

# Branch and Bound Algorithm for Vertex Bisection Minimization Problem

Pallavi Jain, Gur Saran and Kamal Srivastava

**Abstract** Vertex Bisection Minimization problem (VBMP) consists of partitioning the vertex set  $V$  of a graph  $G = (V, E)$  into two sets  $B$  and  $B'$  where  $|B| = \lfloor |V|/2 \rfloor$  such that its vertex width ( $VW$ ) is minimized. Vertex width is defined as the number of vertices in  $B$  which are adjacent to at least one vertex in  $B'$ . It is an NP-complete problem in general but polynomially solvable for trees and hypercubes. VBMP has applications in fault tolerance and is related to the complexity of sending messages to processors in interconnection networks via vertex disjoint paths. In this paper, we propose a branch and bound algorithm for VBMP which uses a greedy heuristic to determine upper bound for the vertex width. We have devised a strategy to obtain lower bounds on the vertex width of partial solutions. A tree pruning procedure which reduces the size of search tree is also incorporated into the algorithm. This algorithm has been experimented on selected benchmark graphs. Results indicate that except for five of the selected graphs, the algorithm is able to, run through the search tree very fast.

**Keywords** Vertex bisection · Branch and bound · Vertex width · NP-complete · Graph layout

## 1 Introduction

Vertex Bisection Minimization problem (VBMP) consists of partitioning a vertex set of a connected graph  $G = (V, E)$ ,  $|V| = n$ , into two sets  $B$  and  $B'$  where  $|B| = \lfloor n/2 \rfloor$  such that vertex width ( $VW$ ) is minimized where vertex width is

---

P. Jain (✉) · G. Saran · K. Srivastava  
Department of Mathematics, Faculty of Science, Dayalbagh Educational Institute, Agra, India  
e-mail: pallavijain.t.cms@gmail.com

G. Saran  
e-mail: gursaran@dei.ac.in

K. Srivastava  
e-mail: kamal.sri@dei.ac.in

defined as the number of vertices in  $B$  which are adjacent to at least one vertex in  $B'$ . Formally, for a partition  $P = (B, B')$ , its vertex width is  $VW(G, P) = |\{u \in B: \exists v \in B' \wedge (u, v) \in E(G)\}|$ . VBMP is to find a partition  $P^*$  such that  $VW(G, P^*) = \min_{\text{partition } P} VW(G, P)$ . This problem has been treated as a graph layout problem by Diaz et al. [1]. VBMP is relevant to fault tolerance and is related to the complexity of sending messages to processors in interconnection networks via vertex disjoint paths [1]. NP-completeness of VBMP for general graphs is proved by Brandes and Fleischer [2] but it is shown to be polynomially solvable for trees and hypercubes. In spite of its practical applications, this problem remains almost unstudied so far. However, Fraire et al. [3] recently proposed two Integer Linear Programming (ILP) models and a branch and bound (B&B) algorithm for VBMP. They observed that ILP2 is the most promising method. In B&B, they have not used or provided any approach for determining the lower bound of the partial solutions. Their implementation also does not include any procedure for tree pruning. Therefore, it turns out to be an enumerative technique which is able to solve only 2 instances of graphs out of 108 instanced tested by them.

In this paper, we present a comprehensive B&B algorithm for VBMP. In order to generate an initial upper bound, a greedy heuristic has been designed (Sect. 4). A good heuristic which gives solution close to the optimal solution is always preferred as it is responsible for fathoming nodes at each level of the search tree in the B&B algorithm. We have also devised a strategy for finding lower bound of nodes representing partial solutions (defined in Sect. 3) at each level of the tree. Besides this, a procedure for tree pruning which helps to discard a large number of nodes in the search tree has also been designed (Sect. 5). The search tree is explored using depth first strategy. The proposed B&B algorithm (BBVBMP) is simulated on a large number of graphs including Small graphs, grids, trees and Harwell-Boeing graphs [3] and the algorithm is able to run through the search tree very fast except for Grid6×6, Grid7×7, bcsprw01, bcsprw02 and bcsstk1. We have also compared BBVBMP with ILP2 (Sect. 6). Conclusion is presented in Sect. 7.

