

# A Genetic Programming Approach to the Matrix Bandwidth-Minimization Problem

Behrooz Koohestani and Riccardo Poli

School of Computer Science and Electronic Engineering,  
University of Essex, CO4 3SQ, UK  
{bkooshe, rpoli}@essex.ac.uk

**Abstract.** The bandwidth of a sparse matrix is the distance from the main diagonal beyond which all elements of the matrix are zero. The bandwidth minimisation problem for a matrix consists of finding the permutation of rows and columns of the matrix which ensures that the non-zero elements are located in as narrow a band as possible along the main diagonal. This problem, which is known to be NP-complete, can also be formulated as a vertex labelling problem for a graph whose edges represent the non-zero elements of the matrix. In this paper, a Genetic Programming approach is proposed and tested against two of the best-known and widely used bandwidth reduction algorithms. Results have been extremely encouraging.

**Keywords:** Bandwidth Minimization Problem; Genetic Programming; Graph Labelling; Sparse Matrices; Combinatorial Optimisation.

## 1 Background

The Bandwidth Minimization Problem (BMP) is a very well-known problem, familiar to applied mathematicians and arising in many applications in science and engineering [18]. BMP consists of finding the permutation of rows and columns of a matrix which ensures that the non-zero elements are located in as narrow a band as possible along the main diagonal. One of the most common applications of bandwidth-minimisation algorithms arises from the need to efficiently solve large systems of equations [19]. In such a scenario, more efficient solutions are obtained if the rows and columns of the matrix representing the set of equations can be permuted in such a way that the bandwidth of the matrix is minimized [19]. BMP has also connections with a wide range of other problems, including: finite element analysis of mechanical systems, large scale power transmission systems, circuit design, VLSI design, data storage, chemical kinetics, network survivability, numerical geophysics, industrial electromagnetics, saving large hypertext media and topology compression of road networks.

The BMP is NP-complete [18] and, hence, it is highly unlikely that there exists an algorithm which finds the minimum bandwidth of a matrix in polynomial time. It has also been proved that the BMP is NP-complete even for trees with

a maximum degree of three and only in very special cases it is possible to find the optimal ordering in polynomial time [5].

The first direct method for the BMP was proposed by Harary [9]. Cuthill and McKee [3] introduced the first heuristic approach to the problem. Their method is still one of the most important and widely used methods to (approximately) solve the problem. In this method, the nodes in the graph representation of a matrix are partitioned into equivalence classes based on their distance from a given root node. The partition is known as *level structure* for the given node. In Cuthill and McKee's algorithm, the root node for the level structure is chosen from the peripheral nodes in the graph (i.e., nodes which have the highest distance to any other node in the graph). The permutation chosen to reduce the bandwidth of the matrix is then simply obtained by visiting the nodes in the level structure in increasing-distance order. A few years later, Gibbs et al. [7] proposed an algorithm, known as GPS (which stands for "Gibbs, Poole and Stockmeyer"), that makes more extensive use of level structures. The algorithm is substantially faster than the Cuthill and McKee algorithm, and it can occasionally outperform it. However, the algorithm is significantly more complex to implement. More recently, Barnard *et al.* [1] have proposed the use of spectral analysis of the Laplacian matrix associated with the graph representing the non-zero elements in a sparse matrix as an effective method for the reduction of the envelope of a graph. In particular, the method permutes a matrix based on the eigenvector associated with the first non-zero eigenvalue of the Laplacian matrix. While the envelope is only indirectly related to the bandwidth of a matrix, this algorithm is very effective at reducing it. Further information on these and other classic methods for the BMP can be found in [2, 8].

Recently meta-heuristic approaches have been tested to see if they can be viable alternatives to solve the BMP. For example, Tabu search was employed by Marti *et al.* [17] while Lim *et al.* [14, 16] used a hybrid between genetic algorithms and hill-climbing to solve this problem. Lim *et al.* also introduced two other hybrid algorithms to solve BMP: one combining ant colony optimization with hill-climbing [12] and one combining particle swarm optimization with hill-climbing [13]. Recently also simulated annealing has been used to attack the problem [21].

