A New Measure for the Bandwidth Minimization Problem

Jose Torres-Jimenez and Eduardo Rodriguez-Tello

ITESM, Campus Morelos. Paseo de la Reforma 182-A Lomas de Cuernavaca
62589, MEXICO Telephone (52 7) 329 7100
jtorres@campus.mor.itesm.mx, ertello@campus.mor.itesm.mx

Abstract. The Bandwidth Minimization Problem for Graphs (BMPG) can be defined as finding a labeling for the vertices of a graph, where the maximum absolute difference between labels of each pair of connected vertices is minimum. The most used measure for the BMPG algorithms is β, that indicates only the maximum of all absolute differences.
After analyzing some drawbacks of β, a measure, called γ, which uses a positional numerical system with variable base and takes into account all the absolute differences of a graph is given.
In order to test the performance of γ and β a stochastic search procedure based on a Simulated Annealing (SA) algorithm has been applied to solve the BMPG. The experiments show that the SA that uses γ has better results for many classes of graphs than the one that uses β.

Keywords: Bandwidth Metric, Bandwidth Minimization Problem, Graphs, Simulated Annealing.

1 Introduction

There are essentially two ways in which the BMP can be approached, wether as a graph or as a matrix. The equivalence of a graph and a matrix is made clear by replacing the nonzero entries of the matrix by $1's$ and interpreting the result as the adjacency matrix of a graph.

The Matrix Bandwidth Minimization Problem seems to have been originated in the 1950's when structural engineers first analyzed steel frameworks by computer manipulation of their structural matrices [10][11]. In order that operations like inversion and finding determinants take the least time as possible, many efforts were made to discover an equivalent matrix in which all the nonzero entries would lay within a narrow band near the main diagonal (hence the term "bandwidth") [15].

The BMP for graphs (BMPG) was proposed independently by Harper [7] and Harary [6]. This can be defined as finding a labeling for the vertices of a graph, where the maximum absolute difference between labels of each pair of connected vertices is minimum.

Formally, Let $G = (V, E)$ be a finite undirected graph, where V defines the set of vertices (labeled from 1 to N) and E is the set of edges. And a linear layout

$\tau = \{\tau_1, \tau_2, ..., \tau_N\}$ of G is a permutation over $\{1, 2, ...N\}$, where τ_i denotes the label of the vertex that originally was identified with the label i. The bandwidth β of G for a layout τ is:

$$\beta_\tau(G) = Max_{\{u,v\}\in E}|\tau(u) - \tau(v)|. \quad (1)$$

Then the BMPG can be defined as finding a layout τ for which $\beta_\tau(G)$ is minimum [17].

The first work that helped to understand the computer complexity of the BMPG, was developed by Papadimitriou, who demonstrated that the decision problem associated to the BMPG is a NP-Complete problem [14]. Later, it was demonstrated that the BMPG is NP-Complete even for trees with a maximum degree of three [12].

There are several algorithms reported to solve the BMPG, they can be divided into two classes: exact and approximate algorithms. Exact algorithms, guaranteed always to discover the optimal bandwidth, example of exact algorithm is the one published by Gurari and Sudborough [5], it solves the BMPG in $O(N^K)$ steps, where k is the bandwidth searched for that graph. Approximate algorithms are best known as approximate in the sense that they do not guarantee to find the actual bandwidth of the graph, examples of this sort of algorithms are [2] [13] [4] [8].

All algorithms mentioned in last paragraph use as a measure of the quality for a solution β. The only reported exception is a Dueck's work in which not only β is used, but he also takes into account differences among adjacent vertices close to β [3]. The advantage of Dueck's approach is the ability to distinguish small improvements that not necessarily lower the value of β.

Following the idea of Dueck's work, a new measure (namely γ) is proposed in this paper. This new measure is able to capture even the smallest improvement that orients the searching of better solutions (i.e. solutions in which all the absolute differences are minimized).

