Mapana – Journal of Sciences
2023, Vol. 22, Special Issue 1, *101-128*
ISSN 0975-3303 | https://doi.org/10.12723/mjs.sp1.9

Grid bandwidth Minimization Problem: Simulated Annealing Approach

Aditi Khandelwal*, Kamal Srivastava*, Gur Saran*

Abstract

For a given graph G, the Grid Bandwidth Minimization Problem (GBMP) is to find an embedding of G into a host graph H such that the bandwidth over all edges is minimized. It is an NP-hard problem with applications in VLSI circuit design, numerical analysis, computational biology, graph theory and scheduling. In this paper, a Simulated Annealing (SA) algorithm is developed for GBMP in which initial solution is generated using two problem-specific construction heuristics. Four neighbourhood strategies are designed to explore the search space. Experiments conducted on benchmark instances achieve results which fall within the bounds whereas for grid graphs it attains optimal values.

Keywords: Grid embedding, Bandwidth, Simulated Annealing

1. Introduction

An embedding of G on H is a one-to-one association of the vertices of a graph G with vertices of a host graph H [1]. If the If the host graph H is a path, cycle and grid then the resultant embedding is known as linear, cyclic and grid embedding respectively. One of the measures associated with an embedding is dilation which can be defined as the maximum distance between adjacent vertices of G on the host graph H. The problem of finding a linear (cyclic) embedding

* Dayalbagh Educational Institute, Agra, Uttar Pradesh, 282005, India; aditikhandelwal8193@gmail.com

of G such that dilation over all edges is minimized is the well-known Bandwidth Minimization Problem (Cyclic Bandwidth Minimization Problem) [2]. For grid embeddings, the problem is termed as Two-Dimensional Bandwidth Minimization Problem [3]. We will refer to this problem as Grid Bandwidth Minimization Problem (GBMP).

In this paper we focus on GBMP. Formally, let G be a graph with vertex set $V(G)$ and edge set $E(G) \subseteq V(G) \times V(G)$ with $|V(G)| = n$. For $v \in V(G)$, $N(v) = \{u \in V(G) | (u, v) \in E(G)\}$ and degree of vertex v, $\deg(V) = |N(V)|$. Let H be a fixed host graph which is a grid of size $2 \times Z$, where $Z = \left\lceil \frac{n}{2} \right\rceil$. The embedding of graph G in H is a one-one mapping $\pi: V(G) \rightarrow V(H)$ which assigns a row and a column number to each vertex that correspond to positions on the grid. Explicitly, $\pi(x) = (a, b)$ means that the vertex $x \in V(G)$ is located in the grid at row a and column b and is denoted by $x(a, b)$ in H.

Let $x(a, b), y(a', b') \in V(H)$. Then the distance between x and y, defined by the L_1 norm distance, is given by:

$$d_H(x, y) = |a - a\ b - b'| \tag{1}$$

The bandwidth of an embedding π is defined as:

$$b_\pi(G) = \max\{d_H(\pi(x), \pi(y)) : xy \in E(G)\}. \tag{2}$$

The bandwidth of the graph G, denoted by $b(G)$, is the minimum of $b_\pi(G)$ over all embeddings π, i.e.;

$$b(G) = \min\{b_\pi(G) : \pi \text{ is an embedding of } G \text{ in } H\}. \tag{3}$$

For an edge $uv \in E(G)$, $d_H(u, v)$ be the maximum over all edges of $E(G)$, then $b_\pi(G) = d_H(u, v)$ and the edge(u, v) is known as a critical edge. Throughout, we will refer an embedding π as a solution and $b_\pi(G)$ as the $cost$ of the solution. Neighbourhood of a solution s, denoted by $nbd(s)$, is the set of solutions obtained by small perturbations in the solution s.

Figure 1 illustrates the embedding of an undirected graph (1(a)), into a $2 \times Z$ grid (1(b)). The vertices are selected randomly and placed

beginning from the centre of the grid, that is from position $(2, 2)$. The bandwidth of the resultant embedding 1(b) is 4. Since the number of vertices in the given graph are odd, one position on the grid will always remain freely available for easy movement of others which further aids in converging towards near optimal results.

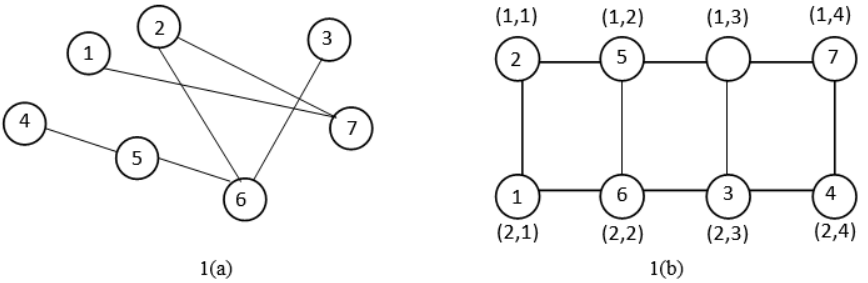

Fig 1. Embedding of an undirected graph of (a) into a $2 \times Z$ grid in (b).