In this paper, a Genetic Programming (GP) [11, 20] approach is proposed and tested against a high-performance version of the Cuthill and McKee algorithm [3] and a version of the spectral analysis proposed by Barnard *et al.* [1]. To the best of our knowledge, no prior attempt to use GP to solve BMP has been reported in the literature. Despite this lack of prior art, the results of our first investigations have been very encouraging and suggest this is a fertile area for further research.

The paper is organised as follows. In Sect. 2, we provide two equivalent formulations of the bandwidth-minimisation problem. In Sect. 3, we describe our GP system for the solution of BMP. In Sect. 4, we report the results of our experiments. Finally, in Sect. 5, we provide some conclusions.

## 2 Graph-Theoretic and Matrix Formulations of the BMP

Let  $G = (V, E)$  be a finite undirected graph, such that  $V$  is the set of vertices,  $E$  is the set of edges and  $f : V \rightarrow \{1, \dots, n\}$  is a labeling of its nodes where  $n = |V|$ , then the *bandwidth* of  $G$  under  $f$  can be defined as:

$$B_f(G) = \max_{(u,v) \in E} |f(u) - f(v)| \quad , \quad (1)$$

i.e., as the maximum absolute difference between the labels of the adjacent nodes (i.e., nodes connected by an edge). The *bandwidth minimisation problem* consists in finding a labeling  $f$  which minimises  $B_f(G)$  while the easier *bandwidth reduction problem* requires finding any labeling which reduces  $B_f(G)$ . Since there are  $n!$  possible labellings for a graph with  $n$  vertices, it stands to reason that the BMP is, in general, a very difficult combinatorial optimisation problem.

The BMP can also be stated in the context of matrices. If  $A = [a_{ij}]_{n \times n}$  is a sparse matrix, its bandwidth is defined as

$$B(A) = \max_{(i,j):a_{ij} \neq 0} |i - j| \quad . \quad (2)$$

The matrix bandwidth minimisation problem consists of finding a permutation of rows and columns which brings all non-zero elements of  $A$  into the smallest possible band around the diagonal. More formally, if  $\sigma$  is a permutation of  $(1, 2, \dots, n)$ , and  $A_\sigma$  is the matrix obtained by permuting the rows and columns of  $A$  according to  $\sigma$  (i.e.,  $A_\sigma = [a_{\sigma_i \sigma_j}]$ ), then the problem can be formulated as

$$\min_{\sigma} B(A_\sigma) \quad . \quad (3)$$

An important concept related to the bandwidth is the notion of *profile*. Given a matrix  $A$ , its profile is:

$$P(A) = \sum_{i=1}^n \max_{j:j < i, a_{ij} \neq 0} (i - j) \quad . \quad (4)$$

Naturally, also the profile is influenced by permutations of  $A$ .

## 3 GP for BMP

We used a tree-based GP system implemented in C# with some additional decoding steps required by BMP and our particular choice of representation for solutions. A high-level description of the operations of the system is given in Algorithm 1. Below, the elements of the system are described in detail.

### 3.1 GP Setup

Individuals in our GP system are tree-like expressions (which, for efficiency reasons, are internally stored as linear arrays using a flattened representation for trees).

**Algorithm 1.** Pseudo-code of our GP system for BMP

- 
- 1: Randomly generate an initial population of programs from the available primitives.
  - 2: **repeat**
  - 3:   Execute each program in the population.
  - 4:   Create the permutation represented by each program.
  - 5:   Apply each permutation to the adjacency list of the initial graph/matrix and generate new adjacency lists.
  - 6:   Compute the bandwidth and profile for the adjacency lists obtained in step 5.
  - 7:   Calculate the fitness value of each program in the population using Equation (5).
  
