Exact and heuristic methods for the vertex separator problem

Haeder Althoby, Mohamed Didi Biha, André Sesboüé

Exact and heuristic methods for the vertex separator problem

Haeder Y. Althoby[1,2], Mohamed Didi Biha[1] and André Sesboüé[1]

[1] Normandie Université ,UNICAEN,Labratoire des Mathématiques Nicolas Oresm CNRS UMR 6139, 14032 Caen Cedex ,France.
mohamed.didi-biha, andre.sesboue@unicaen.fr

[2] Dept. of Math., College of Science, AL-Qadisiyah University, AL-Diwaniyah,Iraq.
hadiryounis@gmail.com

Abstract

In this paper, we propose a pratical and efficient methods to solve the vertex separator problem (VSP for short), based on branch-and-bound procedure, which uses linear programming, and a greedy algorithm. This problem arises in many areas of applications such as graph algorithms, communication networks, solving systems of equations, finite element and finite difference problems. We show, by computational experiments, that our approach is able to solve in short time large-scale instances of VSP from the literature to optimality or near optimality.

Key words: Graph partitioning, Vertex separator, Integer programming, Branch-and-Bound, Greedy algorithm.

1 Introduction

The graphs we consider are finite, undirected and connected. We denote a graph by $G = (V, E)$, where V is the node set and E is the edge set. Let $G = (V, E)$ be a graph and $\beta(n)$ be a positive integer, where $n = |V|$. The vertex separator problem (VSP for short) is to find a partition of V into three nonempty classes A, B, and C such that:

(i) There is no edge between A and B;

(ii) $\max\{|A|, |B|\} \leq \beta(n)$;

(iii) $|C|$ is minimum.

The subset C is called a *separator*. The subsets A and B are called *shores* of the separator C. For convenience, a partition $\{A, B, C\}$ of V which satisfies (i) and (ii) will be also called a separator.

The VSP appears in wide range of applications. In the field of graph algorithms, the computation of balanced small-sized separators is very useful, especially for divide-and-conquer algorithms. In communication networks, a separator is seen as a bottleneck when a graph represents the network. To check the capacity and brittleness of a network, the separator is used to find bounds and brittle nodes. In bioinformatics and computational biology, separators are wanted in grid graphs providing a simplified representation of proteins.

The VSP is NP-hard [3, 11] and still NP-hard in planar graphs [9].

When $\beta(n) = n - k$ for some positive constant k, the problem becomes polynomially solvable [1]. As it was mentionned in [1], the VSP is trivial if $\beta(n) = 1$ and it is polynomially solvable if $\beta(n) \geq n - 1$.

A few exact algorithms have been developped for VSP. They are based on branch-and-cut [1, 20], branch-and-bound [7], lagrangian relaxation [5, 4]. Theses methods achieve good performance for instances with up 150 vertices, but fail to solve larger instances. Many heuristics have been proposed to obtain good solutions for large VSP instances. Benlic and Hao [2] and Zhang and Shao[23] used breakout local search BLS. Jesús, Juan and Abraham [17] used four shake algorithms and embed them into a Variable Neighborhood Search scheme. Sanchez-Oro et al. [19] introduced several Variable Neighborhood Search(VNS) algorithms, which alternate between a local search phase and a shaking phase. Hager et al. [15] proposed a continuous optimization approach.

The article is organised as follows. In the next section, we present the mixed linear integer programming introduced by [1] for VSP and we show how we can strengthen it's linear relaxation. In Section 3, we present a simple greedy algorithm to solve VSP. Our computational results are presented in Section 4, and finally some concluding remarks are given in Section 5.

2 Mixed integer linear programming formulation

Let $G = (V, E)$ be an undirected graph and $\beta(n)$ be a positive integer, where $n = |V|$. For every separator $\{A, B, C\}$ we associate an incidence vector

$(x, y) \in \mathbb{R}^{2n}$ defined by $x_v = 1$ if $v \in A$ and 0 otherwise, and $y_v = 1$ if $v \in B$ and 0 otherwise. In [1], Balas and de Souza give the following mixed integer linear prrogram for VSP:

$$(MIP)\begin{cases} \max \sum_{v\in V}(x_v + y_v) & \\ x_u + y_v \leq 1 & \forall (uv) \in E \\ x_v + y_u \leq 1 & \forall (uv) \in E \\ x_v + y_v \leq 1 & \forall v \in V \\ 1 \leq \sum_{v\in V} y_v \leq \beta(n) & \\ 1 \leq \sum_{v\in V} x_v \leq \beta(n) & \\ x_v \in \{0, 1\} & \forall v \in V \\ y_v \geq 0 & \forall v \in V \end{cases}$$

First and second inequalities come from the fact that there is no edge between A and B. Third inequalities come from the fact that $A \cap B = \emptyset$. Fourth and fifth inequalities come from the fact that $A \neq \emptyset \neq B$ and $\max\{|A|, |B|\} \leq \beta(n)$.