The GBMP has been proved to be NP-complete [4]. A large number of problems in different domains can be formulated as graph embedding problems, including optimization of networks for parallel computer architectures, VLSI circuit design, information retrieval, numerical analysis, computational biology, graph theory, scheduling and archaeology [3, 5, 6]. In spite of having wide range of applications, the problem has not received much attention so far. Only a few bounds exist in literature for the problem. Lin [7] introduced the term two-dimensional bandwidth, denoted by $B_2(G)$, and proved the following density lower bound:

For any graph G of n vertices,

$$B_2(G) \geq \left\lceil \frac{\sqrt{n}-1}{D(G)} \right\rceil,$$

Lin [7] proved that for any graph G with n vertices and diameter $D(G)$,

$$\frac{\delta(n)}{D(G)} \leq B_2(G) \leq \delta(n),$$

where $\delta(n) = min\left\{2\left\lceil\frac{\sqrt{2n-1}-1}{2}\right\rceil, 2\left\lceil\sqrt{\frac{n}{2}}\right\rceil - 1\right\}$. It is also proved that for a k-level binary tree, denoted by $T_{2,k}$, the two-dimensional bandwidth has an upper bound. Optimal results for certain classes of graphs are also obtained [7, 8]. It is worthy to mention here that the grid considered in [7] is an $n \times n$ grid.

Metaheuristics have become an important tool to tackle hard combinatorial optimization problems. However, for GBMP only a few heuristic approaches are available in the literature. Rodriguez-Garcia et al. [3] have presented a greedy construction heuristic and a local search procedure for 2D-bandwidth minimization problem. They further gave three Constraint Satisfaction problem models and a Basic Variable Neighbourhood Search Algorithm[2] with host graph as a $\sqrt{n} \times \sqrt{n}$ grid. In the literature surveyed by us so far, no other metaheuristic procedure has been proposed so far.

In this paper we have developed a Simulated Annealing algorithm for the Grid Bandwidth Minimization Problem (SAGBMP) with size of the host graph being $2 \times Z$. We have designed two construction heuristics which help generate high quality initial solutions to aid faster convergence. The first heuristic selects vertices randomly and places them on the grid using a systematic scheme. The second heuristic attempts to place adjacent vertices as close as possible in the embedding in order to reduce their contribution to the bandwidth. Four neighbourhood strategies are also developed that are required in the SA procedure which help in better exploration of the search space. In order to have best configuration of SA parameter tuning experiments are done. Final experiments are conducted to access the performance of SAGBMP.

The rest of the paper is organised as follows. Section 2 contains the algorithm proposed for SAGBMP. Heuristics and neighbourhood strategies used in the algorithm are explained in Section 3 and Section 4 respectively. Section 5 discusses the preliminary experiments and final experiments conducted on particular classes of graphs using the proposed algorithm for GBMP. Section 5 concludes the paper.

2. Simulated Annealing Algorithm for Bandwidth Minimization Problem of $2 \times Z$ grid embedding

Simulated Annealing (SA) is a single solution-based metaheuristic which was first introduced by Kirkpatrick et. al. [9] and originated from statistical mechanics. It is inspired from the metallurgical technique of annealing where a material is heated at high temperatures and cooled down gradually to obtain a well-ordered state with minimal energy. SA helps to transpose solutions of optimization problems by gradually minimizing the objective function, similar to annealing, by controlling parameters of the algorithm.

The proposed SA algorithm for the Grid Bandwidth Minimization Problem (SAGBMP) begins with generating initial solutions using two heuristics and assigning an initial value to (i) the temperature parameter T, (ii) the cooling rate α that reduces the temperature T by a factor α as execution proceeds, until the final temperature is reached, (iii) $iter$, that determines the number of iterations of SAGBMP for each instance, and (iv) L, that defines the number of iterations to be performed at the temperature T at any given point of time. Further, at each iteration, the best solution among all solutions of $nbd(s)$ is stored in s'.

The neighbourhood strategies are designed to target the critical edge in order to improve the solution. The solution s' is accepted if $cost(s') \leq cost(s)$, and it replaces s. On the other hand, if $cost(s') \geq cost(s)$ then, s' can be accepted with a probability $e^{-(cost(s\prime)-cost(s))/T}$. The global $best$ (Step 17) is then updated if $cost(s) < cost(best)$. The SAGBMP algorithm is outlined in Algorithm 1.

Step 1: Input- $T_f, \alpha, iter, L, L_f$

Step 2: $T \leftarrow T_0$

Step 3: compute γ

Step 4: Generate solution s_1 using $H1$ and s_2 using $H2$

Step 5: $s \leftarrow argmin(cost(s_1), cost(s_2))$

Step 6: $best \leftarrow s$

Step 7: **while**$T > T_f$ do

Step 8: $improvement \leftarrow$ false

Step 9: **for**$i \leftarrow 1$ to L do

Step 10: $s' \leftarrow s^*$ when $cost(s^*)$is minimum among all solutions of $nbd(s)$

