A 0.821-ratio purely combinatorial algorithm for maximum *k*-vertex cover in bipartite graphs

Edouard Bonnet[1,2], Bruno Escoffier[4,5], Vangelis Th. Paschos[1,2(b)], and Georgios Stamoulis[1,2,3]

[1] PSL* Research University, Université Paris-Dauphine, LAMSADE
{edouard.bonnet,paschos}@lamsade.dauphine.fr,georgios.stamoulis@dauphine.fr
[2] CNRS UMR 7243
[3] Universitá della svizzera Italiana
[4] Sorbonne Universités, UPMC Universite Paris 06
[5] UMR 7606, LIP6
bruno.escoffier@lip6.fr

Abstract. Our goal in this paper is to propose a *combinatorial algorithm* that beats the only such algorithm known previously, the greedy one. We study the polynomial approximation of MAX k-VERTEX COVER in bipartite graphs by a purely combinatorial algorithm and present a computer assisted analysis of it, that finds the worst case approximation guarantee that is bounded below by 0.821.

1 Introduction

In the MAX k-VERTEX COVER problem, a graph $G = (V, E)$ with $|V| = n$ and $|E| = m$ is given together with an integer $k \leqslant n$. The goal is to find a subset $K \subseteq V$ with k elements such that the total number of edges covered by K is maximized. We say that an edge $e = \{u, v\}$ is covered by a subset of vertices K if $K \cap e \neq \emptyset$. MAX k-VERTEX COVER is **NP**-hard in general graphs (as a generalization of MIN VERTEX COVER) and it remains hard in bipartite graphs [1,2].

The approximation of MAX k-VERTEX COVER has been originally studied in [3], where an approximation $1 - {}^{1}/{e}$ was proved, achieved by the natural greedy algorithm. This ratio is tight even in bipartite graphs [4]. In [5], using a sophisticated linear programming method, the approximation ratio for MAX k-VERTEX COVER is improved up to ${}^{3}/{4}$. Finally, by an easy reduction from MIN VERTEX COVER, it can be shown that MAX k-VERTEX COVER can not admit a polynomial time approximation schema (PTAS), unless $\mathbf{P} = \mathbf{NP}$ [9].

Obviously, the result of [5] immediately applies to the case of bipartite graphs. Very recently, [2] improves this ratio in bipartite graphs up to ${}^{8}/{9}$, still using linear programming.

Finally, let us note that MAX k-VERTEX COVER is polynomial in regular bipartite graphs or in semi-regular ones, where the vertices of each color class

(b) Institut Universitaire de France

have the same degree. Indeed, in both cases it suffices to chose k vertices in the color class of maximum degree.

Our Contribution. Our principal question motivating this paper is *to what extent combinatorial methods for this problem compete with linear programming ones*. In other words, *what is the ratios level, a purely combinatorial algorithm can guarantee?* In this purpose, we first devise a very simple algorithm that guarantees approximation ratio $2/3$, improving so the ratio of the greedy algorithm in bipartite graphs. Our main contribution consists of an approximation algorithm which computes six distinct solutions and returns the best among them.

There is an obvious difficulty in analyzing the performance guarantee of such an algorithm. Indeed it seems that there is no obvious way to compare different solutions and argue globally over them. Another factor that contributes to this difficulty is that we provide analytic expressions for all the solutions produced, fact that involves a number of cases per each of them and a large number of variables (in all 48 variables are used for the several solution-expressions). Similar situation was faced, for example, in [10] where the authors gave a 0.921 approximation guarantee for MAX CUT of maximal degree 3 (and an improved 0.924 for 3-regular graphs) by a computer assisted analysis of the quantities generated by theoretically analyzing a particular semi-definite relaxation of the problem at hand. Similarly, by setting up a suitable non-linear program and solving it, we give a computer assisted analysis of a 0.821-approximation guarantee for MAX k-VERTEX COVER in bipartite graphs. We give all the details of the implementation in Section 6.

2 Preliminaries

The basic ideas of the algorithm we propose are the following:
1. fix an optimal solution O (i.e., a vertex-set on k vertices covering a maximum number of edges in E) and guess the cardinalities k_1 and k_2 of its subsets O_1 and O_2 lying in the color-classes V_1 and V_2, respectively;
2. compute the sets S_i of k_i vertices in V_i, $i = 1, 2$ that cover the most of edges; obviously S_i is a set of the k_i largest degree vertices in V_i (breaking ties arbitrarily);
3. guess the cardinalities k_i' of the intersections $S_i \cap O_i$, $i = 1, 2$;
4. compute the sets X_i of the $k_i - k_i'$ best vertices from V_i in graphs $B[(V \setminus S_1), V_2]$ and $B[V_1, (V_2 \setminus S_2)]$, respectively;
5. choose the best among six solutions built as described in Section 4.