  - 8:   Perform selection to choose individual program(s) from the population based on fitness to participate in genetic operations.
  - 9:   Create a new generation of individual program(s) by applying genetic operations with specified probabilities.
  - 10: **until** the maximum number of generations is reached.
  - 11: **return** the permutation represented by the best program in the last generation.
- 

We used the function set  $\{+, -, \times, \%(protected\ division), \text{Sin}, \text{Abs}\}$  and the terminal set  $\{X, Y, Z, c_1, \dots, c_{100}\}$  where  $X, Y$  and  $Z$  are floating-point input variables whose role will be described later and  $c_1, \dots, c_{100}$  are uniformly-distributed random constants with floating-point values in the interval  $[-5.0, +5.0]$ .

The initial population was generated randomly using a modified version of the ramped half-and-half method [11, 20]. In our system, during the process of tree initialisation, terminals are not chosen with equal probability from the terminal set. Instead, we artificially increase the chance of selecting the variables  $X, Y$  and  $Z$  ensuring that every tree contains at least one variable.<sup>1</sup>

Tournament selection was used to choose individual program(s) from the population based on fitness to participate in genetic operations. New individual programs were created by applying the genetic operations of reproduction, sub-tree crossover and point mutation with specified probabilities. These and other parameters of the runs are presented in Table 1. We used elitism to ensure the best individual in one generation was transferred unaltered to the next. The termination criterion used was based on the predetermined maximum number of generations to be run. The permutation extracted from the best program tree appearing in the last generation was designated as the final result of a run.

### 3.2 The Generation of Permutations

The GP system described in the previous section is effectively quite similar to a system one might want to use to solve symbolic regression problems. One may wonder then how we transform program trees into the permutations which are needed to actually solve the BMP. We do this using a simple trick: we interpret

---

<sup>1</sup> Trees without variables are a problem for our system because they produce a constant output and, thus, they represent the permutation  $\sigma = (1, 2, \dots, n)$ . Such a permutation is useless since it produces no change in bandwidth.

**Table 1.** Parameters used in our GP experiments

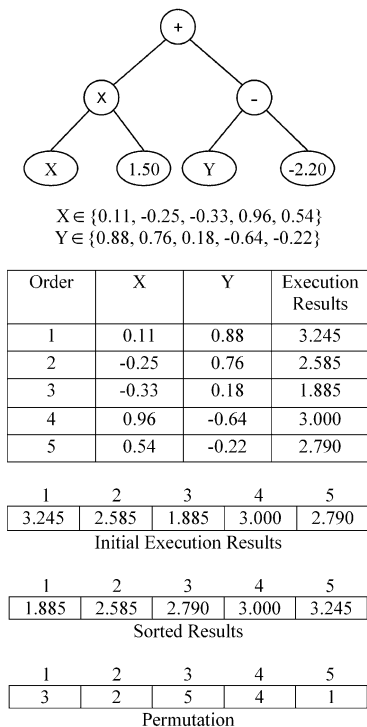
Parameter	Value
Maximum Number of Generations	100
Maximum Length of any GP Program	500
Maximum Depth of Initial Programs	3
Population Size	1000
Tournament Size	5
Elitism Rate	0.1%
Reproduction Rate	0.9%
Crossover Rate	70%
Mutation Rate	29%
Mutation Per Node	0.05%

the outputs produced by a program tree when executed over a set of fitness cases as a permutation. In this section, we explain this decoding process. The process is exemplified for a  $5 \times 5$  array (or a 5-node graph) in Fig. 1.

For each program tree, the interpreter is called  $n$  times where  $n$  is the number of nodes of a given graph or the dimension of a given matrix. In other words, the number of fitness cases used depends on the dimension of the permutation to be generated. Each call of the interpreter executes the selected program with respect to the different values of the independent variables  $X$ ,  $Y$  and  $Z$ .