The rest of this work is organized into eight more sections. Section 2 concentrates on an analysis of the β measure, cardinality of its equivalency classes, and possible drawbacks of β. In Section 3 a new measure called γ is proposed, which takes into account all the edges of the graph. Section 4 makes a comparison between β and γ. Section 5 explains problems in the computation of γ for graphs with a big number of vertices. Section 6 shows the details of the implementation of the SA algorithm that was used to study the performance obtained when it uses either γ or β. Section 7 explains the computational results obtained with the SA algorithm for both metrics γ and β. Finally, in Section 8 some conclusions of this research are presented.

2 The β measure

The β measure for graphs equivalent to matrices, is a special case where the number of reflexive edges for each vertex in the graph is at most one, and the number of edges among vertex i and vertex j $(i \neq j)$ is at most two.

β, is the most used measure for the BMP algorithms [2] [13] [1] [8] (an exception is a Dueck's work in which he uses a measure in which not only β is used, but he also takes into account differences among adjacent vertices close to β [3]). Since the number of vertices is N, then β can only take N different values (from 0 to $N-1$), $\beta = 0$ implies that a graph that has either no edges or only reflexive edges. Thus, the space of all possible solutions ($N!$) is partitioned in N different equivalency classes.

Next, some features of the β measure are analyzed. Let ω_i be the cardinality of the equivalency class with $\beta = i$. Then, it is easy to show that $\omega_0 = 2^N - 1$ (the case of a graph with no edges is subtracted) and $\omega_1 = 2^N(2^{2(N-1)} - 1)$ then:

$$\omega_i = \prod_{j=1}^{i-1} 2^{2(N-j)}(2^{2(N-i)} - 1) \tag{2}$$

Now, in order to verify that all the possible graphs are taken into account, a partial summation of cardinalities of the equivalency classes as $S_i = \sum_{j=0}^{i} \omega_i$ is defined; then $S_0 = \omega_0$, $S_1 = S_0 + \omega_1$, and

$$S_i = S_{i-1} + \omega_i = 2^N \prod_{j=1}^{i} 2^{2(N-j)} - 1 \tag{3}$$

Therefore, it is straightforward to show that the summation of all cardinalities of the equivalency classes equals the total number of graphs minus 1 (the case of a graph without edges):

$$S_{N-1} = 2^{N^2} - 1 \tag{4}$$

where 2^{N^2} is in fact the number of all possible graphs.

The β measure is very gross with very few equivalency classes, in consequence, each equivalency class has high cardinality. Additionally the β measure does not take into account all the absolute differences between labels of adjacent vertices, but the maximum absolute difference. In this sense there is no way to make distinctions between elements that belong to the same β equivalency class.

3 The γ measure

Given the features of β it has been developed a new measure, called γ, which takes into account all the absolute differences of the graph (remember that it has been referred to graphs equivalent to matrices). The proposed measure, which represents a positional numerical system with variable base is the following:

$$\gamma = \sum_{i,j|(i,j)\in E} P(N, |i-j|) \tag{5}$$

Where the sum is carried out for all the edges of the graph and P is defined in an iterative way according to:

$$P(N,k) = \begin{cases} 1 & k = 0 \\ (N+1)\prod_{j=2}^{k}(2N-2j+3) & 1 \leq k \leq N \end{cases} \tag{6}$$

It can be verified, that if i and j differ significantly, $P(N,k)$ gives a very large value, and that the total number of equivalency classes is given by the expression $P(N,N)$.