Step 11: **if**cost$(s') < cost(s)$

Step 12: $s \leftarrow s'$

Step 13: **else if**$random(0,1) < e^{-(cost(s')-cost(s))/T}$

Step 14: $s \leftarrow s'$

Step 15: **end**

Step 16: **if**cost$(s) < cost(best)$

Step 17: $best \leftarrow s$

Step 18: $improvement \leftarrow$ true

Step 19: **end**

Step 20: **end**

Step 20: **if**$improvement =$ false

Step 21: $T \leftarrow T * \alpha$

Step 22: $L \leftarrow L * \gamma$

Step 23: **end**

Step 24: **end**

Step 25: return $best$

Algorithm 1

In algorithm 1, the purpose of γ at step 3 is to increment the number of iterations to be performed at a certain temperature to ensure that when temperature reaches its final value, the iterations also reach their final value. If $r = \frac{\log(T_f) - \log(T_0)}{\log(\alpha)}$, is the number of temperature reductions then $\gamma = \exp\left(\frac{\log(L_f) - \log(L)}{r}\right)$. The $improvement$ variable defined at step 8 keeps account of the improvement in solutions and ensures that T and L are modified if the global best solution is not improved. It remains noteworthy that feasibility of solution is ensured at each step.

3. Heuristics for generating Initial Solution

Heuristic methods are designed for generating feasible initial solutions for the problem that are of reasonable quality, requiring small computing time. In this section we describe two construction heuristics for generating the initial solution. The main idea behind both the heuristics is to ensure that feasible solutions are obtained by placing more and more vertices adjacent to each other.Note: A position (a, b) is said to be $free$ if no vertex is assigned to it.

Heuristic$H1$:

Heuristic $H1$ is a random heuristic procedure which begins with selecting any random vertex and places it in the middle of the grid (the i^{th} row and j^{th}column of the grid will be denoted by $grid(i, j)$). From the remaining set of vertices, a vertex is randomly selected and placed on the nearest available position randomly. The process continues until all the vertices are placed on the grid. Pseudocode for this heuristic is presented in Algorithm 2.

Step 1: $i \leftarrow 1$

Step 2: select $u \in V$ randomly

Step 3: $grid(a, b) \leftarrow u$ where $a = 1$ and $b = \left(\frac{1}{2}Z\right)$

Step 4: $V \leftarrow V/\{u\}$

Step 5: repeat until $V \neq \emptyset$

Step 6: select$u \in V$ randomly

Step 7: $x \leftarrow (b - i)$

Step 8: $y \leftarrow (b + i)$

Step 9: $j \leftarrow x$ or $j \leftarrow y$whichever $free$

Step 10: else

Step 11: $i \leftarrow i + 1$

Step 12: $grid(i, j) \leftarrow u$

Step 13: $V \leftarrow V/\{u\}$

Step 14: end

Step 15: end

Step 16: return $grid$

Algorithm 2

For an illustration, consider the example graph of Fig 1(a). Let the randomly selected vertices are in the order $5, 4, 2, 6, 7, 1$ and 3. Vertex 5 is first placed at centre of the grid at $(1, 2)$ (Step 3). The next available positions on the grid are $(1, 1)$ and $(1, 3)$ (Step 7 and Step 8). Since both the positions are available, vertex 4 is placed at $(1, 3)$. Similarly, the remaining vertices are placed by following the algorithm and the final embedding obtained from $H1$ is shown in

Figure 2. Since the number of vertices are odd it can be seen that one position on the grid remains available.

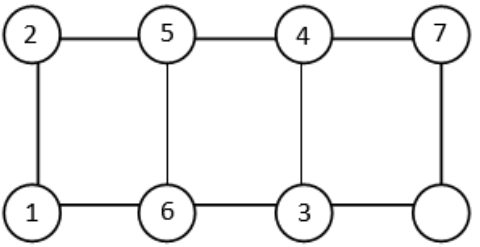

Figure 2. Solution from H1

Heuristic $H2$:

Heuristic $H2$ is a greedy heuristic that involves degrees of vertices for placing them on the grid. Let D be the list of vertices sorted in descending order of their degrees. The heuristic begins with selecting the first vertex u from the sorted list, i.e., the vertex with the highest degree and places it right in the middle of the grid at $\left(1, \frac{1}{2}Z\right)$. Then the neighbors nbr of the vertex are considered from the graph G and are placed at nearest $free$ positions on the host grid graph H. The vertices that are placed on the grid are removed from D. The next highest degree vertex remaining in D is selected and placed on the first $free$ position encountered. Again, the neighbors of this vertex are assigned positions on the grid graph and the process continues until all vertices are placed. The heuristic is outlined in outlined in Algorithm 3.

Step 1: Initialize $grid\ (a, b) = -1\ \forall\ a \in \{1, 2\}$and $\forall\ b \in \{1, \dots, Z\}$