In general, the linear relaxation of (MIP) is very weak and not appropriate to obtain a good upper bound for VSP. Let u and v be tow non-adjacent nodes. Denote by α_{uv} the maximum number of node-disjoint paths between u and v. Define

$$\alpha = \min\{\alpha_{uv} : u, v \in V, (uv) \notin E\}.$$

We note that for any path between any two vertices $a \in A$ and $b \in B$, then C have at least one vertex in common. Therefore $|C| \geq \alpha$. Thus, α is as a lower bound of the cardinality of any separator. In order to strengthen the linear relaxation of (MIP), Didi Biha and Meurs [7] added the following valid inequality:

$$\sum_{v\in V}(x_v + y_v) \leq n - \alpha \tag{2.1}$$

The calculation of α can be done by solving p maximum flow problems, where $p = |\{(uv) : u, v \in V,\ (uv) \notin E\}|$, and thus can be achieved in polynomial time by various maximum flow algorithms [6, 8, 10, 12, 13]. For a survey of efficient maximum flow algorithms see Goldberg and Tarjan [14].
For two non-adjacent vertices i and j the calculation of α_{ij} is described below. We first construct a directed graph from G by replacing each edge of E by

two parallel arcs of opposite direction, then we replace any vertex u by two vertices u', u'' and make an arc $(u'u'')$ with capacity equal to 1. Finally, we replace every arc (uv) by the arc $(u''v')$ with infinite capacity. Let α_{ij} be the value of the maximum $i-j$-flow in this network. Then, by Menger's theorem, α_{ij} is equal to the maximum number of vertex-disjoint paths between u and v (see Schrijver's book for more details [21]) .

For small and medium instances, with up 200 vertices, inequality (2.1) will plays an important role in the successful computational experiments reported in Section 4. For large instances, this inequality make a significant improvement in the quality of the upper bound obtained by the branch-and bound procedure based on the formulation (MIP). It is obvious that when α is small then the inequality (2.1) is weak. We can improve it by replacing α by $\bar{\alpha} \geq \alpha$. In the following, we will explain how we obtain such $\bar{\alpha}$. For every $u \in V$, define $\alpha_u = \min\{\alpha_{uv} : v \in V, (uv) \notin E\}$. Without loss of generality, we suppose that $V = \{u_1, \ldots, u_n\}$ and $\alpha_{u_1} \leq \alpha_{u_2} \leq \ldots \leq \alpha_{u_n}$.

Remark 2.1 *If (A, B, C) is a separator, then $|C| \geq \max\{\alpha_u : u \in A \cup B\}$.*

Let $(A^\star, B^\star, C^\star)$ be a solution of VSP, for example, the solution obtained from the greedy algorithm presented in the next section. If $(\bar{A}, \bar{B}, \bar{C})$ is a solution of VSP such that $|\bar{C}| < |C^\star|$, or in equivalent manner $|\bar{A}| + |\bar{B}| > |A^\star| + |B^\star|$, then we must have $u_k \in \bar{A} \cup \bar{B}$ for some $k \geq |A^\star| + |B^\star| + 1$. Thus, by Remark 2.1, $|\bar{C}| \geq \alpha_k \geq \alpha_{|A^\star|+|B^\star|+1}$. Thus, a suitable value of $\bar{\alpha}$ is given by $\alpha_{|A^\star|+|B^\star|+1}$ and we can replace (2.1) by the following inequality :

$$\sum_{v \in V} (x_v + y_v) \leq n - \bar{\alpha} \tag{2.2}$$

3 A simple greedy algorithm

Le $G = (V, E)$ be a graphe. The *neighborhood* of a vertex $u \in V$, denoted by $N(u)$, is the set of vertices $v \in V$ such that $(uv) \in E$. The neighborhood of a subset $A \subseteq V$, denoted by $N(A)$, is the set of vertices $v \in V \setminus A$ such that $(uv) \in E$ for some $u \in A$.