For a particular graph there are: W_0 absolute differences between adjacent vertices with value zero, corresponding to reflexive edges; W_1 differences with value one; ...; and W_{N-1} differences with value $N-1$. Then the total number of equivalent graphs for that particular graph is:

$$\binom{N}{W_0}\binom{2(N-1)}{W_1}\cdots\binom{2(N-i)}{W_i}\cdots\binom{2}{W_{N-1}} \tag{7}$$

that can be expressed in a shorter way with the formula:

$$\binom{N}{W_0}\prod_{i=1}^{N-1}\binom{2(N-1)}{W_i} \tag{8}$$

To demonstrate that the summation of the cardinalities of all equivalency classes equals the number of possible graphs, it is necessary to use the Formula 8 instantiated with all possible values of $W_0, W_1, ..., W_{N-1}$, compute the sum, and check if it equals to 2^{N^2}. This can be expressed as:

$$\sum_{i=0}^{N}\binom{N}{i}\prod_{j=1}^{N-1}\sum_{r=0}^{2(N-j)}\binom{2(N-j)}{r} \tag{9}$$

but since $\sum_{i=0}^{N}\binom{N}{i} = 2^N$ and $\sum_{r=0}^{2(N-j)}\binom{2(N-j)}{r} = 2^{2(N-j)}$ then Formula 9 is simplified as follows:

$$2^N\prod_{j=1}^{N-1}2^{2(N-j)} \tag{10}$$

rewriting Equation 10:

$$2^N 2^{2\sum_{j=1}^{N-1}(N-j)} = 2^N 2^{2\frac{N(N-1)}{2}} = 2^{N^2} \tag{11}$$

and finally:

$$2^N 2^{2\frac{N(N-1)}{2}} = 2^{N^2} \tag{12}$$

4 Comparing γ and β

In this section the β measure and the γ measure are compared. First the number of equivalency classes of β and γ referred as ω_β(it can be verified that its value

N	$\boldsymbol{\omega}_\beta$	$\boldsymbol{\omega}_\gamma$
5	5	5670.0
10	10	7.202×10^{9}
70	70	6.7587×10^{121}
150	150	5.6674×10^{308}
300	300	6.1101×10^{705}

Table 1. $\boldsymbol{\omega}_\beta$ and $\boldsymbol{\omega}_\gamma$ values for some values of N

N	$2^{N^2}/\omega_\beta$	$2^{N^2}/\omega_\gamma$
5	6.7109×10^{6}	5917.9
10	1.2677×10^{29}	1.7601×10^{20}
70	1.5918×10^{1473}	1.6486×10^{1353}
150	9.9727×10^{6770}	2.6395×10^{6464}
300	1.6691×10^{27090}	8.1952×10^{26386}

Table 2. The average cardinality of β and γ for some values of N

is N) and ω_γ(it can be verified that $\omega_\gamma = P(N, N)$) are contrasted, then the average of cardinalities for the equivalency classes for each of these measures are illustrated.

In Table 1, $\boldsymbol{\omega}_\beta$ and $\boldsymbol{\omega}_\gamma$ for different values of N are shown, it is important to emphasize that $\boldsymbol{\omega}_\beta$ has a linear increment and $\boldsymbol{\omega}_\gamma$ has an exponential increment.

Table 2 shows many average values of cardinalities for the equivalency classes for β and γ ($2^{N^2}/\omega_\beta$ and $2^{N^2}/\omega_\gamma$) according to the number of vertices N of a graph.

As it can be observed in Tables 1 and 2, γ is a finer measure than β since it has the ability to create more equivalency classes with a lower cardinality.

5 Computing γ for big values of N

A problem in computing γ is that the resulting values may exceed easily the precision of a computer when N takes large values. A possible solution could be to use the logarithm of γ, but it seems that is not possible to obtain the logarithm of the expression: $\gamma = \sum_{i,j|(i,j)\in E} P(|i-j|)$, because of it is a summation. However, an expression has been obtained that permits to compute γ in terms of logarithms and does not get involved in using huge numbers. To illustrate this, it is assumed the graph in Figure 1.

