A hybrid ACO approach to the Matrix Bandwidth Minimization Problem

Camelia Pintea
G. Coşbuc N. College
Cluj-Napoca

Gloria Cerasela Crişan
V. Alecsandri University
Bacău

Camelia Chira
Babes-Bolyai University
Cluj-Napoca

R O M A N I A

Real World HAIS Applications and Data Uncertainty, HAIS 2010, San Sebastian

The problem

Bandwidth Minimization Problem (BMP)

Finding an equivalent matrix in which all the nonzero entries would lay within a narrow band near the main diagonal.

Bandwidth Minimization Problem for Graphs

Finding a labeling for the vertices where the maximum absolute difference between labels of each pair of connected vertices is minimum.

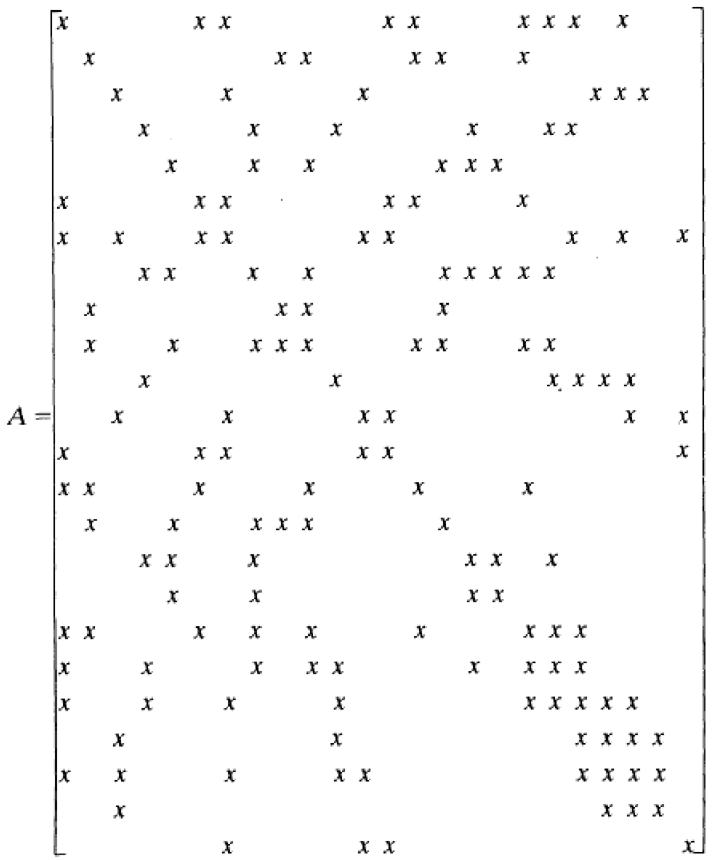
Global Optimization Applications

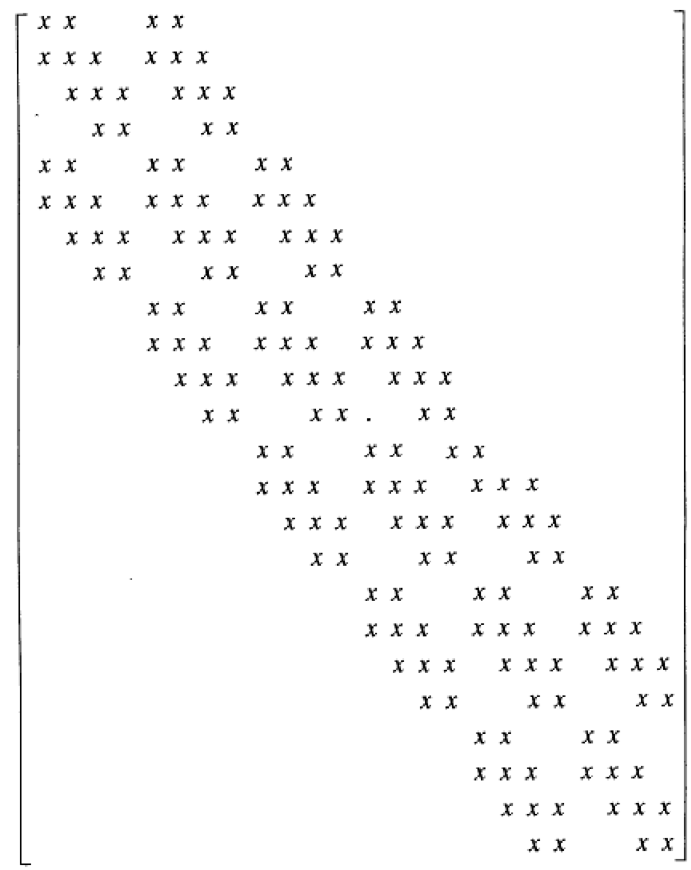
Preprocessing the coefficient matrix for solving the system of equations, finite element methods for partial differential equations