## 2 Branch and Bound Algorithm

Branch and Bound Algorithm (B&B) is an exact combinatorial approach. It generates and explores the entire set of solutions to the problem by examining a search tree. It starts by generating an initial solution using some heuristic, or randomly, whose objective function value serves as upper bound of the optimal value. During the search process, it finds a lower bound at each node of the search tree. If this lower bound is greater than or equal to the upper bound then this node is fathomed because it guarantees that this node cannot result in a better solution. When exploration reaches a leaf node, it computes the objective function value of the corresponding solution and updates the upper bound if necessary. The B&B algorithm stops when all the nodes have been explored (either branched or fathomed), and returns the

optimal solution to the problem [4]. B&B is an important class of algorithms and has been applied to a diversity of problems [4, 5, 6, 7].

In the context of VBMP, the search tree starts at the root which is an empty set. The tree branches into  $\lceil n/2 \rceil + 1$  nodes from the root where each node is a singleton consisting of vertices from 1 to  $\lceil n/2 \rceil + 1$  each. This forms level 1 of the search tree. At level 2, each node of level 1 is branched into nodes which now contain two vertices. Thus number of vertices in the nodes of level  $i+1$  is one more than those at level  $i$ . Finally, level  $\lfloor n/2 \rfloor$  consists of leaves each containing  $\lfloor n/2 \rfloor$  vertices. In order to avoid duplicate nodes at each level, a pruning procedure is used which is described in Sect. 5. The tree is explored using depth first search strategy.

### 3 Lower Bound for Partial Solutions

Each node of the B&B tree represents a set  $S \subseteq V$ . This set will eventually be extended to a collection  $G$  of sets such that each  $B$  in  $G$  gives vertex bisection  $(B, V \setminus B)$ . We will refer to the set  $S$  itself as a partial solution hereafter. The set of adjacent vertices (neighbors) of  $u \in V$  is denoted by  $N(u) = \{v \in V : uv \in E\}$ . The procedure *Compute\_lowerbound*( $S$ ) (Fig. 1) is based on identifying those vertices of  $S$  whose neighbors cannot all be accommodated in  $B$ . In this strategy, number of vertices for which all the neighbors cannot be included in set  $B$  is identified.

**Proposition 1**  $|count \cup P|$  in the procedure *Compute\_lowerbound*( $S$ ) gives a lower bound for a node  $S$ .

*Proof* In the procedure, *count* is the set of vertices whose all the neighbors cannot be included in the solution generated by  $S$  at the lower level of the search tree because the number of adjacent vertices is more than  $\lfloor n/2 \rfloor - |S|$ . Therefore,  $|count|$  number of vertices will always contribute to the vertex width of the partition generated by  $S$ . It is clear that in order to minimize the contribution of a vertex to the vertex width, it is required to place all the adjacent vertices in the same partition

**Fig. 1** Procedure for computing lower bound

```

Procedure Compute_lowerbound( $S$ )
Step 1:  $count = \{v \in S : |N(v) \setminus S| > (\lfloor n/2 \rfloor - |S|)\}$ 
Step 2:  $A = S \setminus count$ 
Step 3:  $P = \emptyset$ 
Step 4: for each vertex  $u \in A$ 
Step 5:    $P' = \emptyset$ 
Step 6:    $C = S \cup N(u)$ 
Step 7:    $P' = \{v \in A : |N(v) \setminus C| > (\lfloor n/2 \rfloor - |C|)\}$ 
Step 8:   if  $|P' \cup P| < |P|$  or  $P = \emptyset$ 
Step 9:      $P = P'$ 
Step 10:  endif
Step 11: endfor
Step 12:  $lb = \max(|count \cup P|, 1)$ 