The fitness cases in our system are somehow peculiar. Firstly, no target output is specified in the fitness cases. In addition, the values for  $X$ ,  $Y$  and  $Z$  for each fitness case are carefully chosen during the initialisation of the system in such a way as to maximise the chance that they represent good permutations. In particular, following the ideas of [1] (see Sect. 1) the  $X$  values were set to be the components of the eigenvector associated with the first non-zero eigenvalue of the Laplacian matrix associated with the matrix to be optimised. The values of  $Y$  and  $Z$  were obtained by using the notion of level structure, which is at the basis of other high-performance BMP solvers, namely: the GPS algorithm and the reverse Cuthill-McKee algorithm which construct the level structure of peripheral nodes. Since the identification of peripheral nodes is expensive, we used the approach followed in [16] which picked random nodes in a graph and built permutations using their associated level structures. In other words, to initialise  $Y$  and  $Z$  we picked two random nodes and built their level structures. By traversing them we constructed two permutations. Then we converted these permutations into two floating point arrays such that, if sorted, they would produce those permutations back (see below). Finally, we used such arrays to provide the fitness case inputs associated with the  $Y$  and  $Z$  variables. It should be noted that the time required for the process of the initialization of the independent variables  $X$ ,  $Y$  and  $Z$  is negligible compared to the total computational time.

The outputs obtained from each execution of the given tree were stored in a one dimensional array. This array was then sorted in ascending order while also recording the position that each element originally had in the unsorted



**Fig. 1.** The process of generating a permutation from a program tree

array. Reading such positions sequentially from the sorted array produced the permutation associated with the original tree.

### 3.3 The Calculation of Fitness Values

After the creation of the permutation representing a program, this permutation is applied to the adjacency list of the initial graph (or to a sparse matrix) in order to generate a new adjacency list which is used for calculating the bandwidth via (1) or (2).

The standard objective function for the BMP is simply that given in (2) and its calculation is straightforward. However, although this is precisely the value that needs to be minimized, it is not an ideal fitness function for a metaheuristic search. The main reason for this is that there may be many candidate solutions which have formally the same bandwidth, but which are quite different, with some being much closer to better solutions than others. A more effective approach is to use a fitness function which incorporates information about how close the candidate solution is to better areas of the search space.

As suggested in [10], the incorporation of the profile (see Sect. 2) into the fitness function used for the BMP can help capture this information. For these

reasons, in this work we used the product of bandwidth and profile as our fitness function, i.e.,

$$f(p) = B(A_{\sigma(p)}) \times P(A_{\sigma(p)}) \tag{5}$$

where  $\sigma(p)$  is the permutation associated with program tree  $p$ .

## 4 Experimental Results

In the experiments conducted in this study, we used a set of 15 instances from the well-known Harwell-Boeing sparse matrix collection (available from <http://math.nist.gov/MatrixMarket/data/Harwell-Boeing>). This collection includes benchmark matrices arising from problems in linear systems, least squares and eigenvalue calculations.

In order to evaluate the performance of the proposed GP method, we compared it against two high-performance methods. Firstly, we used the Reverse Cuthill-McKee (RCM) algorithm contained in the MATLAB library (RCMM). This is based closely on the SPARSPAK implementation described by George and Liu [6]. As indicated in Sect. 1, the algorithm first finds a (pseudo) peripheral vertex of the graph of the matrix. It then generates a level structure by breadth-first search and orders the vertices by decreasing distance from the chosen vertex. This algorithm is still one of the best and most widely used methods for the BMP. Indeed, the results reported in [4] and [15] indicate that RCMM

**Table 2.** Comparison of our GP approach with the RCMM and Spectral algorithms. Best results are shown in boldface, while ties are shown in italics.

Harwell-Boeing Matrix	Dimension	RCMM $B_f(G)$	Spectral $B_f(G)$	GP Approach Mean $B_f(G)$ (10 runs)
ash85	85 × 85	13	17	<b>12.0</b>
bcsfwr01	39 × 39	5	11	<i>5.0</i>
bcsfwr02	49 × 49	13	12	<b>10.4</b>
bcsstk01	48 × 48	27	<b>20</b>	24.5
can_24	24 × 24	7	6	<b>5.6</b>
can_61	61 × 61	19	14	<b>13.4</b>
can_62	62 × 62	9	10	<b>8.0</b>
can_73	73 × 73	27	27	<b>23.3</b>
can_96	96 × 96	23	18	<b>15.6</b>
dwt_59	59 × 59	8	10	<b>7.2</b>
dwt_66	66 × 66	3	3	<i>3.0</i>
dwt_72	72 × 72	7	12	<b>6.0</b>
dwt_87	87 × 87	17	19	<b>13.3</b>
lap_25	25 × 25	9	7	<b>6.0</b>
nos4	100 × 100	12	<b>11</b>	11.3
Mean of $B_f(G)$ values		13.27	13.13	10.97
Standard deviation		7.73	5.98	6.16
Standard error of the mean		2.00	1.54	1.59