Step 2:$D \leftarrow$ list of vertices in descending order of degree

Step 3: $u \leftarrow$ first vertex in D

Step 4: $grid\left(1, \frac{1}{2}Z\right) \leftarrow u$

Step 5: $D \leftarrow D \backslash \{u\}$

Step 6:$Q \leftarrow \{u\}$

Step 7: repeat until D$\neq \phi$

Step 8: $v \leftarrow$first vertex in Q

Step 9: $Q \leftarrow Q \backslash \{v\}$

Step 10: $i \leftarrow 1$

Step 11: $neighbor \leftarrow N(v) \cap D$

Step 12: $x_1 \leftarrow (a, b - i)$

Step 13: $x_2 \leftarrow (a, b + i)$

Step 14: $x_3 \leftarrow ((a + 1), b + (i - 1))$

Step 15: $x_4 \leftarrow (a - 1, b + (i - 1))$

Step 16: repeat until $neighbor \neq \phi$

Step 15: assign $w \in neighbor$ to x_1, x_2, x_3 or x_4 whichever is $free$

Step 16: if none is $free$

Step 17: $i \leftarrow i + 1$

Step 18: $Q \leftarrow Q \cup \{w\}$

Step 19: $D \leftarrow D \backslash \{w\}$

Step 20: return

Algorithm 3

Figure 3. illustrates the initial solution obtained using $H2$ for the graph G in Figure 1(a). The heuristic begins with a list of vertices sorted in descending order of their degrees given by $D = (6, 2, 7, 1, 5, 4, 3)$. The vertices 2 and 7 have the same degree so they

are randomly permuted and placed in the set. Similarly, vertices $1, 3, 4$ and 5have the same degrees and are thus permuted randomly.

The first vertex $v = 6 \in D$ is picked and placed at position $(1, 2)$ (Step 2). Now $N(6) = \{2, 3, 5\}$. As given in the algorithm, nearest possible positions on the grid to $(1, 2)$ are $(1, 1), (1, 3)$ and $(2, 2)$ (Steps 11-14). Vertices $2, 3$ and 5 are randomly placed at $(2, 2), (1, 1)$ and $(1, 3)$ respectively. Now $D = \{7, 1, 4\}$, so the next vertex 7 is selected and is placed at an available position on the grid say $(2, 4)$. $N(7) = \{1, 2\}$ but 2 has already been placed. The possible positions for 1 are $(1, 4)$ and $(2, 3)$ hence it is randomly placed at, say, $(1, 4)$. The only remaining vertex 4 is then placed at $(2, 1)$.

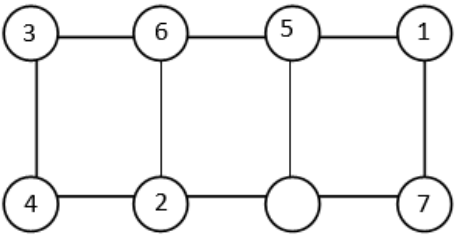

Figure 3. Solution generated by $H2$

4. Neighbourhood Strategies

SA requires perturbation of the current solution to explore the search space for new solutions. Four neighbourhood generation strategies are designed that work on the current solution to both intensify and diversify the solution. All the neighbourhood strategies move the vertices contributing to the bandwidth in order to minimize it. The first and second strategies cause a major change in the solution due to their exchanging nature whereas the third and fourth strategies change the solution by shifting vertices lying on the critical path. These strategies are described below.

Throughout, vertices $x = grid(a, b)$ and $y = grid(c, d)$ are end points of a critical path of solution s.

Neighbourhood Strategy $N1$

The first neighbourhood function is an exchange operator that works over the solution s. It fixes the position of vertex y and exchanges the vertex x with the vertex on either the left or the right of y randomly. This process ensures that $d_H(s(x), s(y)) = 1$ and the bandwidth due to this edge is instantly reduced. Figure 4 shows the state of the solution s before and after the application of the neighbourhood strategy $N1$. The bandwidth of the solution in Figure

(a) Initial Solution (b) Solution after exchange operator N1

Figure 4: Representation of $N1$, where the exchanged vertices are highlighted.

4(a)is 4 and it corresponds to the critical edge $(1, 7)$. The position of vertex 1 is fixed and vertex 7 is exchanged with vertex 6 which is currently adjacent to vertex 1. This process may result in the reduction of the bandwidth of the graph as the bandwidth due to this edge is reduced.

Neighbourhood Strategy $N2$

The second neighbourhood function is also an exchange operator, similar to $N1$, that works over the current solution s. It fixes the position of vertex x and exchanges the vertex y with the vertex on either the left or the right of the fixed vertex. Figure 5 shows the state of the solution s before and after the application of the neighbourhood strategy $N2$.The position of vertex 7 is fixed and vertex 1 is shifted to the $free$ postion on the grid. In this illustration, no other vertex is moved since there is a $free$ space on the grid. This new solution s' may result in the reduction of the bandwidth of the embedding occurring due to edge (1,7). If there is no $free$ position available then the usual exchange takes place.

(a) Initial Solution (b) Solution after exchange operator N2

Figure 5: Representation of $N2$, where the exchanged vertices are highlighted.

Neighbourhood Strategy $N3$

The third neighbourhood strategy is an insertion cum shifting operator that increases the exploration of the search space. The position of vertex x is fixed. The element to the right of this vertex is replaced with y. The other vertices are then shifted in anticlockwise direction to fill up the freshly vacated positions. The vertex that is replaced is repositioned to a newly available position post shifting. Figure 6 illustrates a solution obtained after $N3$ is applied to the current solution. The position of vertex 1 is fixed and vertex 7 is

(a) Initial Solution (b) Solution after exchange operator N3

Figure 6: Representation of $N3$, where the modified positions of vertices are highlighted.

shifted right next to it. The vertices 6 and 3 are shifted to the right in anticlockwise direction and the resultant solution is obtained as shown in Figure 6(b).