We build a solution (A, B, C) of VSP as follows: to initialize the algorithm, choose a vertex $a \in V$ with minimum degree in G, set $A = \{a\}$, $C = N(a)$ and $B = V \setminus (A \cup C)$. In the second step, choose a vertex $i \notin A$ with

minimum neighborhood in $B \setminus \{i\}$ (*i.e.,* $|N(i) \cap B|$ is minimum). Set $A = A \cup \{i\}$, $C = N(A)$ and $B = V \setminus (A \cup C)$, and so on until satisfy the condition $|A| + |C| \geq n - \beta(n)$. The last condition guarantees that, at the end of the algorithm, (A, B, C) is a separator such that $\max\{|A|, |B|\} \leq \beta(n)$. We denote this algorithm by GA (for Greedy Algorithm).
The following code specifies the algorithm more formally.

Algorithm 1 GA

1: *choose a vertex a with minimum degree.*
2: $A = \{a\}$; $C = N(A)$; $B = V \setminus (A \cup C)$.
3: **while** $|A| + |C| < n - \beta$ **do**
4: *Let* $i \in V \setminus A$ *such that* $|N(i) \cap B|$ *is minimum* .
5: $A = A \cup \{i\}; C = N(A); B = V \setminus (A \cup C)$.
6: *End while*

Notice that since the while loop runs at most $n - \beta$ times, the running time of this heuristic is $O(n)$.

4 Computational results and discussion

Our numerical test were programmed in C++ and compiled with GNU g++ under GNU/Windows 7 running on a Dell laptop with an Intel Core i7-4800MQ with 2.70 GHz and 8 GB of RAM. In order to compare our approach with the state-of-the-art, they have been performed on the same instances considered by De Souza and Cavalcante [4] and Benlic and Hao [2]. De Souza and Cavalcante consider 62 instances with a number of vertices between 73 and 212. These instances can be download from [25]. Benlic and Hao consider 54 more challenging graphs generated by Helmberg and Rendl [16]. This benchmark consists of toroidal, planar, and random graph instances ranging from 800 to 3000 vertices. These instances can be download from [27]. Based on the arguments for the development of efficient divide-and-conquer algorithms [18], the value of the parameter $\beta(n)$ is set to $\lfloor \frac{2n}{3} \rfloor$, which is more used in literature.

We apply branch-and-bound procedure (BB) and algorithm GA to the instances mentioned above. For instances with more than 250 vertices, branch-and-bound procedure fails to obtain optimal solutions. However, it allows us ton obtain good upper bounds. The branch-and-bound procedure has been conducted in two steps. We first calculated the parameter α, the lower bound of any cardinality of any separator. This achived by calculating p max-flow problems, where $p = |\{(uv) : u, v \in V,\ (uv) \notin E\}|$. We use the Lemon library

[24] in order to solve the max-flow problems. After adding inequality (2.2), we then solve the mixed-integer program (MIP) introduced in Section 2 using Ilog-CPLEX 12.6 [26].

The results of our computational experiments are presented in Tables 1,2,3 and 4, where each line corresponds to one specific instance.

Table 1 summarizes our results compared with those obtained by Cavalcante and de Souza [4]. For every instance, we compare our results with the best one among the four algorithms developped by Cavalcante and de Souza. The first column contains the instance name. The second column contains the number of vertices of the graph. The third column contains the value of the objective function obtained by the heuristic GA. The fourth column contains the value of the objective function obtained by the branch-and-bound procedure (BB), actually, the optimal solution. The fifth column gives the running time of BB in seconds. The two last columns contain respectively the best values of the objective function obtained by Cavalcante and de Souza and the corresponding running times. We didn't record the running time of the heuristic GA. In fact, for all instances, this time is less than 0.05 second.