Sets S_i, X_i and O_i separate each color-class in 6 regions, namely, $S_i \cap O_i$, $S_i \setminus O_i$, $X_i \cap O_i$, $X_i \setminus O_i$, $O_i \setminus (S_i \cup X_i)$ (denoted by $\bar{O}_i$, in what follows) and $V_i \setminus (S_i \cup X_i \cup O_i)$. So, there totally exist 36 groups of edges (cuts) among them, the group $(V_1 \setminus (S_1 \cup X_1 \cup O_1), V_2 \setminus (S_2 \cup X_2 \cup O_2))$ being irrelevant as it will be hopefully understood in the sequel. We will use the following notations to refer to the values of the 35 relevant cuts (illustrated in Figure 1.):

B**:** the number of edges in the cut $(S_1 \setminus O_1, S_2 \cap O_2)$;

C: the number of edges in the cut $(S_2 \setminus O_2, S_1 \cap O_1)$;

F_1, F_2, F_3: the number of edges in the cuts $(S_1 \setminus O_1, X_2 \setminus O_2)$, $(S_1 \setminus O_1, O_2 \setminus (X_2 \cup S_2))$ and $(S_1 \setminus O_1, O_2 \cap X_2)$, respectively;

H_1, H_2: the number of edges in the cuts $(S_1 \cap O_1, X_2 \setminus O_2)$ and $(S_1 \cap O_1, V_2 \setminus (S_2 \cup X_2 \cup O_2))$, respectively;

$\{I_i\}_{i \in [6]}$: the number of edges in the cuts $(X_1 \setminus O_1, X_2 \setminus O_2)$, $(X_1 \setminus O_1, V_2 \setminus (S_2 \cup X_2 \cup O_2))$, $(O_1 \setminus (S_1 \cup X_1), X_2 \setminus O_2)$, $(O_1 \setminus (S_1 \cup X_1), V_2 \setminus (S_2 \cup X_2 \cup O_2))$, $(X_1 \cap O_1, X_2 \setminus O_2)$ and $(X_1 \cap O_1, V_2 \setminus (S_2 \cup X_2 \cup O_2))$, respectively;

J_1, J_2, J_3: the number of edges in the cuts $(S_2 \setminus O_2, X_1 \setminus O_1)$, $(S_2 \setminus O_2, O_1 \setminus (S_1 \cup X_1))$ and $(S_2 \setminus O_2, O_1 \cap X_1)$, respectively;

$\{L_i\}_{i \in [9]}$: the number of edges in the cuts $(S_1 \cap O_1, S_2 \cap O_2)$, $(S_1 \cap O_1, X_2 \cap O_2)$, $(S_1 \cap O_1, O_2 \setminus (S_2 \cup X_2))$, $(X_1 \cap O_1, S_2 \cap O_2)$, $(X_1 \cap O_1, X_2 \cap O_2)$, $(X_1 \cap O_1, O_2 \setminus (S_2 \cup X_2))$, $(O_1 \setminus (S_1 \cup X_1), S_2 \cap O_2)$, $(O_1 \setminus (S_1 \cup X_1), X_2 \cap O_2)$, and $(O_1 \setminus (S_1 \cup X_1), O_2 \setminus (S_2 \cup X_2))$, respectively;

N_1, N_2: the number of edges in the cuts $(S_2 \cap O_2, X_1 \setminus O_1)$ and $(S_2 \cap O_2, V_1 \setminus (S_1 \cup X_1 \cup O_1))$, respectively;

$\{P_i\}_{i \in [5]}$: the number of edges in the cuts $(X_2 \setminus O_2, V_1 \setminus (S_1 \cup X_1 \cup O_1))$, $(O_2 \setminus (S_2 \cup X_2), X_1 \setminus O_1)$, $(O_2 \setminus (S_2 \cup X_2), V_1 \setminus (S_1 \cup X_1 \cup O_1))$, $(X_2 \cap O_2, X_1 \setminus O_1)$, and $(X_2 \cap O_2, V_1 \setminus (S_1 \cup X_1 \cup O_1))$, respectively;

U_1, U_2, U_3: the number of edges is the cuts, $(S_1 \setminus O_1, S_2 \setminus O_2)$, $(S_1 \setminus O_1, V_2 \setminus (S_2 \cup X_2 \cup O_2))$ and $(S_2 \setminus O_2, V_1 \setminus (S_1 \cup X_1 \cup O_1))$, respectively.

Based upon the notations above and denoting by $\delta(V')$, $V' \subseteq V$, the number of edges covered by V' and by $\mathrm{opt}(B)$ the value of an optimal solution (i.e., the number edges covered) for MAX k-VERTEX COVER in the input graph B , the following holds (see also Figure 1):

$$\delta(S_1) = B + C + F_1 + F_2 + F_3 + H_1 + H_2 + L_1 + L_2 + L_3 + U_1 + U_2 \tag{1}$$

$$\delta(S_2) = B + C + J_1 + J_2 + J_3 + L_1 + L_4 + L_7 + N_1 + N_2 + U_1 + U_3 \tag{2}$$

$$\delta(X_1) = I_1 + I_2 + I_5 + I_6 + J_1 + J_3 + \sum_{i=4}^{6} L_i + N_1 + P_2 + P_4 \tag{3}$$