The fourth neighbourhood operator is also an insertion cum shifting operator similar to $N3$. In this, the position of y is fixed and the element to the left is replaced with x. The vertices are now shifted in clockwise direction repositioning all the vertices in the path and finally the replaced vertex is positioned at the new available position after shifting of vertices. Figure 7 shows the movement of vertices.

Figure 7 shows the movement of vertices with respect to N4.

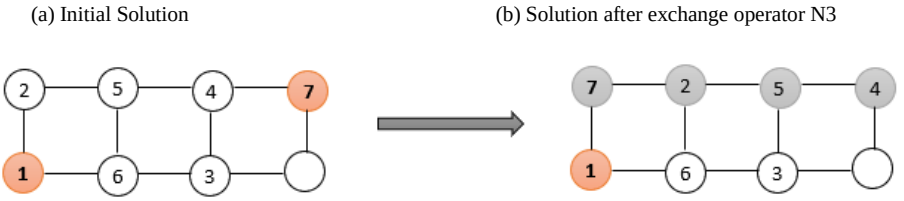

Figure 7: Representation of $N4$, where the modified positions of vertices are highlighted.

5. Experiments and Results

This section is devoted to experiments which are conducted in two phases to study the performance of the proposed algorithm. The test set consists of instances of Grid Graphs and graphs from the Harwell-Boeing Sparse Matrix Collection which are previously used in [10]. SAGBMP is implemented in C++ and experiments are conducted on Ubuntu 16.04 LTS machine with Intel (R) Core (TM) i5-2400 CPU @3.10_4 GHz and 7.7 GiB of RAM. Preliminary experiments are conducted on a representative set of 15 instances for tuning the parameters.

5.1. Preliminary Experiments

Preliminary experiments are conducted to set the parameters of SAGBMP i.e. temperature, cooling rate and number of iterations. These experiments are conducted on the representative set listed in Table 1.

Instance Name	$\|V\|$	$\|E\|$
ash85	85	219
bcspwr03	118	179

Instance Name	$\|V\|$	$\|E\|$
bcsstk01	48	176
bcsstk05	153	1135
bus494	494	586
can24	24	68
can73	73	152
can715	715	2975
dwt162	162	672
dwt245	245	608
ibm32	32	90
R10_20	10	28
R25_25	25	165

Instance Name	$\|V\|$	$\|E\|$
R40_30	40	234
R100_30	100	1485

Table 1. Set of instances used for preliminary experiments.

5.1.1. Comparison between Heuristics

Prior to parameter tuning, experiments are conducted to compare the performance of heuristics $H1$ and $H2$ (Section 3). These experiments are conducted on two sets of graphs, namely, Random graphs and Harwell Boeing graphs given in Table 1. For each heuristic and each graph instance 10 solutions are generated and their bandwidth is recorded over 30 trials. The mean bandwidth for each instance is computed and tested on the hypothesis that there is no difference in the mean for $H1$ and $H2$.

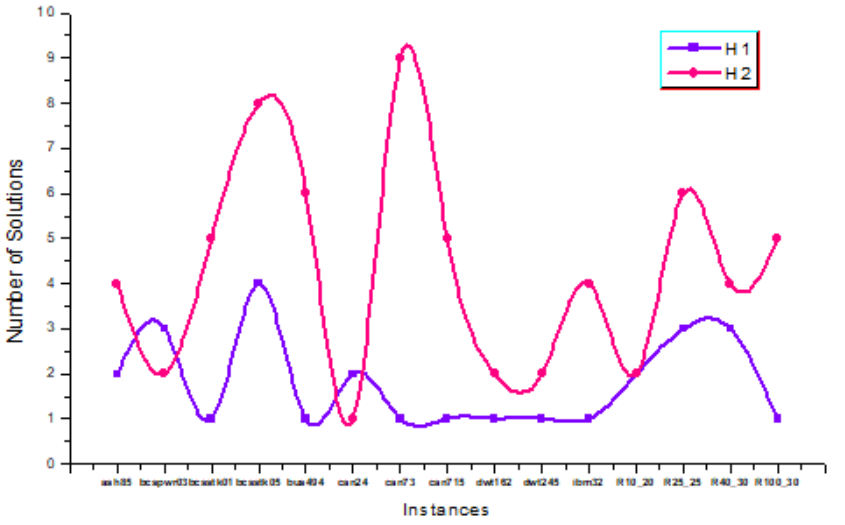

Figure 8: Mean bandwidth of solutions generated using $H1$ and $H2$

The null hypothesis is rejected using the results of t-test which means that the strategic heuristic $H2$ generates better solutions than the

random heuristic $H1$ as is also evident from Figure 8.But it can also be observed that sometimes$H1$ produces better solutions for some instances and therefore both $H1$ and $H2$ are used to generate initial solution.