With the branch-and-bound procedure, we were able to find the optimal solution for all instances in small time. The only exception being instance dim.DSJC125.1 for wich the optimal solution was found after 61 minutes. These good results prove the importance of inequality (2.2). From Table 1, we see that 16 instances (the value in sixth column appears in bold) could not be solved by Cavalcante and de Souza. We recall that for every instance, to make a comparison with our result, we have choosen the best result obtained by Cavalcante and de Souza among the four algorithms they developped. If we consider every algorithm separately, we can observe in [4] that there is at least 27 instances for which their algorithm could not reach an optimal solution. We can observe also that they proved optimiality only for 4 instances. We note the good performance of our heuristic GA. It was able, in less than 0.05 second, to obtain an optimal solution for 34 instances (the value in third column is indicated with *) and quasi-optimal solution for the other instances.

Table 1: Comparison of our results with those obtained by Cavalcante and de Souza (62 instances)

Instances	$\|V\|$	***Heuristic GA***	*BB*	*time BB*	*CS*	*time CS*
dim.DSJC125.1	125	89	91	3666	**89**	4.12
dim.games120	120	99	102	5.23	**99**	1.67
dim.myciel7	191	156*	156	4.91	**155**	3.71
dim.myciel6	95	76*	76	0.58	**75**	1.18
dim.queenl2_12	144	97*	97	6.47	97	6.69
dim.Queenl1_11	121	81*	81	3.18	81	5.56
dim.Queenl0_10	100	67*	67	1.45	67	4.04
dim.queen8_12	96	65*	65	2.93	65	3.69
dim.Queen9_9	81	55*	55	0.9	55	2.73
dim.Queen8_8	64	43*	43	0.08	43	1.67
miles1000	128	110*	110	1.87	**109**	8.23
dim.Queen7_7	49	31*	31	0.06	31	0.9
dim.DSJC125.5	125	74*	74	3.29	74	5.11
dim.DSJC125.9	125	22*	22	0.58	22	5.73
mat.can96	96	72*	72	0.69	72	1.78
mat.can73	73	53*	53	0.53	53	1.60
mat.rw136	136	120	121	1.06	**120**	2.49
mat.gre_115	115	95*	95	0.87	**93**	3.45
mat.L125.gre_185	125	104*	104	1.45	104	4.64
mat_can144	144	126*	126	0.12	126	5.60
mat.L125.can_161	125	95	97	8.42	97	4.10
mat.lund_a	147	115	118	1.05	**116**	4.83
mat.L125.bcsstk05	125	99	101	1.08	101	3.48
mat.L125.dwt_193	125	94	95	1.30	95	3.49
mat.L125.fs-183-1	125	92	98	1.89	98	2.70
mat.bcsstk04	132	83	84	0.45	84	4.63
mat.arc130	130	88*	88	0.47	88	7.51
mat.L100.steam2	100	75	76	0.39	76	2.84
mat.L120.fidap025	120	100	102	0.25	102	2.57
mat.L120.cavity01	120	95	99	0.41	99	2.60
mat.L120.fidap021	120	93	98	0.34	98	2.84
mat.L120.rbs480a	120	87	88	0.75	88	3.40
mat.L120.wm2	120	96	98	2.67	**92**	1.73
mat.L100.rbs480a	100	72	73	0.45	73	2.26
mat.L80.wm2	80	60	61	1.09	61	1.96
mat.L100.wm3	100	75	77	13.79	77	2.43
mat.L120.e05r0000	120	85	90	0.19	90	2.39
mat.L100.wm1	100	74*	74	2.21	**73**	2.30
mat.L120.fidap022	120	84*	84	0.39	84	3.87
mat.L100.fidapm02	100	69*	69	0.77	69	2.28
mat.L120.fidap001	120	82*	82	0.48	82	4.08
mat.L100.e05r0000	100	69	70	0.16	70	1.92
mat.L80.fidapm02	80	53*	53	0.22	53	1.54
mat.L120.fidapm02	120	85	86	0.98	86	3.44
mat.L100.fidap001	100	64*	64	0.17	64	2.76
mat.L80.fdap001	80	52	54	0.16	54	1.40
mat.L100.fidap022	100	62*	62	0.33	62	2.83
mat.L80.fidap022	80	41*	41	0.09	41	1.65
mat.L100.fidap027	100	69*	69	1.69	69	2.48
mat.L100.fidap002	100	66*	66	1.19	66	1.91
mat.L120.fidap002	120	67	68	0.30	68	3.58
mat.L120.fidap027	120	83*	83	0.47	83	2.33
miplib.noswot.p	182	166	167	2.07	**146**	2.77
miplib.khb05250.p	100	75*	75	1.54	75	1.20
miplib.stein27r.p	118	62*	62	3.49	62	3.78
miplib.10teams.p	210	119	120	13.04	120	10.55
miplib.mod010.p	146	90*	90	61.84	**88**	3.99
miplib.1152lav.p	97	60	61	2.34	**60**	1.73
miplib.lp41.p	85	50*	50	3.28	**49**	4.32
miplib.air03.p	124	73	75	6.21	**74**	5.83
miplib.misc03.p	96	52*	52	3.70	52	2.81
miplib.misc07.p	212	113	116	43.9	**115**	20.19