$$\delta(X_2) = F_1 + F_3 + H_1 + I_1 + I_3 + I_5 + L_2 + L_5 + L_8 + P_1 + P_4 + P_5 \tag{4}$$

$$\delta(O_1) = C + H_1 + H_2 + I_3 + I_4 + I_5 + I_6 + J_2 + J_3 + \sum_{i=1}^{9} L_i \tag{5}$$

$$\delta(O_2) = B + F_2 + F_3 + \sum_{i=1}^{9} L_i + N_1 + N_2 + \sum_{i=2}^{5} P_i \tag{6}$$

$$\begin{aligned}\mathrm{opt}(B) = B + C + \sum_{i=2}^{3} F_i + \sum_{i=1}^{2} H_i + \sum_{i=3}^{6} I_i + \sum_{i=2}^{3} J_i + \sum_{i=1}^{9} L_i \\ + \sum_{i=1}^{2} N_i + \sum_{i=2}^{5} P_i\end{aligned} \tag{7}$$

Without loss of generality, we assume $k_1 \leqslant k_2$ and we set: $k_1 = \mu k_2$ ($\mu \leqslant 1$), $k_1' = |S_1 \cap O_1| = \nu k_1$ ($0 \leqslant \nu \leqslant 1$) and $k_2' = |S_2 \cap O_2| = \xi k_2$ ($0 \leqslant \xi \leqslant 1$). Let us

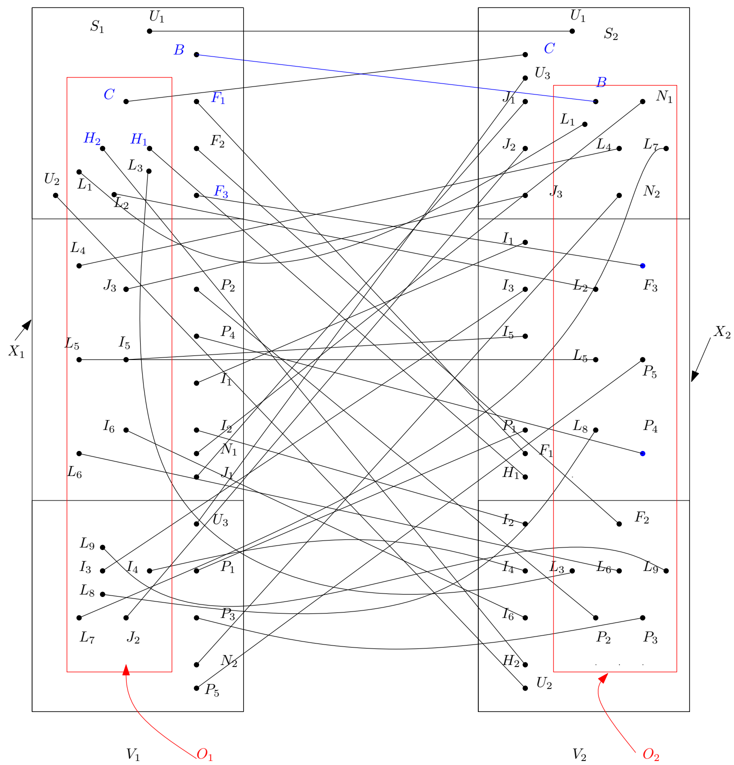

Fig. 1. Sets S_i, O_i, X_i $i = 1, 2$ and cuts between them.

note that, since k'_i vertices lie in the intersections $S_i \cap O_i$, the following hold for $\bar{O}_i = O_i \setminus (S_i \cup X_i)$, $i = 1, 2$: $|\bar{O}_1| = |O_1 \setminus (S_1 \cup X_1)| \leqslant (1-\nu)k_1 = \mu(1-\nu)k_2$ and $|\bar{O}_2| = |O_2 \setminus (S_2 \cup X_2)| \leqslant (1-\xi)k_2$. From the definitions of the cuts and using (1) to (6) and the expressions for $|\bar{O}_1|$ and $|\bar{O}_2|$, simple average arguments and the assumptions for k_1, k_2, k'_1 and k'_2 just above, the following holds:

$$\begin{array}{ll}
\delta(S_1) & \geq \delta(O_1) \\
\delta(S_2) & \geq \delta(O_2) \\
\delta(X_1) + C + H_1 + H_2 + L_1 + L_2 + L_3 & \geq \delta(O_1) \\
\delta(X_2) + B + N_1 + N_2 + L_1 + L_4 + L_7 & \geq \delta(O_2) \\
\delta(S_1) & \geq {}^{1}\!/_{1-\nu} \cdot \delta(X_1) \\
\delta(S_2) & \geq {}^{1}\!/_{1-\xi} \cdot \delta(X_2) \\
\delta(S_1) + \delta(X_1) & \geq {}^{2-\nu}\!/_{1-\nu} \cdot (I_3 + I_4 + J_2 + L_7 + L_8 + L_9) \\
\delta(S_2) + \delta(X_2) & \geq {}^{2-\xi}\!/_{1-\xi} \cdot (F_2 + L_3 + L_6 + L_9 + P_2 + P_3) \\
B + F_1 + F_2 + F_3 + U_1 + U_2 & \geq \delta(X_1) \\
C + J_1 + J_2 + J_3 + U_1 + U_3 & \geq \delta(X_2)
\end{array} \tag{8}$$