Largescale power transmission systems, circuit design, hypertext layout, numerical geophysics

The problem

- Arises in Systems of linear equations:





from:

Norman E. Gibbs; William G. Poole, Jr.; Paul K. Stockmeyer - An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix, *SIAM Journal on Numerical Analysis*, Vol. 13, No. 2. (1976), pp. 236-250

BMP

Find a permutation of the rows and the columns of matrix A that keeps all the non-zero elements in a band that is as close as possible to the main diagonal

Matrix: $A=\{a_{ij}\}_{nxn}$

Bandwidth: $\beta = \max_{a_{ij} \neq 0} | i - j |$ → **MIN**

Bandwidth Minimization Problem for Graphs (BMPG)

- *G=(V,E)*

  *c*onsider a vertex for each row (column) and an edge if $a_{ij} \neq 0$ or $a_{ji} \neq 0$.

- Find a labeling *of the vertices that minimizes the maximum* difference between labels of adjacent vertices

BMPG

G(V, E) a graph with *n* vertices

the set of all vertex orderings:

$$\Phi(G) = \{\varphi : V \to \{1, 2, \ldots n\}, \varphi \text{ } is \text{ } a \text{ } bijection\}$$

the bandwidth:

$$B(G) = \min_{\varphi \in \Phi(G)} \left(\max_{(i\ j) \in E} \left(\left| \varphi(i) - \varphi(j) \right| \right) \right)$$

Approaches to BMPG

The problem is $\mathcal{NP}$- complete.

- Specific classes of instances could be exactly solved in polynomial time (chain graphs).
- Specific classes of instances have polynomial time approximation algorithms (caterpillars).
- Heuristics come from early 70s intense investigations.

Approaches

Successful heuristic algorithm.

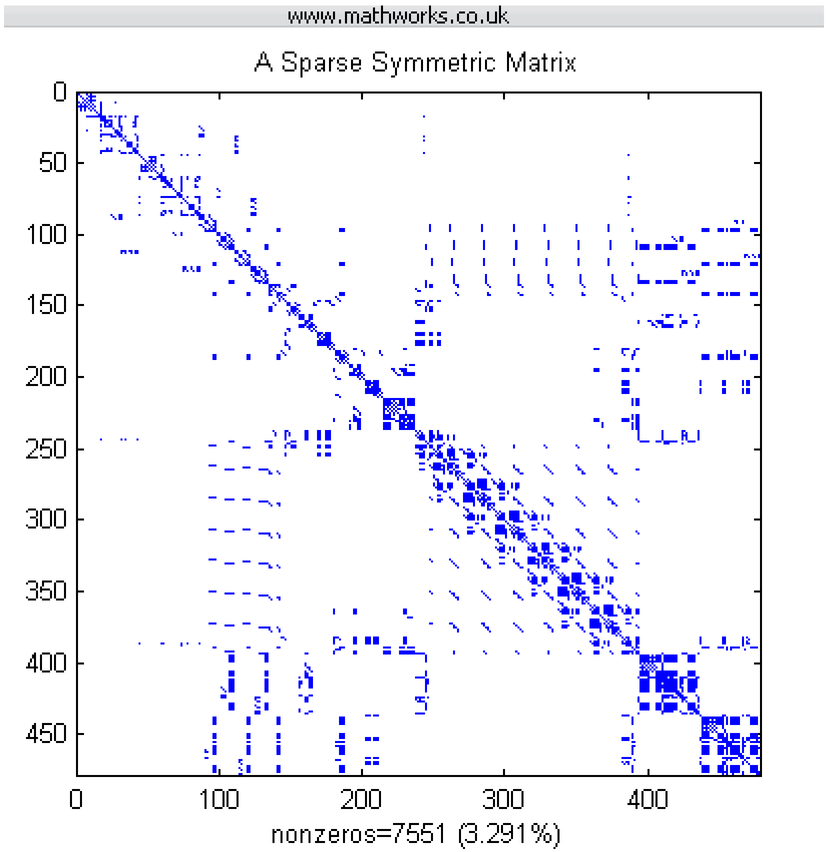
Cuthill-McKee (1969)