```

either in  $B$  or  $B'$ . Thus to accomplish this, for each vertex  $v \in S_{count} = A$ ,  $N(v)$  is placed in  $S$  to give  $C$  and the number of vertices in  $S$  whose all the neighbors cannot be included in  $C$  are counted (Steps 6–7). Since  $P$  (Steps 8–9) records the smallest subset of  $S_{count}$  of which not all neighbors can be accommodated in  $B$ ,  $|count \cup P|$  represents the smallest number of elements of  $S$  whose neighbors cannot all be included in the complete set  $B$ . Hence, minimum number of vertices which will contribute to the vertex width have been considered which provides a lower bound guaranteeing that the solution generated by expanding the partial solution  $S$  will always give the vertex width greater than this lower bound.  $\square$

## 4 Initial Upper Bound

We have designed a greedy heuristic to generate a solution which can give a close upper bound for the vertex bisection minimization problem. Pseudocode for this heuristic is presented in Fig. 2.

## 5 Tree Pruning

In this section, we describe the method of branching a node along with tree pruning. We have adopted two strategies for tree pruning.

1. The main idea for pruning is that at each level of tree, all those nodes are discarded which give duplicate nodes at this level or will give duplicate nodes in

### Procedure *H1*

```

Step 1:  $B = \phi$ 
Step 2:  $u =$  minimum degree vertex in  $G$ 
Step 3: if  $|N(u)| < \lfloor n/2 \rfloor$ 
Step 4:    $B = \{u\} \cup N(u)$ 
Step 5: else
Step 6:   Randomly select  $X \subset N(u)$  s.t.  $|X| = \lfloor n/2 \rfloor - 1$ 
Step 7:    $B = \{u\} \cup X$ 
Step 8: endif
Step 9: while  $|B| < \lfloor n/2 \rfloor$ 
Step 10:  select a vertex  $v$  from  $B$  s.t.  $|N(v) \setminus B| \neq 0$  is
          minimum
Step 11:   $Y = N(v) \setminus B$ 
Step 12:  if  $|Y| \leq \lfloor n/2 \rfloor - |B|$ 
Step 13:     $B = B \cup Y$ 
Step 14:  else
Step 15:     $B = B \cup Z$  where  $Z$  is a randomly selected sub
          set of  $Y$  s.t.  $|Z| = \lfloor n/2 \rfloor - |B|$ 
Step 16:  endif
Step 17: endwhile

```

**Fig. 2** Pseudocode of heuristic *H1*

**Fig. 3** Procedure for branching a tree

```

Procedure Branch&Prune(Node)
Step 1:  $R = \{v \in V(G) \setminus \text{Node} : v > \max(\text{Node})\}$ 
Step 2: for  $i = 1: |R|$ 
Step 3:    $P = \{v \in V(G) : v > R[i]\}$ 
Step 4:   if  $|P| \geq \lfloor n/2 \rfloor - |\text{Node} \cup \{R[i]\}|$ 
Step 5:      $\text{Node}_i = \text{Node} \cup \{R[i]\}$ 
Step 6:   endif
Step 7: endfor
Step 8:  $\text{branches} = \{\text{Node}_i\}$ 

```

further branching at lower levels. Procedure *Branch&Prune(Node)* outlines the method for branching a node in the search tree without duplicate nodes (Fig. 3). Let the node to be branched be represented by an array *Node* representing the partial solution *S*. Now only those vertices  $v \in V$  are considered for which  $v > p$  where  $p = \max\{w \in \text{Node}\}$ . Let this set of vertices be represented by *R*. If the number of vertices having identifiers greater than  $R[i]$  is more than  $\lfloor n/2 \rfloor - |\text{Node} \cup \{R[i]\}|$ , then *Node* is branched into  $\text{Node} \cup \{R[i]\}$  otherwise it is not branched (Steps 3–6). In this manner the initial node is branched into  $\lfloor n/2 \rfloor$  nodes  $1, 2, \dots, \lfloor n/2 \rfloor + 1$  each with a single different vertex. It may be noted that in VBMP maximum number of possible different leaves is  $\binom{n}{\lfloor n/2 \rfloor} C$ .