For $i = 1, 2$, the two first inequalities in (8) hold because S_i is the set of k_i highest-degree vertices in V_i; the third and fourth ones because the lefthand side quantities are the number of edges covered by $X_i \cup (S_i \cap O_i)$; each of these sets has cardinality k_i and obviously covers more edges than O_i; the fifth and sixth inequalities because the average degree of S_i is at least the average degree of X_i and $|X_1| = (1-\nu)k_1$ and $|X_2| = (1-\xi)k_2$; seventh and eighth ones because the average degree of vertices in $S_i \cup X_i$ is at least the average degree of vertices in $O_i \setminus (S_i \cup X_i)$; finally, for the last two inequalities the sum of degrees of the $k_i - k_i'$ vertices in $S_i \setminus O_i$ is at least the sum of degrees of the $k_i - k_i'$ vertices of X_i.

In Section 4, we specify the approximation algorithm sketched above. In Section 6 a computer assisted analysis of its approximation-performance is presented. The non-linear program that we set up, not only computes the approximation ratio of our algorithm but it also provides an experimental study over families of graphs. Indeed, a particular configuration on the variables (i.e., a feasible value assignments on the variables that represent the set of edges $B, C, \dots$) corresponds to a particular family of bipartite graphs with similar structural properties (characterized by the number of edges belonging to the several cut considered). Given such a configuration, it is immediate to find the ratio of the algorithm, because we can simply substitute the values of the variables in the corresponding ratios and output the largest one. We can view our program as an *experimental analysis* over all families of bipartite graphs, trying to find the particular family that implements the worst case for the approximation ratio of the algorithm. Our program not only finds such a configuration, but also provides data about the range of approximation factor on other families of bipartite graphs. Experimental results show that the approximation factor for the *absolute majority* of the instances is very close to 1 i.e., ≥ 0.95. Moreover, our program is *independent* on the size of the instance. We just need a particular configuration on the relative value of the variables $B, C, \dots$, thus providing a compact way of representing families of bipartite graphs sharing common structural properties.

For the rest of the paper, we call "best" vertices a set of vertices that cover the most of *uncovered* edges[6] in B. Given a solution $\mathrm{SOL}_k(B)$, we denote by $\mathrm{sol}_k(B)$ its value. For the quantities implied in the ratios corresponding to these solutions, one can be referred to Figure 1 and to expressions (1) to (7).

Let us note that the algorithm above, since it runs for any value of k_1 and k_2, it will run for $k_1 = k$ and $k_2 = k$. So, it is optimal for the instances of [4], where the greedy algorithm attains the ratio $^{(e-1)}/_{e}$.

Observe finally that, when $k \geqslant \min\{|V_1|, |V_2|\}$, then $\min\{|V_1|, |V_2|\}$ is an optimal solution since it covers the whole of E. This remark will be useful for some solutions in the sequel, for example in the completion of solution $\mathrm{SOL}_5(B)$.

[6] For instance, saying "we take S_1 plus the k_2 best vertices in V_2, this means that we take S_1 and then k_2 vertices of highest degree in $B[(V_1 \setminus S_1), V_2]$.

3 Some easy approximation results

3.1 A $^2/_3$-approximation algorithm

The algorithm goes as follows: fix an optimal solution $O \subseteq V_1 \cup V_2$, guess k_1 and k_2, build the following three solutions and output the best among them:

- SOL_1: take S_1 plus the k_2 remaining best vertices from V_2;
- SOL_2: take S_2 plus the k_1 remaining best vertices from V_1;
- SOL_3: take S_1 plus S_2.

SOL_1 will cover more than $\delta(S_1) + \delta(O_2) - \delta(S_1, \bar{O}_2)$, where $\bar{O}_2$ is $O_2 \setminus S_2$ and $\delta(S_1, \bar{O}_2)$ denotes the cardinality of the cut $(S_1, \bar{O}_2)$. The fact that this solution covers more than $\delta(O_1)$ from the V_1 side is obvious by the definition of S_1. The k_2 remaining best vertices from V_2 will cover at least as many edges as $O \cap V_2$, except those that are already covered. This is precisely $\delta(O_2) - \delta(S_1, O_2)$ (we take something better than the "surviving" part of O_2).

With a complete analogy as for SOL_1, we have that SOL_2 will cover at least $\delta(S_2) + \delta(O_1) - \delta(S_2, \bar{O}_1) \geq \delta(O_2) + \delta(O_1) - \delta(S_2, \bar{O}_1)$.

SOL_3 will cover at least $\delta(S_1, O_2) \geq \delta(S1, \bar{O}_2)$ from V_1. From S_2 it will cover at least $\delta(S_2, \bar{O}_1) + \delta((S_2 \cap O_2), \bar{O}_1) \geqslant \delta(S_2, \bar{O}_1)$.

