grant & Hall, Vol. 108, p629 of European Journal of Operational Research.)

This paper

- In the original algorithm the branch-and-bound tree was elaborated by extending assignments of a given facility to all locations. Of necessity this involves a depth-first branching strategy. Elaboration was done in branch numerical order.

- It was observed that some portions of the tree fathom (are eliminated) more quickly than do others. How could this help to speed up the algorithm?

- Regions having higher bounds fathom more quickly.

- Experimentation showed that the choice of which facility (location) is extended strongly impacts runtime.

- The algorithm was modified to make use of this information making it possible to solve the Nugent 25 and Krarup 30a.

- The DP is an iterative procedure to compute a suboptimal dual solution to a continuous relaxation of a linearization of the QAP:

Linearization of QAP:

LP : Minimize :

$$\sum_{\substack{i,j,k,n \\ i \neq k \\ j \neq n}} C_{ijkn} v_{ijkn} + \sum_{i,j} C_{ijij} v_{ijij}$$

Subject to :

$$\sum_{\substack{k \\ k \neq i}} v_{ijkn} = v_{ijij} = 0,1 \qquad (i,j,n),\ j \neq n \qquad (8)$$

$$\sum_{\substack{n \\ n \neq j}} v_{ijkn} = v_{ijij} = 0,1 \qquad (i,j,k),\ i \neq k \qquad (9)$$

$$v_{knij} = v_{ijkn} \qquad (i,j,k,n),\ k < i,\ j \neq n \qquad (10)$$

$$\sum_{i} v_{ijij} = 1 \qquad (j) \qquad (11)$$

$$\sum_{j} v_{ijij} = 1 \qquad (i) \qquad (12)$$

$$v_{ijkn} \geq 0 \qquad (i,j,k,n),\ k \neq i,\ j \neq n \qquad (13)$$

Continuous Relaxation of Linearization of QAP

CLP1: Minimize :

$$\sum_{\substack{i,j,k,n \\ i>k \\ j\neq n}} \tilde{C}_{ijkn} v_{ijkn} + \sum_{i,j} C_{ijij} v_{ijij}$$

where $\tilde{C} = C_{ijkn} + C_{knij}$

Subject to :

$$0 \leq \sum_{\substack{k \\ k<i}} v_{ijkn} + \sum_{\substack{k \\ k>i}} v_{knij} = v_{ijij} \leq 1 \qquad (i, j, n), j \neq n \qquad (14)$$

$$0 \leq \sum_{\substack{n \\ n\neq j}} v_{ijkn} = v_{ijij} \leq 1 \qquad (i, j, k), i > k \qquad (15)$$

$$0 \leq \sum_{\substack{n \\ n\neq j}} v_{knij} = v_{ijij} \leq 1 \qquad (i, j, k), i < k \qquad (15a)$$