For this particular graph: $W_0 = 5$, $W_1 = 3$, $W_2 = 4$, $W_3 = 1$, $W_4 = 2$ (where W_i refers to the number of absolute differences with value i between adjacent vertices); additionally it is verified that $P(5,0) = 1$, $P(5,1) = 6$, $P(5,2) = 54$, $P(5,3) = 378$, $P(5,4) = 1890$.

Then, $\gamma = \sum_{i=0}^{4} W_i P(5,i) = 5 + 18 + 216 + 378 + 3780 = 4397$, but γ could be expressed in the following way:

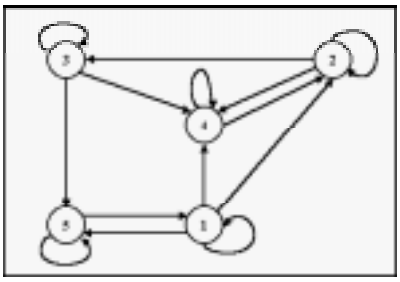

Fig. 1. An example of a graph with five vertices

$$\gamma = \left(\frac{\frac{W_0}{\left(\frac{P(5,1)}{P(5,0)}\right)} + W_1}{\frac{\left(\frac{P(5,2)}{P(5,1)}\right)} + W_2}{\frac{\left(\frac{P(5,3)}{P(5,2)}\right)} + W_3}{\left(\frac{P(5,4)}{P(5,3)}\right)} + W_4 \right) P(5,4),$$ taking logarithms in both sides of this equation:

$$\log(\gamma) = \log \left(\frac{\frac{\frac{W_0}{\left(\frac{P(5,1)}{P(5,0)}\right)} + W_1}{\left(\frac{P(5,2)}{P(5,1)}\right)} + W_2}{\frac{\left(\frac{P(5,3)}{P(5,2)}\right)} + W_3}{\left(\frac{P(5,4)}{P(5,3)}\right)} + W_4 \right) + \log(P(5,4)) \tag{13}$$

The great advantage of the previous expression is that it never computes very large numbers, all denominators in the first term of the right side of the equation correspond to the series: $(N+1), (2N-1), (2N-3), (2n-5), ..., 7, 5$ and the second term can be expressed in the next way:

$$\log(P(N,K)) = \begin{cases} \log(N+1) + \sum_{j=2}^{k} \log(2N-2j+3) & 1 \leq k \leq N \\ 0 & k = 0 \end{cases} \tag{14}$$

then the logarithm of γ is:

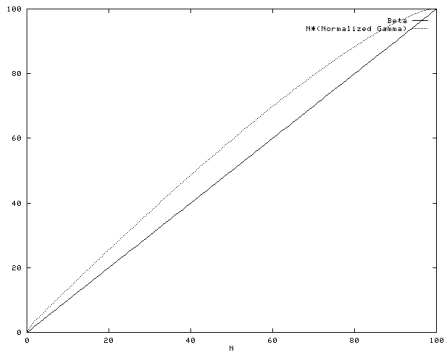

Fig. 2. Comparative graph between β and $N * \gamma_{Norm}$.

$$\log(\gamma) = \log\left(\frac{\frac{\frac{W_0}{N+1}+W_1}{\frac{2N-1}{2N-3}}+W_2}{2N-5}+W_3+W_4\right)+\log(N+1)+\sum_{j=2}^{N-1}\log(2N-2j+3) \tag{15}$$

Summarizing, it has been demonstrated that it is quite possible to compute the logarithm of γ without computing big numbers, and in this way, to overcome the numeric precision problem in a computer.

Additionally, $\log(\gamma)$ can be normalized from 0 to 1 using the formula:

$$\gamma_{Norm} = \frac{\log(\gamma)}{\log(P(N,N))} \tag{16}$$

The importance of this normalization of γ is that it permits to see the similarity with β, this can be seen in Figure 2 (it is relevant to emphasize that γ_{Norm} keeps up the ability to represent more equivalency classes with a lower cardinality).