5.1.2 Parameters Tuning Experiments:

As described in Algorithm 1, SAGBMP requires the temperature parameter T, the cooling rate α, $iter$, that determines the number of iterations of SAGBMP for each instance and L, that defines the number of iterations to be performed at the temperature T. A parameter tuning process helps to establish the values of these parameters to enhance the performance of SAGBMP. [10]

1. Initial temperature ($\boldsymbol{T}$): Initial temperature is one of the most important parameters of SA procedure and requires tuning to aid better exploration during algorithm execution. An experiment is conducted by fixing $\alpha = 0.95$, $L = 1000$ and $iter = 750$ while varying T for values 1000, 5000 and 10000. For each graph instance solutions are recorded over 30 trials. Two-way ANOVA with replication is applied for statistical comparison and no significant difference is found. However, it is observed that at $T = 10000$ time taken is considerably high (Figure 9). Therefore, we choose $T = 5000$ for further experimentation in the interest of performance.

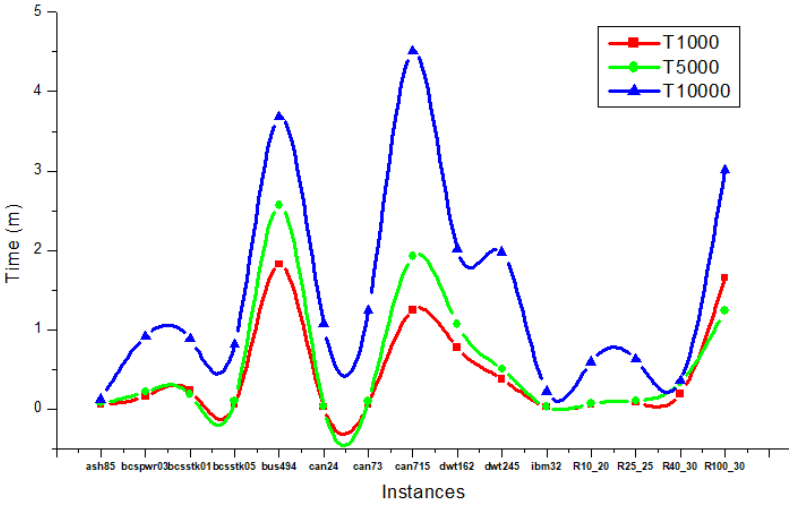

Figure 9: Execution Time for different value of T

2. Cooling Rate ($\boldsymbol{\alpha}$): The initial temperatureis temperature is reduced after each iteration using a Geometric cooling schedule $t \leftarrow t \times \alpha$, where α is the cooling factor. To investigate the performance of the algorithm at different values of α, an experiment is conducted by fixing $T = 5000$, $L = 1000$ and $iter = 750$ and varying α for values $0.80, 0.90, 0.95$ and 0.99. For each graph instance solutions are recorded over 30 trials. Two-way ANOVA with replication is applied for statistical comparison and results shows that there is significant difference between the means. Tukey's HSD test is then used to compare the means pairwise and it is found that there is significant pairwise difference between $\alpha = 0.80, 0.95$ and $\alpha = 0.90, 0.99$. Since the average objective value at $\alpha = 0.90$ is least, this value is used for final experiments.

Cooling Rate ($\boldsymbol{\alpha}$)	**0.80**	**0.90**	**0.95**	**0.99**
Average objective value	15.63	11.9	12.22	22.3
Average deviation from best (%)	4.88	2.32	5.32	6.89
Average time (sec)	0.23595	0.355942	0.489558	0.904378

Table 2. Experimental results for different values of α

3. Number of iterations of SAGBMP ($\boldsymbol{iter}$): The algorithm needs to be terminated so that the process does not go on endlessly. We need to set some criterion for termination so that maximum exploration takes place in reasonable amount of time. An experiment is conducted to tune this parameter. The values of T, α and L are fixed as $5000, 0.90$ and 1000 respectively and $iter$ varies over $200, 500, 750$ and 1000. For each graph instance results are recorded over 30 trials and statistical tests are conducted. Two-way ANOVA with

replication is used for comparison and no significant difference is found. However, it is observed that computational time taken for $iter = 750$ and 1000 is relatively high. The graph in Figure 10 shows the time elapsed increases considerably with the increase in number of iterations, but the difference in average objective value is not significant. Therefore, $iter$ is taken as 500 for final experiments.