$$\sum_{i} v_{ijij} = 1 \qquad (j) \qquad (16)$$

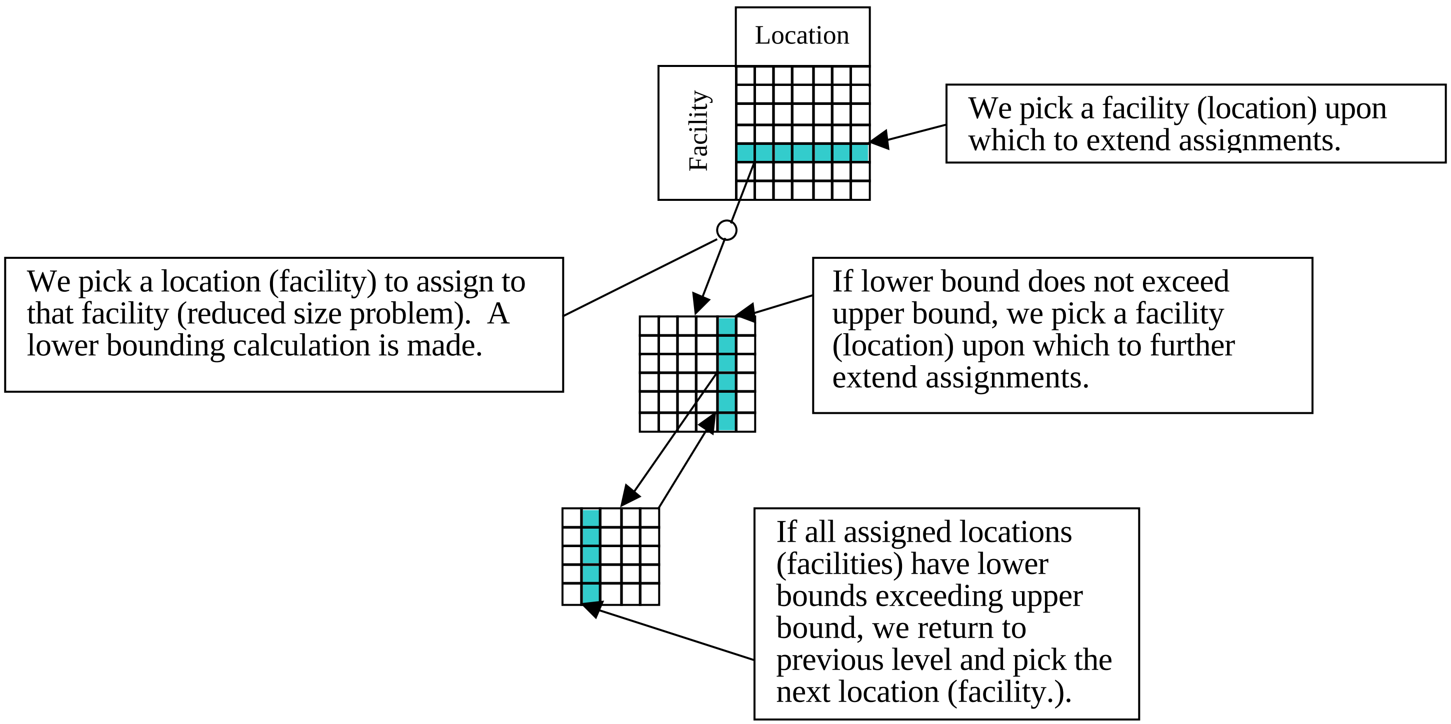
$$\sum_{j} v_{ijij} = 1 \qquad (i) \qquad (17)$$

$$v_{ijkn} \geq 0 \qquad (i, j, k, n), i > k, j \neq n \qquad (18)$$

Branch-and-bound algorithm

- The DP bound is first calculated on the original problem. The result is a reformulated problem with reduced costs C'_{ijkn}.
- Single facility-location assignment is the first level of partial assignment as well as subsequent levels of partial assignment.
- Algorithm proceeds as follows:
  - Partial assignment reduces problem size by 1. The DP bound is calculated for this reduced-size subproblem.
  - This results in a reformulated reduced-size problem.
  - Reduced costs for reduced-size problem are stored and used as the basis for further partial assignments, and so on …

Branch-and-bound tree elaboration



Tree elaboration strategy

(How to pick the facility or location
upon which to extend assignments.)