It is easy to see that $\mathrm{sol}_1(B) + \mathrm{sol}_2(B) + \mathrm{sol}_3(B) \geqslant 2(\delta(O_1) + \delta(O_2)) \geqslant 2\mathrm{opt}$, qed.

Let us note that that the algorithm above guarantees ratio $^4/_5$, when both $k'_i = 0$, $i = 1, 2$ [11]. Note also that, since it runs for any value of k_1 and k_2, it will run for $k_1 = k$ and $k_2 = k$. So, it is optimal for the instances of [4], where the greedy algorithm attains the ratio $^{e-1}/_e$.

3.2 The case $\nu = \xi = 0$

We present in this section a simple algorithm (Algorithm **??**) handling the case where $O_1 \cap S_1 = \emptyset$ and $O_2 \cap S_2 = \emptyset$ (notice that this case is not polynomially detectable). We show that in this case, a $^4/_5$-approximation ratio can be achieved.

Consider the following algorithm:

1. for $i := 0$ to k do:
   (a) compute the set A_i (resp., B'_i) on i (resp., $k - i$) vertices of highest degrees in V_1 (resp., V_2);
   (b) remove A_i (resp. B'_i) from the graph, and compute the set A'_i (resp., B_i) on $k - i$ (resp. i) vertices of highest degrees in V_2 (resp., V_1) in the surviving graph;
   (c) store the two solutions $(A_i \cup A'_i)$ and $(B_i \cup B'_i)$;
2. returnn the best solution stored (denoted by $\mathrm{SOL}(B)$).

We now prove that if $\nu = \xi = 0$, then $\mathrm{sol}(B) \geqslant {^4/_5} \cdot \mathrm{opt}(B)$.

Fix an optimal solution $O = O_1 \cup O_2$ and consider the iteration of the algorithm with $i = k_1$. Set $A = A_i \cup A'_i$ and $B = B_i \cup B'_i$. Since the algorithm is symmetric, we can assume w.l.o.g. that $k_1 \leqslant k/2$. For some set $A \subseteq V$ denote by $e(A)$ the number of edges covered by A.

Once A_i has been taken, then the choice of A'_i is optimal among the possible sets of $k - i$ vertices in V_2. Hence:

$$\text{sol}(B) \geqslant e(A) \geq e\left(A_i \cup O_2\right) = \delta\left(A_i\right) + \delta\left(O_2\right) - \delta\left(A_i, O_2\right) \tag{9}$$

where $\delta(A_i, O_2)$ denotes the set of edges having one endpoint in A_i and the other one in O_2. Similarly,

$$\text{sol}(B) \geq e(A) \geq e\left(A_i \cup B'_i\right) \geq \delta\left(B'_i\right) + \delta\left(A_i, O_2\right) \tag{10}$$

Now, consider the solution when $i = k$, i.e., when Algorithm **??** takes the set A_k of k best vertices in V_1. Since $k_1 \leq k/2$ and O_1 and A_i are disjoint, it holds that:

$$\text{sol}(B) \geq e\left(A_k\right) \geq e\left(A_i \cup O_1\right) = \delta\left(A_i\right) + \delta\left(O_1\right) \tag{11}$$

Now, sum up (9), (10) and (11) with coefficients respectively 2, 2 and 1, respectively. Then:

$$5\text{sol}(B) \geq 4e(A) + e\left(A_k\right) \geq 3\delta\left(A_i\right) + \delta\left(O_1\right) + 2\delta\left(B'_i\right) + 2\delta\left(O_2\right)$$

Note that $\text{opt}(B) \leq \delta(O_1) + \delta(O_2)$. The results follows since by the choice of A_i and B'_i we have $\delta(A_i) \geq \delta(O_1)$ and $\delta(B'_i) \geq \delta(O_2)$.

4 A 0.821-approximation for the bipartite max k-vertex cover

Consider the following algorithm for MAX k-VERTEX COVER (called k-`VC_ALGORITHM` in what follows) which guesses k_1, k_2, k'_1 and k'_2, builds several feasible solutions and, finally, returns the best among them.

Fix an optimal solution O, guess the cardinalities k_1 and k_2 of O_1 and O_2 (swap these sets if necessary in order that $k_1 \leqslant k_2$), compute the sets S_i of k_i vertices in V_i, $i = 1, 2$, that cover the most of edges, guess the cardinalities k'_i of the intersections $S_i \cap O_i$, $i = 1, 2$, compute the sets X_i of $k_i - k'_i$ best vertices in $V_i \setminus S_i$, $i = 1, 2$ and build the following MAX k-VERTEX COVER-solutions:
$\mathbf{SOL_1(B)}$ and $\mathbf{SOL_2(B)}$, take, respectively, S_1 plus the k_2 remaining best vertices from V_2, and S_2 plus the k_1 remaining best vertices from V_1;
$\mathbf{SOL_3(B)}$ takes first $S_1 \cup X_1$ in the solution and completes it with the $(1 - \mu(1 - \nu))k_2$ best vertices from V_2;
$\mathbf{SOL_4(B)}$ takes S_2 and completes it either with vertices from V_2, or with vertices from both V_1 and V_2;
$\mathbf{SOL_5(B)}$ takes a π-fraction of the best vertices in S_1 and X_1, $\pi \in (0, 1/2]$; then, solution is completed with the $k_1 + k_2 - \pi(2k_1 - k'_1)$ best vertices in V_2;
$\mathbf{SOL_6(B)}$ takes a λ-fraction of the best vertices in S_2 and X_2, $\lambda \in (0, (1+\mu)/(2-\xi)]$; then solution is completed with the $k_1 + k_2 - \lambda(2k_2 - k'_2)$ best vertices in V_1.