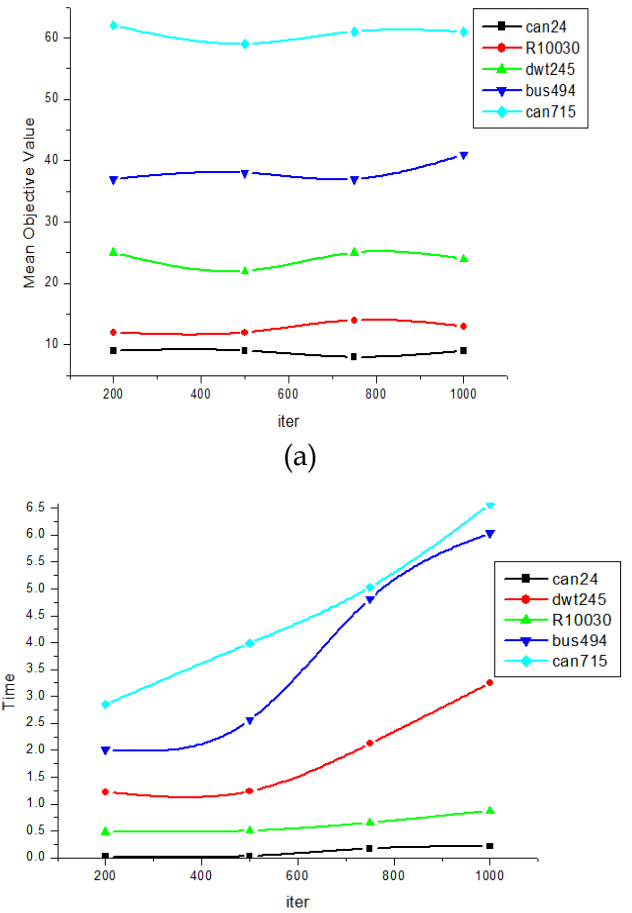

Figure 10. a) Average bandwidth and b) Time taken by instances for different values of $iter$.

4. Number of iterations at temperature $\boldsymbol{T}(\boldsymbol{L})$: The variable L is used to fix the number of perturbations to be performed at a certain temperature each time the temperature is reduced by the cooling factor. An experiment is conducted with fixed values of $T = 5000$, $\alpha = 0.90$ and $iter = 500$ and the value of L varies overthe set {500, 1000, 1500, 2000}. For each graphs instances results are recorded over 30 trials and statistical tests are conducted. Two-way ANOVA with replication is used for comparison. Again, no significant difference is found, but the algorithm takes longer to terminate at higher values of L. Therefore, we fix $L = 1000$.

5.2 Final Experiments

This section is dedicated to final experiments to evaluate the performance of SAGBMP. On the basis of the preliminary experiments conducted the final values of the parameters are tuned for final experimentation as $T = 5000$, $\alpha = 0.90$, $iter = 500$ and $L = 1000$. The test set consists of grid graphs and diverse graphs from the Harwell-Boeing Sparse Matrix Collection. The size of graphs varies from $10 \leq n \leq 1000$. Table 3 and 4 summarize the results obtained in the experiment of SAGBMP and the best objective value along with average time taken by the algorithm is reported.

Graphs	**|V|**	**Time(s)**
G2x4	8	10.40
G2x8	16	18.33
G2x16	32	32.00
G2x33	66	80.91
G2x51	102	95.87

Graphs	|V|	Time(s)
G2x62	124	111.26
G2x76	152	165.62
G2x84	168	264.63
G2x95	190	298.11
G2x100	200	381.01

Table 3: Results obtained by SAGBMP for Grid Graphs

Graphs	$\|V\|$	Objective Value	Average Time(s)	Graphs	n	Objective Value	Average Time(s)
can24	24	4	31.44	ash292	292	19	382.52
pores_1	30	6	39.30	can_292	292	10	380.23
ibm32	32	7	41.92	dwt_310	310	15	406.10
bcspwr01	39	6	51.09	gre_343	343	14	449.33

bcsstk01	48	8	62.88	dwt_361	361	13	472.91
bcspwr02	49	8	64.19	str_200	363	14	475.53
curtis54	54	7	70.74	dwt_419	419	16	548.89
will57	57	7	74.67	bcsstk06	420	18	550.20
dwt_59	59	7	77.29	bcsstm07	420	17	551.00
impcol_b	59	8	75.99	impcol_d	425	18	556.75
can_61	61	5	79.91	bcspwr05	443	22	580.33
bfw62a	62	9	81.22	can_445	445	17	582.95
bfw62b	62	8	77.51	nos5	468	20	613.08
can_62	62	5	66.69	west0479	479	19	627.49
bcsstk02	66	6	86.46	bcsstk020	485	19	635.35
dwt_66	66	6	87.15	mbeause	492	15	644.52
dwt_72	72	8	94.32	bus494	494	21	647.14

can_73	73	7	95.63	mbeacxc	496	19	649.76
steam3	80	9	104.8	mbeaflw	496	18	652.10
ash85	85	10	111.35	dwt_503	503	13	658.93
dwt_87	87	6	113.97	lns_511	511	11	669.41
can_96	96	11	125.76	gre_512	512	16	670.72
nos4	100	8	131.01	pores_3	532	20	696.92
gent113	113	9	148.03	fs_541_1	541	19	708.71
gre_115	115	9	150.65	dwt_592	592	21	775.52
bcspwr03	118	10	154.58	steam2	600	22	786.00
arc130	130	12	170.30	west0655	655	19	858.05
hor_131	131	11	171.61	bus662	662	25	867.22
lns_131	131	13	175.02	shl_200	663	19	868.53
bcsstk04	132	15	172.92	nnc666	666	19	872.46

west0132	132	11	172.11	fs_680_1	680	19	890.80
impcol_c	137	11	179.47	bus685	685	20	897.35
bcsstk22	138	9	180.78	can_715	715	20	936.65
can_144	144	10	188.64	nos7	729	28	954.99
bcsstk05	153	13	200.43	fs_760_1	760	23	995.6
can_161	161	12	210.91	mcfe	765	32	1002.15
dwt_162	162	15	212.22	bcsstk19	817	20	1070.27
west0167	167	14	218.77	bp_0	822	18	1076.82
mcca	180	16	235.80	bp_1000	822	22	1101.48
fs_183_1	183	18	239.73	bp_1200	822	23	1099.00
gre_185	185	9	242.35	bp_1400	822	24	1102.59
can_187	187	10	244.97	bp_1600	822	25	1121.08
dwt_193	193	10	252.83	bp_200	822	23	1085.04