2. A node is fathomed if lower bound (*lb*)  $\geq$  upper bound (*UB*).

**Proposition 2** *Procedure Branch&Prune(Node) (Fig. 3) eliminates the duplicate nodes of the same level of the search tree and the nodes which will give duplicate nodes in further branching.*

*Proof* In Step 1, only those vertices  $v$  are considered for which  $v > \max\{w \in \text{Node}\}$  because the combination with the smaller ones is already present in their sibling nodes. Thus all the duplicate nodes at a level are eliminated. In Step 3, let  $v$  be the vertex considered for the branching. The set *P* consists of all those vertices  $w$  such that  $w > v$ . If  $|P| \geq \lfloor n/2 \rfloor - |\text{Node} \cup \{R[i]\}|$  then it guarantees that this node will branch into at least one distinct node, otherwise, at any level *Node* will include a vertex whose identifier is less than  $\max\{w \in \text{Node}\}$  which will result into a duplicate node at that level as this combination has already been taken. Hence the result follows.  $\square$

## 6 Computational Experiments

This section describes the computational experiments performed to test the efficiency of our B&B algorithm. The algorithm has been implemented in C++ and the experiments are conducted on a Dual Xeon workstation with 12 GB RAM. The experiments have been conducted on four sets of instances: grids, trees, small graphs and harwell-boeing (HB) graphs [3]. The maximum time for the grid, tree and small instances was set to 5 min. while for HB instances time limit is 1 h as in

**Table 1** Comparison between ILP2 and BBVBMP

Instances		Grids	Small	Trees	HB	Total
Number of instances		5	84	15	4	108
CPU (s)	ILP2	907.021	1598.902	73.198	14402.21	16981.33
	BBVBMP	120	16.63095	19.9	10802	10958.53
Number of optimal solutions	ILP2	2	69	15	0	86
	BBVBMP	3	84	15	1	103
Percentage of optimal solutions (%)	ILP2	40	82.1429	100	0	79.6296
	BBVBMP	60	100	100	25	95.37

[3]. Table 1 compares ILP2 and BBVBMP in terms of number of optimal solutions obtained and the CPU time.

For Grid6×6 and Grid7×7, tree was not explored completely. In HB graphs, the tree was explored completely only for can24. For the other graphs the tree was not explored completely in the specified time. Therefore, we cannot guarantee the optimality of result in these cases. Table 1 shows that BBVBMP outperforms ILP2 in terms of both the number of optimal solutions and CPU time.

## 7 Conclusion

An exact procedure based on B&B algorithm for vertex bisection minimization problem is developed in this paper. We have proposed a strategy for finding lower bound at each level of the tree and also a strategy for tree pruning. These strategies allow us to explore a smaller portion of the search tree. We have proposed a strategy for obtaining upper bound which is able to achieve optimal results for a large number of standard graphs for which experiments have been performed. It has been clearly shown that BBVBMP outperforms the state-of-art exact method for VBMP.

## References

1. Diaz, J., Petit, J., Serna, M.: A survey of graph layout problems. *ACM Comput. Surv.* **34**, 313–356 (2002)
2. Brandes, U., Fleischer, D.: Vertex bisection is hard, too. *J. Graph Algorithms Appl.* **13**, 119–131 (2009)
3. Fraire, H., David, J., Villanueva, T., Garcia, N.C., Barbosa, J.J.G., Angel, E.R. del, Rojas, Y.G.: Exact methods for the vertex bisection problem. *Recent Adv. Hybrid Approaches Des. Intell. Syst. Stud. Comput Intell.* 547, 567–577 (2014)
4. Marti, R., Pantrigo, J.J., Duarte, A., Pardo, E.G.: Branch and bound for the cutwidth minimization problem. *Comput. Oper. Res.* **40**, 137–149 (2014)

5. McCreesh, C., Prosser, P.: A parallel branch and bound algorithm for the maximum labelled clique problem. *Optim. Lett.* **9**, 949–960 (2015)
6. Vlachou, A., Christos, D., Kjetil, N., Yannis, K.: Branch-and-bound algorithm for reverse top-k queries. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 481–492. ACM Press, New York (2013)
7. Delling, D., Fleischman, D., Goldberg, A.V., Razenshteyn, I., Werneck, R.F.: An exact combinatorial algorithm for minimum graph bisection. *Math. Program.* **153**, 417–458 (2015)