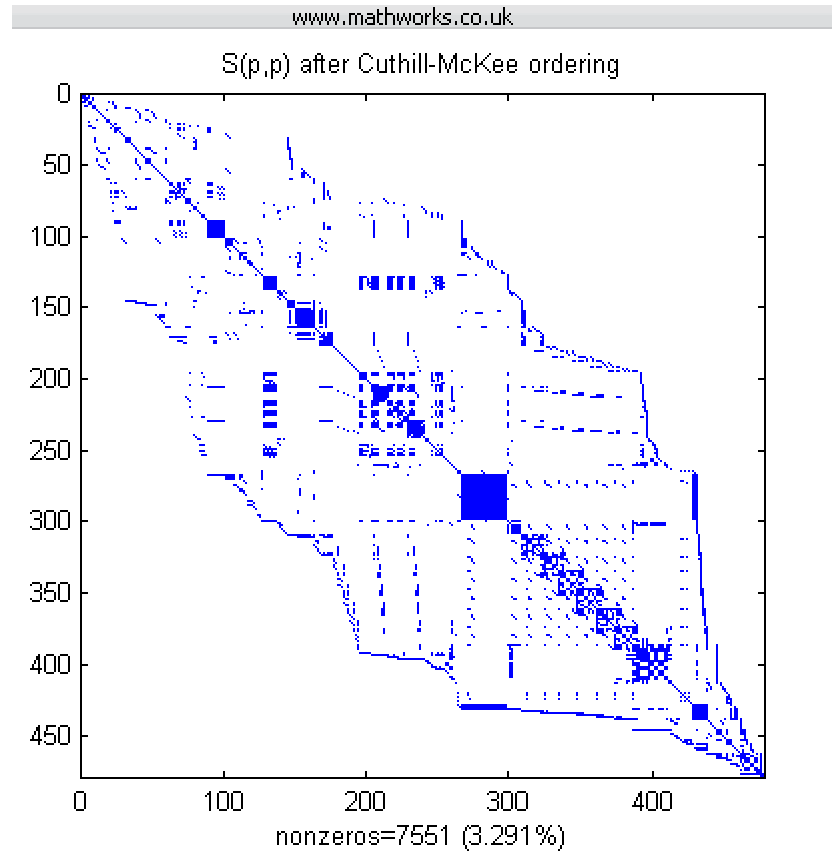
- Start from a (randomly chosen) vertex.
- Number the vertices according to a breadth-first traversal where neighboring vertices are visited in order of increasing degree.

- Simple, stable, reliable, basis for many current software collections.

Cuthill-McKee in MATLAB 7.10

A sparse symmetric positive definite matrix derived from a portion of the Harwell-Boeing test matrix "west0479“; the SYMRCM command uses the reverse Cuthill-McKee algorithm to move all nonzero elements closer to the diagonal.





from:
http://www.mathworks.co.uk/products/matlab/demos.html?file=/products/demos/shipping/matlab/sparsity.html

ACO Approach to BMPG

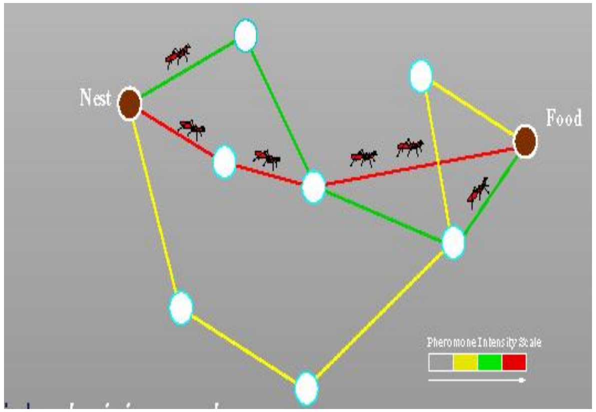
Ant Colony System.

A set of artificial ants cooperate using an indirect form of communication mediated by pheromone they deposit while building solutions.

Ant Colony System (ACS)

m ants are initially positioned on *n* nodes chosen according to some initialization rule (e.g., randomly).

- Stronger pheromone trails are preferred
- Most promising tours build



up more pheromone in time

- State transition rule
- Local and global pheromone update rules
- The result of an ACS algorithm the shortest tour found

Proposed Approach to BMPG

- The initial phase
  - computes the current matrix bandwidth and sets the parameters values
- The construction phase
  - Start: all ants placed in the first level and repeatedly make pseudorandom choices from the available neighbors
  - local update rule after each step
  - global pheromone update rule at the end of the phase
- The final phase
  - writing the best solution

Proposed Hybrid Ant Colony System for BMPG

begin
I. Initialization: computes the current matrix bandwidth; initialize pheromone trails; sets the parameters values;
II. while (maximum number of iterations not reached) **do**
    Swap Procedure
    while (maximum number of ants not reached) **do**
        build a partial solution using ACS
        apply a local pheromone update rule
        Swap Procedure
        apply a global pheromone update rule
    end while
end while
III. write the best solution
end

Solution: one-dimensional array that stores the permutation of $V = \{1, 2, ..., n\}$