Table 2: Comparision of GA with BLS algorithm on graphs generated by Helmberg and Rendl (26 instances).

Instances	$\|V\|$	*GA*	*time*	*BLS*	*time BLS*	*UB*
g1	800	543*	0.96	543	7.2	543
g2	800	543*	0.66	543	8.7	543
g3	800	543*	0.23	543	66.3	543
g5	800	543*	0.19	543	65.9	543
g6	800	543*	0.25	543	8.5	543
g7	800	543*	0.26	543	10.0	543
g8	800	543*	0.26	543	69.6	543
g9	800	543*	0.15	543	34.0	543
g10	800	543*	0.21	543	70.7	543
g11	800	784**	2.68	784	0.0	784
g12	800	768*	1.48	768	0.0	768
g13	800	755*	1.43	755	0.2	755
g14	800	628	1.22	654	768.7	680
g15	800	633	1.22	656	856.7	680
g16	800	626	1.17	656	188.6	680
g17	800	633	1.10	656	401.1	680
g18	800	628	1.18	654	623.5	680
g19	800	633	1.17	656	1085.8	680
g20	800	626	1.09	656	353.2	680
g21	800	634	1.11	656	552.5	680
g22	2000	1412	10.71	1412	657.7	1800
g23	2000	1410	10.40	1410	310.4	1800
g24	2000	1411	10.68	1411	371.5	1800
g25	2000	1411	10.56	1411	832.5	1800
g26	2000	1413	10.70	1413	313.9	1800
g33	2000	1950*	39.59	1950	0.2	1950

In table 2, we present our numerical tests on a subset of instances considered by Benlic and Hao [2]. In their experiments, the value of parameter $\beta(n)$ is set to $\lfloor \frac{2n}{3} \rfloor$ for 26 instances (presented in Table 2) and $\lfloor \frac{1.05 \times n}{2} \rfloor$ for 28 instances (presented in table 3). Obviously, the value of $\beta(n)$ plays a key role in the hardness or easiness of instances. The influence of this parameter is not the subject of this work and needs to be explored in further studies. Thus, we compare ours results with those obtained by Benlic and Hao only for instances for which the value of $\beta(n)$ is set to $\lfloor \frac{2n}{3} \rfloor$. Benlic and Hao proposed an nondertministic heuristic based on Breakout Local Seearch (BLS). For evey instance, they apply their heuristic over 100 runs. The results are presented in Table 2. The first column contains the name of the instance. The second column contains the number of vertices of the graph. The third column contains the values of objective function obtained by our heuristic GA. The fourth column contains the running time of GA in seconds. The fifth and sixth columns contain respectively the values of objective function obtained by Benlic and Hao and the corresponding running time. In order to evaluate the quality of solutions obtained we calculate for every instance an upper bound. This is

done by applying the branch-and-bound procedure within 3 hours. The upper bounds are reported in seventh column.
We see in Table 2 that the heuristic GA has a similare performance to BLS with less computation time. It was able to obtain an optimal solution for 13 instances (the value in third column is indicated with *). For the rest of instances, the percentage of optimality reached by the heuristic GA is more than 92% for 8 instances and more than 78% for 5 instances.