will199	199	9	260.69	bp_400	822	23	1086.89
impcol_a	207	9	271.17	bp_600	822	22	1093.26
dwt_209	209	10	273.79	bp_800	822	20	1093.98
gre_216a	216	9	282.96	can_838	838	21	1097.78
dwt_221	221	10	289.51	dwt_878	878	36	1150.18
impcol_e	225	9	294.75	orsirr_2	886	31	1160.66
can_229	229	15	299.99	gr_30_30	900	25	1179.00
dwt_234	234	13	306.54	dwt_918	918	27	1202.58
nos1	237	11	310.47	jagmesh1	936	26	1226.16
saylr1	238	11	311.78	nos2	957	29	1253.67
steam1	240	20	314.40	nos3	960	30	1440.0
dwt_245	245	16	320.95	west0989	989	29	1483.51
can_256	256	15	335.36	jpwh_991	991	33	1486.5

nnc261	261	16	341.91	dwt_992	992	35	1488.08
lshp265	265	14	347.15	saylr3	1000	25	1511.51
can_268	268	14	351.08	sherman1	1000	27	1500.00
bcspwr04	274	18	358.94	sherman4	1104	35	1656.01

Table 4. Results obtained by SAGBMP for Harwell-Boeing instances

The final experiments show that SAGBMP is capable of achieving satisfactory results on all instances that are tested. The optimal value for grid graphs is known to be one. The algorithm is tested for grid graphs and for all the graphs the optimal values is achieved (Table 3). For Harwell Boeing graphs the objective values for all tested instances lie within bounds (Table 4).

6. Conclusion

In this paper, a simulated annealing algorithm is developed for the bandwidth of a two-dimensional embedding. Problem specific constructions are designed to obtain initial solutions and problem specific neighbourhood operators are designed as they are useful in satisfactory diversification in the search space. An experiment is conducted to compare the construction heuristics in which H2 outperforms $H1$. The parameters of SA are tuned by performing preliminary experiments on a sample of 15. SAGBMP is able to obtain optimal results for embedding grids onto grids. Forinstances For instances in for which optimal results are not known the algorithm produces results that may not be optimal but lie well within bounds. For future work, other single solution based and population-based metaheuristics and problem specific heuristics can be developed. The work can also be extended for embedding the graphs on $m \times n$ grids.

References

[1] Chung, Fan RK. "Labelings of graphs." Selected topics in graph theory 3, no. 25 (1988): 151-168.

[2] Dıaz, Josep, Jordi Petit, and Marıa Serna. "A Survey on Graph Layout Problems." (2000).https://doi.org/10.1145/568522.568523

[3] Rodríguez-García, Miguel Ángel, Jesus Sanchez-Oro, Eduardo Rodriguez-Tello, Eric Monfroy, and Abraham Duarte. "Two-dimensional bandwidth minimization problem: Exact and heuristic approaches." Knowledge-Based Systems 214 (2021): 106651.https://doi.org/10.1016/j.knosys.2020.106651

[4] Miller, Zevi, and James B. Orlin. "NP-completeness for minimizing maximum edge length in grid embeddings." Journal of algorithms 6, no. 1 (1985): 10-16.https://doi.org/10.1016/0196-6774(85)90016-1

[5] Hong, Jia-Wei, Kurt Mehlhorn, and Arnold L. Rosenberg. "Cost trade-offs in graph embeddings, with applications." Journal of the ACM (JACM) 30, no. 4 (1983): 709-728.

[6] Bezrukov, Sergei L., Joe D. Chavez, Larry H. Harper, Markus Röttger, and U-P. Schroeder. "Embedding of hypercubes into grids." In International Symposium on Mathematical Foundations of Computer Science, pp. 693-701. Springer, Berlin, Heidelberg, 1998.https://doi.org/10.1007/BFb0055820

[7] Lin, Y. "On density lower bounds of two-dimensional bandwidth." In Journal Of Mathematical Research And Exposition-Chinese Edition-, vol. 16, pp. 343-349. Mathematical Research and Exposition, 1996.

[8] Lin, Lan, and Yixun Lin. "Square-root rule of two-dimensional bandwidth problem∗." RAIRO-Theoretical Informatics and Applications 45, no. 4 (2011): 399-411.https://doi.org/10.1051/ita/2011120

[9] Kirkpatrick, Scott. "Optimization by simulated annealing: Quantitative studies." Journal of statistical physics 34, no. 5 (1984): 975-986.https://doi.org/10.1007/BF01009452

[10] Torres-Jimenez, Jose, Idelfonso Izquierdo-Marquez, Alberto Garcia-Robledo, Aldo Gonzalez-Gomez, Javier Bernal, and Raghu N. Kacker. "A dual representation simulated annealing algorithm for the bandwidth minimization problem on graphs." Information Sciences 303 (2015): 33-49.https://doi.org/10.1016/j.ins.2014.12.041