Table 3. Run-by-run results of our GP system for BMP

Run	Benchmark Matrix														
	ash85	bcpwr01	bcpwr02	bcsstk01	can_24	can_61	can_62	can_73	can_96	dwt_59	dwt_66	dwt_72	dwt_87	lap_25	nos4
1	11	5	10	24	6	13	9	23	16	8	3	8	13	6	12
2	12	5	10	26	5	14	8	23	16	7	3	8	13	6	11
3	12	5	10	25	6	13	8	23	17	7	3	8	14	6	12
4	12	5	12	24	7	13	8	23	15	7	3	8	13	6	11
5	12	5	10	23	7	13	8	23	16	7	3	8	14	6	12
6	13	5	10	25	5	14	7	25	15	7	3	8	13	6	11
7	11	5	10	25	5	14	8	23	16	7	3	8	13	6	11
8	14	5	10	25	5	13	8	24	15	7	3	9	13	6	11
9	12	5	10	25	5	14	8	23	15	8	3	8	14	6	11
10	11	5	12	23	5	13	8	23	15	7	3	8	13	6	11
Best	11	5	10	23	5	13	7	23	15	7	3	8	13	6	11
Worst	14	5	12	26	7	14	9	25	17	8	3	9	14	6	12
Mean	12.0	5.0	10.4	24.5	5.6	13.4	8.0	23.3	15.6	7.2	3.0	8.1	13.3	6.0	11.3
Standard deviation	0.943	0	0.843	0.972	0.843	0.516	0.471	0.675	0.699	0.422	0	0.316	0.483	0	0.483
Standard error	0.298	0	0.267	0.307	0.267	0.163	0.149	0.213	0.221	0.133	0	0.1	0.153	0	0.153

is also superior to the well-known GPS [7] algorithm. In addition, we compared our method with the spectral analysis approach described in [1], summarised in Sect. 1 and used to set up the variable  $X$  in Sect. 3.2.

Each method was tested on our set of 15 benchmark matrices, which had sizes of up to  $100 \times 100$ . Considering the non-deterministic nature of GP, 10 independent runs were executed for each of the selected benchmark instances and performance values were averaged across such runs. Table 2 shows a performance comparison of the algorithms under test.

A simple inspection of the results of the experiments reported in Table 2 reveals that our GP approach is superior to both the RCMM algorithm and the spectral algorithm with respect to the mean of the bandwidth values and the number of the best results obtained. We performed a paired t-test in order to determine the statistical significance of the results obtaining a two-tailed P value of 0.0009 for the RCMM algorithm, and 0.0113 for the Spectral algorithm, which indicates that performance differences between our approach and these algorithms are statistically significant.

To give an idea of the reliability of the GP system, in Table 3 we provide its run-by-run results over the 15 benchmark problems. As one can see the algorithm produced very good results in all runs.

## 5 Conclusions

In this paper, a genetic programming approach has been introduced for solving the problem of the reduction of the bandwidth of a graph or a matrix. The proposed method was compared to the RCMM and Spectral algorithms which are among the best methods for solving the BMP. The results obtained on 15 standard benchmark matrices show that the proposed approach performs very favourably compared to these algorithms. However, it should be noted that the RCMM and Spectral methods are much faster than our GP-based method: they process a matrix in our dataset in seconds while our method requires approximately 3 minutes. So, the presented method is particularly appropriate when execution speed is not critical.

In future work, we will explore the possibility of building a parallel implementation of the system in order to improve its execution speed. We will also see if performance can be further enhanced by mutating constants, by using non-random populations and by hybridising the system by incorporating a local optimiser as was done in [12–14]. In addition, we will need to assess the generality and scalability of the proposed method by testing it with larger datasets and datasets including larger matrices.