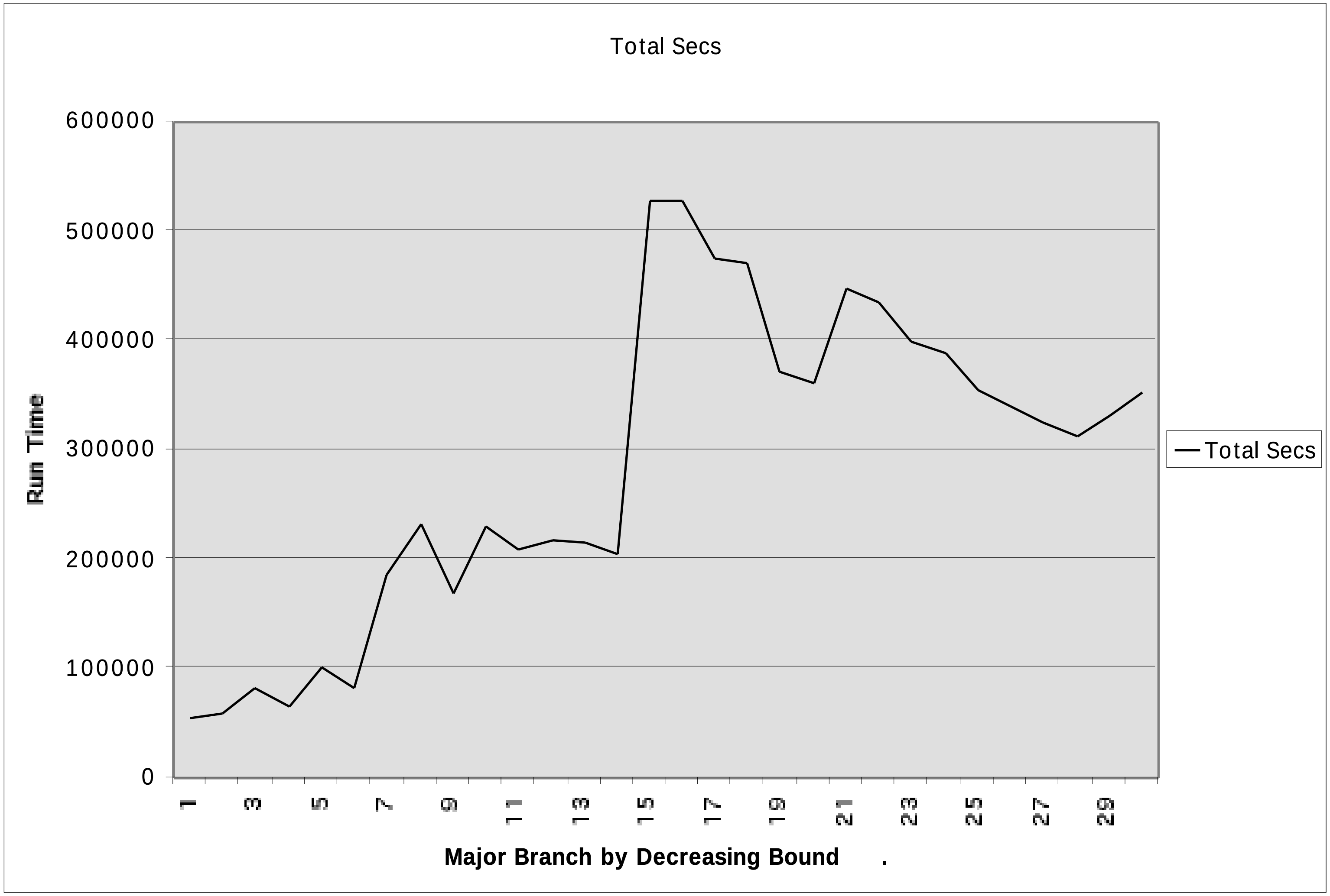
- Preliminary bounding calculations are made for all facility-location assignments.
- Experiments indicate that the highest average of bounds along a row or a column indicates facility (row) or location (column) to extend.
- Better (more computationally intensive) bounds must be utilized near root. Less accurate bounds further into tree.

Tree Elaboration Experimental Results				
QAP Instance	Facility (location) choice by numerical order		Facility (location) choice by highest average bound	
	Number of Nodes	Runtime in minutes	Number of Nodes	Runtime in minutes
Hadley 18	197,487	140.3	53,224	18.2
Nugent 20	724,289	550.8	239,449	268.1
Nugent 22	10,768,366	20,545.4	988,302	3,326.8
Nugent 24	49,542,338	55,101.4	11,674,955	12,875.4
Nugent 25	N/A	N/A	108,738,131	94,980.3
Krarup 30a	N/A	N/A	29,764,589	141,958.4* (98.6 days)

N/A = Not Available

* Latest program (~ 1.5 to 4 times faster than previous versions).

Krarup 30a Time per First Level Branch



IMPORTANT POINTS

- **DP bounds are more effective in branch-and-bound enumeration if reformulation is used to reduce costs at root and throughout the tree.**
- **Depth-first branching strategy is required to take advantage of DP reformulation.**
- **QAP elaboration is structured (extending assignments of a facility to all locations or vice-versa).**
- **Which facility (location) is extended strongly impacts runtime (4 to 7 times faster).**
- **Preliminary bounding directs the choice of facility (location). Larger QAPs need better bounding.**
- **Highest average preliminary bound selects the best facility (location) upon which to extend.**

Krarup 30a Results

- **Optimum objective function value: 88,900**
- **Major branches containing solutions: 4**
- **Number of optimum solutions listed by program: 133 out of 256 known. The remainder is found implicitly by the algorithm but is not explicitly listed by the program. (See "Iterated local search", by Thomas Stützle for the complete list.)**
- **Sample solutions:**

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

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