Table 3: Computational results for the second group of instances generated by Helmberg and Rendl (28 instances)

Instances	$\|V\|$	GA	$time$	UB
g4	800	543*	0.34	543
g27	2000	1412	10.18	1700
g28	2000	1410	10.29	1700
g29	2000	1410	10.52	1700
g30	2000	1410	10.60	1700
g31	2000	1411	10.36	1700
g32	2000	1960*	38.60	1960
g34	2000	1920*	38.09	1920
g35	2000	1580	32.12	1700
g36	2000	1580	32.32	1700
g37	2000	1576	31.91	1700
g38	2000	1573	31.60	1700
g39	2000	1580	32.74	1700
g40	2000	1580	32.74	1700
g41	2000	1576	31.6	1700
g42	2000	1578	31.66	1700
g43	1000	705	0.97	750
g44	1000	705	1.02	750
g45	1000	706	1.03	750
g46	1000	703	0.97	750
g47	1000	704	0.97	750
g48	3000	2900*	143.8	2900
g49	3000	2940*	145.7	2940
g50	3000	2950*	149.5	2950
g51	1000	788	2.92	850
g52	1000	783	2.88	850
g53	1000	789	2.93	850
g54	1000	786	2.77	850

In table 3, we present our computational results on the second class of graphs generated by Helmberg and Rendt. The first column contains the name of the instances. The second column contains the number of vertices of the graph. The third and forth columns contains respectively the results obtained by heuristic GA and the corresponding running times in seconds. The last column contains the values of upper bounds obtained as mentioned above. It appears from Table 3 that 6 instances have been solved to optimality (the

value in third column is indicated with *). For the others instances, we can see that the percentage of otimality reached by the heuristic is more than 83%.

In table 4, we present computational results on the set of 11 new instances which can serve as reference for future works. These instance come from the DIMACS challenge on graph coloring. The first column contains the instance name where all instances have 450 vertices. In order to illustrate that inequality (2.2) is an interesting improvement of inequality (2.1), the values of α and $\bar{\alpha}$ are respectively given in the second and third column. The fourth and fifth column contain respectively the values of objective function obtained by heuristic GA and the corresponding running times. The last column contains the upper bounds of the optimal solutions. We can see in Table 4 that one instance is solved to optimality (le450-25a) and for the other instances, the percentage of optimality reached by the heuristic is more than 96%.

Table 4: DIMACS instances (11 instances)

Instances	α	$\bar{\alpha}$	*GA*	*time GA*	*UB*
le450-5a	13	16	317	0.10	326
le450-5b	12	16	316	0.11	326
le450-5c	27	31	308	0.20	318
le450-5d	29	32	309	0.61	319
le450-15b	1	12	326	0.72	336
le450-15c	18	32	307	0.10	317
le450-15d	18	33	308	0.14	318
le450-25a	2	6	340*	0.62	340
le450-25b	2	8	337	0.14	347
le450-25c	7	21	311	0.20	322
le450-25d	11	23	311	0.41	321

5 Conclusions

In this article, we have considered the vertex separator problem. To resolve this problem, we have proposed a simple greedy heuristic and a branch-and-bound procedure based on the linear integer programming model introduced in [1]. We strengthen the relaxation of this model by adding an efficient inequality which is obtained by exploiting the solution given by the greedy heuristic. Our computational results show that our methods were able to solve, in very small time, to optimality all the instances considered in [4] and to optimality or near optimality all instances considered in [2]. Furthmore, we solved to optimality, for the first time, many benchmark instances.

An interesting contibution of this work is to propose simple and pratical methods to solve to optimality or near optimality a large-scale instances of VSP and therfore can be used by a non specialist. In fact, VSP arises in many non-mathematical applications such bioinformatics and social networks analysis. Furthemore, It could also be used as the basis to build sophisticated algorithms able to solve larger instances of VSP. Another interesting aspect of our approach, due to the combining heuristic and linear integer programming, is the possibility to know how the solution we obtain is close to an optimal solution. This is particularly important when the instances are large, and consequently it is difficult to obtain an optimal solution in reasonable time.

On the larger instances (number of vertices more than 800), our approach shows its limits despite the fact that it is competitive with the state-of-the-art methods. With the aim to solve those instances to optimality, more sophisticated techniques need to be developed. Thus, it would be intersesting to devise a branch-and-cut algorithm based on the valid inequalities introduced by Balas and Souza [1] and Didi Biha and Meurs [7].

In this paper, as often considered in the literature, the parameter $\beta(n)$ is set to $\lfloor \frac{2n}{3} \rfloor$ and the objective is to minimize the size of the separator. It would be intersting to extend the study developed in this paper to more general weigthed case where each vertex v has a weight $w(v)$ and the goal is to minimize the total weight of the separator. Also, it would be intersting to investigate the impact of the parameter $\beta(n)$ on the efficiency of our approach.