## References

1. Barnard, S.T., Pothen, A., Simon, H.D.: A spectral algorithm for envelope reduction of sparse matrices. In: Supercomputing 1993: Proceedings of the 1993 ACM/IEEE Conference on Supercomputing, pp. 493–502. ACM, New York (1993)

2. Corso, G.D., Manzini, G.: Finding exact solutions to the bandwidth minimization problem. *Computing* 62(3), 189–203 (1999)
3. Cuthill, E., McKee, J.: Reducing the bandwidth of sparse symmetric matrices. In: *ACM National Conference*, pp. 157–172. Association for Computing Machinery, New York (1969)
4. Esposito, A., Malucelli, F., Tarricone, L.: Bandwidth and profile reduction of sparse matrices: An experimental comparison of new heuristics. In: *ALEX 1998*, Trento, Italy, pp. 19–26 (1998)
5. Garey, M., Graham, R., Johnson, D., Knuth, D.: Complexity results for bandwidth minimization. *SIAM Journal on Applied Mathematics* 34(3), 477–495 (1978)
6. George, J.A., Liu, J.W.H.: *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Englewood Cliffs (1981)
7. Gibbs, N.E., Poole, W.G., Stockmeyer, P.K.: An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM Journal on Numerical Analysis* 13(2), 236–250 (1976)
8. Gurari, E., Sudborough, I.: Improved dynamic programming algorithms for bandwidth minimization and the min-cut linear arrangement problem. *Journal of Algorithms* 5, 531–546 (1984)
9. Harary, F.: *Graph Theory*. Addison-Wesley, Reading (1969)
10. Koohestani, B., Corne, D.: An improved fitness function and mutation operator for metaheuristic approaches to the bandwidth minimization problem. In: *Proceedings of the 1st International Conference on Bio-Inspired Computational (BICS)*, AIP Conference Proceedings, vol. 1117, pp. 21–28 (2009)
11. Koza, J.R.G.P.: *On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge (1992)
12. Lim, A., Lin, J., Rodrigues, B., Xiao, F.: Ant colony optimization with hill climbing for the bandwidth minimization problem. *Applied Soft Computing* 6(2), 180–188 (2006)
13. Lim, A., Lin, J., Xiao, F.: Particle swarm optimization and hill climbing for the bandwidth minimization problem. *Applied Intelligence* 26(3), 175–182 (2007)
14. Lim, A., Rodrigues, B., Xiao, F.: Integrated genetic algorithm with hill climbing for bandwidth minimization problem. In: Cantú-Paz, E., et al. (eds.) *GECCO 2003*. LNCS, vol. 2724, pp. 1594–1595. Springer, Heidelberg (2003)
15. Lim, A., Rodrigues, B., Xiao, F.: A centroid-based approach to solve the bandwidth minimization problem. In: *37th Hawaii International Conference on System Sciences (HICSS)*, Big Island, Hawaii, p. 30075a (2004)
16. Lim, A., Rodrigues, B., Xiao, F.: A genetic algorithm with hill climbing for the bandwidth minimization problem, available from Citeseer-X (2002)
17. Marti, R., Laguna, M., Glover, F., Campos, V.: Reducing the bandwidth of a sparse matrix with tabu search. *European Journal of Operational Research* 135(2), 450–459 (2001)
18. Papadimitriou, C.H.: The NP-completeness of the bandwidth minimization problem. *Computing* 16(3), 263–270 (1976)
19. Pissanetsky, S.: *Sparse Matrix Technology*. Academic Press, London (1984)
20. Poli, R., Langdon, W.B., McPhee, N.F.: *A Field Guide to Genetic Programming with contributions by J. R. Koza* (2008), Published via <http://lulu.com>
21. Rodriguez-Tello, E., Jin-Kao, H., Torres-Jimenez, J.: An improved simulated annealing algorithm for bandwidth minimization. *European Journal of Operational Research* 185(3), 1319–1335 (2008)