Local Search

hACS

PSwap Procedure
find the maximum and minimum degrees
for all indices x with the maximum degree
    randomly select y, an unvisited node with a minimum degree
    SWAP(x,y)
end for

hMACS

MPSwap Procedure
find the maximum and minimum degrees
for all indices x with the maximum degree
    select y, an unvisited node with a minimum degree such as
        the matrix bandwidth decreases
    SWAP(x,y)
end for

Numerical Experiments

Instances
nine symmetric Euclidean instances
from Harwell-Boeing
sparse matrix collection

No	Instance	Euclidean Characteristics
1.	**can___24**	**24 24 92**
2.	**can___61**	**61 61 309**
3.	**can___62**	**62 62 140**
4.	**can___73**	**73 73 225**
5.	**can___96**	**96 96 432**
6.	**can__187**	**187 187 839**
7.	**can__229**	**229 229 1003**
8.	**can__256**	**256 256 1586**
9.	**can__268**	**268 268 1675**

Parameters
10 ants, 10 iterations, q_0 = 0.95, α =1, β = 2 , ρ = 0.0001, τ_0 = 0.1

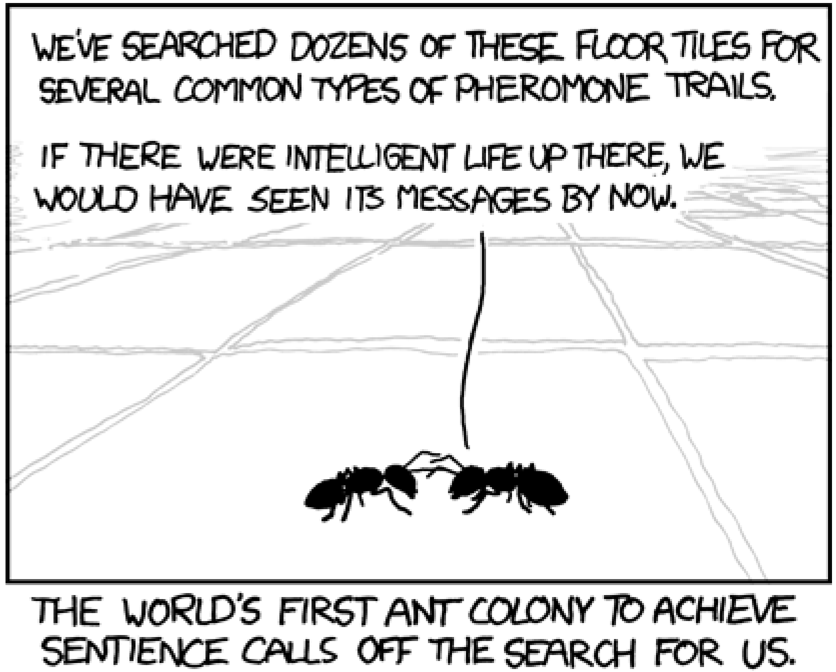
Platform
Java sequential application
AMD 2600 computer with 1024 MB memory and 1.9 GHz CPU clock.

Results

No	Instance	ACS				hACS				hMACS			
		MIN	#	AVG	AVGT	MIN	#	AVG	AVGT	MIN	#	AVG	AVGT
1.	**can___24**	17	2	18.6	1.33	14	7	14.80	0.55	**11**	1	**12.6**	0.6
2.	**can___61**	47	1	49.8	2.23	43	8	43.60	1.22	**42**	6	**42.8**	1.18
3.	**can___62**	39	1	45.75	1.95	20	3	22.00	0.74	**12**	1	**16.35**	0.9
4.	**can___73**	37	6	38.1	1.69	28	1	31.30	1.15	**22**	1	**26.7**	1.3
5.	**can___96**	31	20	31	2.07	**17**	2	27.45	1.56	**17**	**14**	**19.7**	1.68
6.	**can__187**	63	20	63	3.2	63	20	63.00	4.52	**33**	**4**	**37.25**	5.2
7.	**can__229**	163	2	168.35	3.84	**120**	**1**	**131.25**	11.21	**120**	**1**	132.65	9.9
8.	**can__256**	241	1	249.45	5.02	**148**	**1**	**164.50**	44.48	189	1	197.45	26.05
9.	**can__268**	234	1	241.1	5.13	**165**	**1**	**193.80**	30.21	210	1	217.75	21.9

Conclusions

- Procedure **MPSwap** (used in hMACS) performs better on small instances, while **PSwap** (in hACS) is better for larger instances
- The average execution time is better for the modified local search procedure
- A balance between the quality of the solution and the time needed is also observed



from: http://xkcd.com/638/

Thank you!