6 A simulated annealing approach to solve the BMPG

We have developed a heuristic algorithm based on the principle of SA, that will approximate the bandwidth of a graph G by examining randomly generated layouts τ of G. We generate these new layouts by interchanging a pair of distinct labels of the set of vertices V. We have called this interchanging operation, a *move*.

Our SA algorithm begins initializing some parameters as the temperature, T; the maximum number of accepted moves at each temperature, *max_moves*; the maximum number of moves to be attempted at each temperature, *max_attempted_moves*; *max_frozen* is the number of consecutive iterations allowed for

which the number of accepted moves is less than *max_moves*; and the cooling rate *cool_rate*. The algorithm continues by randomly generating a move and then calculating the change in the cost function for the new labelling of the graph, either with γ or β. If the cost decreases then the move is accepted. Otherwise, it is accepted with probability $P(\Delta C) = e^{-\Delta C/T}$ where T is the temperature and ΔC is the increase in cost that would result from a particular move. The next temperature is obtained by using the relation $T_n = T_{n-1} *$ *cool_rate*. *sa_band* is the minimum bandwidth of the labellings generated by the algorithm up that point in time. We count the number of accepted moves and if it falls below a given limit then the system is *frozen*.

Next, we present the algorithm for the simulated annealing general procedure:

Procedure Anneal*(G, best_map)*
  $T = 0.00004$*;* *cool_rate* $= 0.85$*;*
  map = best_map = random labeling;
  sa_band = Bandwidth(G, best_map);
  max_moves $= 50 * |E|$*;*
  max_attempted_moves $= 2 *$ *max_moves;*
  max_frozen $= 50$*;* *frozen* $= 0$*;*
  While *(frozen* $\leq$ *max_frozen)*
    moves = attempted_moves $= 0$*;*
    While *((moves* $\leq$ *max_moves)* **And**
        (attempted_moves $\leq$ *max_attempted_moves))*
      attempted_moves ++;
      a random move is generated, map_ran;
      If *(bandwidth decreases* **Or** *random_number()* $< e^{-\Delta Bandwidth/T}$*)*
        map = map_ran; *moves ++;*
        If *(sa_band < Bandwidth(G, map))*
          best_map = map;
          sa_band = Bandwidth(G, map);
        End If
      End If
    End While
    $T = T *$ *cool_rate;*
    If *(attempted_moves > max_attempted_moves)*
      frozen++;
    Else
      frozen = 0;
    End If
  End While
End Anneal

The parameters of the SA algorithm were chosen taking into account our experience, and some related works [9][16]. It is important to remark that value of *max_moves* depends directly on the number of edges of the graph, because more moves are required for denser graphs; the *max_attempted_moves* is set to a large number ($50 *$ *max_moves*), because few moves will result in bigger bandwidths.

Graph	N	SA-β	SA-γ	Best β	% Improvement
Path50	50	7	1	1	600.00
Path100	100	18	1	1	1700.00
Path150	150	32	2	2	1500.00
Cycle50	50	7	2	2	250.00
Cycle100	100	18	2	2	800.00
Cycle150	150	28	2	2	1300.00
TreeT40	40	8	7	7	14.29
TreeB63	63	12	9	9	33.33
TreeB127	127	28	16	15	75.00
TreeQ85	85	18	15	15	20.00
TreeT121	121	27	16	16	68.75
Grid100	100	23	17	10	35.29
Grid169	169	40	28	13	42.86

Table 3. Results obtained with a SA for the BMPG using γ and β

The *max_frozen* parameter that controls the external loop of our algorithm is set to 50. By modifying these three parameters one can obtain results more quickly, but probably they will not be as close to $\beta(G)$. We found in our experiments that the above values give a good balance between the quality of the results and the invested computational effort.

7 Computational Results and Discussion

In order to test the performance of our new metric called γ we studied six different classes of graphs of increasing sizes, including paths, cycles, binary trees, ternary trees, Quaternary trees and grids.