Let us note that the values of λ and π are *parameters that we can fix.*

In what follows, we analyze solutions $\text{SOL}_1(B) \ldots \text{SOL}_6(B)$ computed by k-`VC_ALGORITHM` and give analytical expressions for their ratios.

4.1 Solution $\mathbf{SOL_1(B)}$

The best k_2 vertices in V_2, provided that S_1 has already been chosen, cover at least the maximum of the following quantities:

$$\begin{array}{ll}
\mathcal{A}_1 = J_1 + J_2 + J_3 + L_4 + L_7 + N_1 + N_2 + U_3 & \text{by } S_2 \\
\mathcal{A}_2 = I_1 + I_3 + I_5 + L_5 + L_8 + P_1 + P_4 + P_5 & \text{by } X_2 \\
\mathcal{A}_3 = L_4 + L_5 + L_6 + L_7 + L_8 + L_9 + N_1 + N_2 + P_2 + P_3 + P_4 + P_5 & \text{by } O_2
\end{array}$$

So, the approximation ratio for $\mathrm{SOL}_1(B)$ satisfies:

$$r_1 = \frac{\delta\left(S_1\right) + \max\left\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\right\}}{\mathrm{opt}(B)} \tag{12}$$

4.2 Solution $\mathbf{SOL_2(B)}$

Analogously, the best k_1 vertices in V_1, provided that S_2 has already been chosen, cover at least the maximum of the following quantities:

$$\begin{array}{ll}
\mathcal{B}_1 = H_1 + H_2 + F_1 + F_2 + F_3 + L_2 + L_3 + U_2 & \text{by } S_1 \\
\mathcal{B}_2 = I_1 + I_2 + I_5 + I_6 + L_5 + L_6 + P_2 + P_4 & \text{by } X_1 \\
\mathcal{B}_3 = H_1 + H_2 + I_3 + I_4 + I_5 + I_6 + L_2 + L_3 + L_5 + L_6 + L_8 + L_9 & \text{by } O_1
\end{array}$$

So, the approximation ratio for $\mathrm{SOL}_2(B)$ satisfies:

$$r_2 = \frac{\delta\left(S_2\right) + \max\left\{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3\right\}}{\mathrm{opt}(B)} \tag{13}$$

4.3 Solution $\mathbf{SOL_3(B)}$

Taking first $S_1 \cup X_1$ in the solution, $k - (k_1 + k_1 - k_1') = k_1 + k_2 - 2k_1 + k_1' = k_2 - (k_1 - k_1') = (1 - \mu(1-\nu))k_2$ vertices remain to be taken in V_2. The best such vertices will cover at least the maximum of the following quantities:

$$\mathcal{C}_1 = (1 - \mu(1-\nu))\left(J_2 + N_2 + L_7 + U_3\right) \tag{14}$$

$$\mathcal{C}_2 = \frac{1 - \mu(1-\nu)}{2 - \xi}\left(I_3 + J_2 + L_7 + L_8 + N_2 + P_1 + P_5 + U_3\right) \tag{15}$$

$$\mathcal{C}_3 = \frac{1 - \mu(1-\nu)}{3 - 2\xi}\left(I_3 + J_2 + L_7 + L_8 + L_9 + N_2 + P_1 + P_3 + P_5 + U_3\right) \tag{16}$$

where (14) corresponds to a completion by the $(1 - \mu(1-\nu))k_2$ best vertices of S_2, (15) corresponds to a completion by the $(1 - \mu(1-\nu))k_2$ best vertices of $S_2 \cup X_2$, while (16) corresponds to a completion by the $(1 - \mu(1-\nu))k_2$ best vertices of $S_2 \cup X_2 \cup \bar{O}_2$. The denominator $3 - 2\xi$ in (16) is due to the fact that, using the expression for $\bar{O}_2$, $|S_2 \cup X_2 \cup (O_2 \setminus (S_2 \cup X_2))| \leqslant (3 - 2\xi)k_2$. So, the approximation ratio for $\mathrm{SOL}_3(B)$ is:

$$r_3 = \frac{\delta\left(S_1\right) + \delta\left(X_1\right) + \max\left\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\right\}}{\mathrm{opt}(B)} \tag{17}$$

4.4 Solution $\mathbf{SOL_4(B)}$

Once S_2 taken in the solution, $k_1 = \mu k_2$ are still to be taken. Completion can be done in the following ways:

1. **if** $k_1 \leqslant k_2 - k_2'$, i.e., $\mu \leqslant 1 - \xi$, the best vertices taken for completion will cover at least either a $\mu/1-\xi$ fraction of edges incident to X_2, or a $\mu/2(1-\xi)$ fraction of edges incident to $X_2 \cup \bar{O}_2$, i.e., at least $\mathcal{M}_1$ edges, where $\mathcal{M}_1$ is given by:

$$\max\left\{\frac{\mu}{1-\xi}\delta\left(X_2\right), \frac{\mu}{2(1-\xi)}\left(\delta\left(X_2\right) + F_2 + L_3 + L_6 + L_9 + P_2 + P_3\right)\right\} \tag{18}$$

2. **else**, completion can be done by taking the whole set X_2 and then the additional vertices taken:
   (a) either within the rest of V_2 covering, in particular, a $\min\{1, \mu-1+\xi/|\bar{O}_2|\} \geqslant \min\{1, \mu-1+\xi/1-\xi\}$ fraction of edges incident to $\bar{O}_2$ (quantity $\mathcal{M}_2$ in (19)),
   (b) or in S_1 covering, in particular, a $\mu-1+\xi/\mu$ fraction of uncovered edges incident to S_1 (quantity $\mathcal{M}_3$ in (19)),
   (c) or in $S_1 \cup X_1$ covering, in particular, a $\mu-1+\xi/\mu(2-\nu)$ fraction of uncovered edges incident to $S_1 \cup X_1$ (quantity $\mathcal{M}_4$ in (19)),
   (d) or, finally, in $S_1 \cup X_1 \cup \bar{O}_1$ covering, in particular, a $\mu-1+\xi/\mu(3-2\nu)$ fraction of uncovered edges incident to this vertex-set (quantity $\mathcal{M}_5$ in (19));

   in any case such a completion will cover a number of edges that is at least the maximum of the following quantities:

$$\begin{aligned}
\mathcal{M}_2 &= \min\left\{1, \tfrac{\mu-1+\xi}{1-\xi}\right\}\left(F_2 + L_3 + L_6 + L_9 + P_2 + P_3\right) \\
\mathcal{M}_3 &= \tfrac{\mu-1+\xi}{\mu}\left(F_2 + H_2 + L_3 + U_2\right) \\
\mathcal{M}_4 &= \tfrac{\mu-1+\xi}{\mu(2-\nu)}\left(F_2 + H_2 + I_2 + I_6 + L_3 + L_6 + P_2 + U_2\right) \\
\mathcal{M}_5 &= \tfrac{\mu-1+\xi}{\mu(3-2\nu)}\left(F_2 + H_2 + I_2 + I_4 + I_6 + L_3 + L_6 + L_9 + P_2 + U_2\right)
\end{aligned} \tag{19}$$

Using (18) and (19), the following holds for the approximation ratio of $\mathrm{SOL}_4(B)$:

$$r_4 = \frac{\delta\left(S_2\right) + \begin{cases} \mathcal{M}_1 & \mu \leq 1-\xi \\ \delta\left(X_2\right) + \max\left\{\mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4, \mathcal{M}_5\right\} & \mu \geq 1-\xi \end{cases}}{\mathrm{opt}(B)} \tag{20}$$

4.5 Vertical separations – solutions $\mathbf{SOL_5(B)}$ and $\mathbf{SOL_6(B)}$

For $i = 1, 2$, given a vertex subset $V' \subseteq V_i$, we call *vertical separation of V' with parameter $c \in (0, 1/2]$*, a partition of V' into two subsets such that one of them contains a c-fraction of the best (highest degree) vertices of V'. Then, the following easy claim holds for a vertical separation of $V' \cup V''$ with parameter c.

Claim. Let $A(V')$ be a fraction c of the best vertices in V' and $A(V'')$ the same in V''. Then $\delta(A(V')) + \delta(A(V'')) \geq c\delta(V' \cup V'')$.

Proof. Assume that in V' we have n' vertices. To form $A(V')$ we take the cn' vertices of V' with highest degree. The average degree of V' is ${}^{\delta(V')}/_{n'}$. The average degree of $A(V')$ is ${}^{\delta(A(V'))}/_{cn'}$. But, from the selection of $A(V')$ as the cn' vertices with highest degree, we have that ${}^{\delta(A(V'))}/_{cn'} \geq {}^{\delta(V')}/_{n'} \Rightarrow \delta(A(V')) \geq c\delta(V')$. Similarly for V'', i.e., $\delta(A(V'')) \geq c\delta(V'')$.

Solutions $\mathrm{SOL}_5(B)$ and $\mathrm{SOL}_6(B)$ are based upon vertical separations of $S_i \cup X_i$, $i = 1, 2$, with parameters π and λ, called π- and λ-vertical separations, respectively.

The idea behind vertical separation, is to handle the scenario when there is a "tiny" part of the solution (i.e. few in comparison to, let's say, k_1 vertices) that covers a large part of the solution and the "completion" of the solution done by the previous cases does not contribute more than a small fraction to the final solution. The vertical separation indeed tries to identify such a small part, and then continues the completion on the other side of the bipartition.