Acknowledgments

We would like to thank the two anonymous referees for their helpful comments and suggestions which helped to improve this paper.

References

[1] E. Balas and C. de Souza, *The vertex separator problem : a polyhedral investigation*, Mathematical Programming, n. 3, 103(2005), pp. 583–608.

[2] U. Benlic and J.K. Hao, *Breakout local search for the vertex separator problem*, Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, August(2013),Beijing,China.

[3] T.N. Bui and C. Jones, *Finding Good Approximate Vertex and Edge Partitions is NP-Hard*, Information Processing Letters,42(1992), pp. 153–159.

[4] V.F. Cavalcante and C.C. de Souza, *Exact algorithms for the vertex separator problem in graphs*, Networks, n. 3,57(2011), pp. 212–230.

[5] V.F. Cavalcante and C.C. de Souza, *Lagrangian relaxation and cutting planes for the vertex separator problem*, , in Combinatorics, Algorithms, Probabilistic and Experimental Methodologies,Lecture Notes in Computer Science, B. Chen, M. Paterson, and G. Zhang, eds., Springer-Verlag,(2007),pp. 471-482.

[6] B.V. Cherkassky and A.V. Goldberg, *On Implementing Push-Relabel Method for the Maximum Flow Problem*, Algorithmica, 19(1997), pp. 390-410.

[7] M. Didi Biha and M.J. Meurs, *An exact algorithm for solving the vertex separator problem*, Journal of Global Optimization,n. 3,49(2011), pp. 425–434.

[8] E. A. Dinic, *Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation*, Soviet Math. Dokl., 11 (1970), pp. 1277–1280.

[9] J. Fukuyama, *NP-completeness of the planar separator problems*, Journal of Graph Algorithms and Applications, 4(2006), pp. 317–328.

[10] L. R. Ford, Jr. and D. R. Fulkerson, *Flows in Networks*, Princeton Univ. Press, Princeton, NJ, (1962).

[11] M.R. Garey and D.S. Johnson, *Computers and Intractabiliy*, W.H. Freeman and Compagny, (1979).

[12] A. V. Goldberg, *A New Max-Flow Algorithm*, Technical Report MIT/LCS/TM-291, Laboratory for Computer Science, M.I.T., (1985).

[13] A. V. Goldberg and R. E. Tarjan, *A New Approach to the Maximum Flow Problem*, J. Assoc. Comput. Mach., 35(1988), pp. 921–940.

[14] A. V. Goldberg and R. E. Tarjan, *Efficient Maximum Flow Algorithms*, Communications of the ACM, Vol. 57, 8(2014), pp. 82–89.

[15] W. Hager and J.T. Hungerford, *Continuous quadratic programming formulations of optimization problems on graphs*, European Journal of Operational Research, 240(2014), pp. 328–337.

[16] C. Helmberg and F. Rendl, *A spectral bundle method for semidefinite programming*, SIAM J. Numer. Anal., 36(1979), pp. 177–189.

[17] S. Jesú s , J.P. Juan and D. Abraham, *Combing inteensification and diversification strategies in VSN.An application to the Vertex Separator problem*, Computers and Operations Research,52(2014), pp. 209–219.

[18] R.J. Lipton and R.E. Tarjan, *A separator theorem for planar graphs*, SIAM J. Num. Anal., 36(1979), pp. 177–189.

[19] J. Sanchez-Oro , N. Mladenovic and A. Duarte, *General variable neighborhood search for computing graph separators*, Optimization Letters,(2014), pp. 1–21.

[20] C. de Souza and E. Balas, *The vertex separator problem : algorithms and computations*, Mathematical Programming, n. 3, 103(2005), pp. 609–631.

[21] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*, Spring-Verlag, Berlin Heidelberg, 2003.

[22] R. E. Tarjan, *A Simple Version of Karzanov's Blocking Flow Algorithm*, Operations Research Letters, 2(1984), pp. 265–268.

[23] Z. Zhang and Z. Shao, *An improved K-opt local search algorithm for the vertex separator problem*, Journal of Computational and Theoretical Nanoscience, n. 11, 12(2015), pp. 4942–4958.

[24] http://http://lemon.cs.elte.hu/trac/lemon

[25] http://www.ic.unicamp.br/~cid/Problem-instances/VSP.html

[26] http://www.ilog.com

[27] http://www.optsicom.es/maxcut/♯instances.com