Table 3 shows a column with the name of the graphs, which represents the class of each graph. Column titled N represents the number of nodes in the graph. Columns SA-β and SA-γ represent the bandwidth that was obtained with the SA algorithm that uses the metric β or γ respectively. In column Best β, the value of the best bandwidth obtained with the SA-γ is presented. Finally the last column presents the improvement obtained when the γ metric was used.

It is important to say that we ran each SA algorithm five times with every graph. In table 3 the results presented are the arithmetic average of those experiments. The results of these experiments show that the SA that uses γ consistently has better results for many classes of graphs than the one that uses β. So we could conclude that γ is a better metric than β.

8 Conclusions

1. It has been presented a measure called γ for the BMPG, which has the advantage over β, that it represents more equivalency classes with lower cardinality.

2. To avoid the precision problems when computing the γ measure for large values of N, an algorithm based on the use of logarithms was proposed.
3. In order facilitate the use of γ measure, it has been defined a procedure to normalize its values from 0 to 1.
4. The features of γ seem to fit very well for optimization algorithms to solve the BMPG, it is evidenced by the results presented in the previous section.

References

1. F. Malucelli A. Esposito, S. Fiorenzo and L. Tarricone, *A wonderful bandwidth matrix reduction algorithm*, Submitted to Operations Research Letters.
2. E. Cutchill and J. McKee, *Reducing the bandwidth of sparse symmetric matrices*, Proceedings 24th National of the ACM (1969), 157–172.
3. G. Dueck and J. Jeffs, *A heuristic bandwidth reduction algorithm*, Journal of combinatorial mathematics and computers (1995), no. 18.
4. A. George and W. Liu, *Computer solution of large sparse positive definite systems*, Prentice Hall, Englewood Cliffs, NJ, 1981.
5. E.M. Gurari and I.H. Sudborough, *Improved dynamic programming algorithms for bandwidth minimization and the min-cut linear arrangement problem*, Journal of Algorithms, 5 (1984), 531–546.
6. F. Harary, *Theory of graphs and its aplications*, 1967, p. 161.
7. L.H. Harper, *Optimal assignment of numbers to vertices*, Journal of SIAM **12** (1964), 131–135.
8. F. Makedon J. Haralambides and B. Monien, *An aproximation algorithm for caterpillars*, Journal of Mathematical Systems Theory (1991), no. 24, 169–177.
9. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by simulated annealing*, Science **220** (1983), 671–680.
10. E. Kosko, *Matrix inversion by partitioning*, The Aeronautical Quarterly (1956), no. 8, 157.
11. R. R. Livesley, *The analysis of large structural systems*, Computer Journal **3** (1960), 34.
12. D.S. Johnson M.R. Garey, R.L. Graham and D.E. Knuth, *Complexity results for bandwidth minimization*, SIAM Journal of Applied Mathematics 34 (1978), 477–495.
13. W.G. Poole N.E. Gibbs and P.K. Stockmeyer, *An algorithm for reducing the bandwidth and profile of a sparse matrix*, SIAM Journal on Numerical Analysis 13 (1976), 235–251.
14. C.H. Papadimitriou, *The NP-Completeness of the bandwidth minimization problem*, Journal of Computing (1976), no. 16, 263–270.
15. A.K. Dewdney P.Z. Chinn, J. Chvátalová and N.E. Gibbs, *The bandwidth problem for graphs and matrices – a survey*, Journal of Graph Theory **6** (1982), 223–254.
16. William M. Spears, *Simulated annealing for hard satisfiability problems*, Tech. Report AIC-93-015, AI Center, Naval Research Laboratory, Washington, DC 20375, 1993.
17. José Torres, *Minimización del ancho de banda de un grafo usando un algoritmo genético*, Ph.D. thesis, Instituto Tecnologico y de Estudios Superiores de Monterrey, Campus Morelos, 1997.