Solution $\mathbf{SOL_5(B)}$. It consists of *separating $S_1 \cup X_1$ with parameter $\pi \in (0, 1/2]$, of taking a π fraction of the best vertices of S_1 and of X_1 in the solution and of completing it with the adequate vertices from V_2*. A π-vertical separation of $S_1 \cup X_1$ introduces in the solution $\pi\,(2k_1 - k_1') = \pi(2-\nu)\mu k_2$ vertices of V_1, which are to be completed with:

$$k - \pi(2-\nu)\mu k_2 = (1+\mu)k_2 - \pi(2-\nu)\mu k_2 = (1 - \mu(2\pi - 1) + \mu\nu\pi)k_2$$

vertices from V_2. Observe that such a separation implies the cuts with corresponding cardinalities B, C, F_i, $i = 1, 2, 3$, H_1, H_2, I_1, I_2, I_5, I_6, J_1, J_3, L_j, $j = 1, \dots, 6$, N_1, P_2, P_4, U_1 and U_2. Let us group these cuts in the following way:

$$\begin{aligned} \Pi_1 &= C + J_1 + J_3 + U_1 \\ \Pi_2 &= B + L_1 + L_4 + N_1 \\ \Pi_3 &= F_3 + L_2 + L_5 + P_4 \\ \Pi_4 &= I_1 + I_5 + F_1 + H_1 \\ \Pi_5 &= F_2 + L_3 + L_6 + P_2 \\ \Pi_6 &= I_2 + I_6 + H_2 + U_2 \end{aligned} \tag{21}$$

We may also notice that group Π_1 refers to $S_2 \setminus O_2$, Π_2 refers to $S_2 \cap O_2$, Π_3 to $X_2 \cap O_2$, Π_5 to $\bar{O}_2$ and Π_4 to $X_2 \setminus O_2$. Assume that a $\pi_i < 1$ fraction of each group Π_i, $i = 1, \dots 6$ contributes in the π vertical separation of $S_1 \cup X_1$. Then, a π-vertical separation of $S_1 \cup X_1$ will contribute with a value:

$$\sum_{i=1}^{6} \pi_i \Pi_i \geqslant \pi \sum_{i=1}^{6} \Pi_i \tag{22}$$

to $\mathrm{sol}_5(B)$. We now distinguish two cases.
Case 1: $(1 - \mu(2\pi - 1) + \mu\nu\pi)k_2 \geqslant k_2$, i.e., $1 - \mu(2\pi - 1) + \mu\nu\pi \geqslant 1$. Then we have:
1. $\mu(1 - 2\pi) + \mu\nu\pi \leq 1 - \xi$; then, the partial solution induced by the π-vertical

separation will be completed in such a way that the contribution of the completion is at least equal to $\max\{Z_i, i = 1, \ldots, 5\}$, where:
Z_1 refers to S_2 plus the best $(1-\mu(2\pi-1)+\mu\nu\pi)k_2 - k_2 = (\mu(1-2\pi)+\mu\nu\pi)k_2$ vertices of O_2 having a contribution of:

$$\begin{aligned} Z_1 = \sum_{i=1}^{2} (1-\pi_i)\, \Pi_i + (J_2 + L_7 + N_2 + U_3) + \frac{\mu(1-2\pi)+\mu\nu\pi}{1-\xi} [(1-\pi_3)\, \Pi_3 \\ + (1-\pi_5)\, \Pi_5 + (L_8 + L_9 + P_3 + P_5)] \end{aligned} \tag{23}$$

Z_2 refers to S_2 plus the best $(\mu(1-2\pi)+\mu\nu\pi)k_2$ vertices of X_2 having a contribution of:

$$\begin{aligned} Z_2 = &\sum_{i=1}^{2} (1-\pi_i)\, \Pi_i + (J_2 + L_7 + N_2 + U_3) \\ &+ \frac{\mu(1-2\pi)+\mu\nu\pi}{1-\xi} \left[\sum_{j=3}^{4} (1-\pi_i)\, \Pi_i + (I_3 + L_8 + P_1 + P_5) \right] \end{aligned} \tag{24}$$

Z_3 and Z_4 refer to the best $(1-\mu(2\pi-1)+\mu\nu\pi)k_2$ vertices of $S_2 \cup X_2$ and of $S_2 \cup O_2$ having, respectively, contributions:

$$\begin{aligned} Z_3 = &\frac{1-\mu(2\pi-1)+\mu\nu\pi}{2-\xi} \left[\sum_{i=1}^{4} (1-\pi_i)\, \Pi_i \right. \\ &\left. + (I_3 + J_2 + L_7 + L_8 + N_2 + P_1 + P_5 + U_3) \right] \end{aligned} \tag{25}$$

$$\begin{aligned} Z_4 = &\frac{1-\mu(2\pi-1)+\mu\nu\pi}{2-\xi} \left[\sum_{i=1}^{3} (1-\pi_i)\, \Pi_i + (1-\pi_5)\, \Pi_5 \right. \\ &\left. + (J_2 + L_7 + L_8 + L_9 + N_2 + P_3 + P_5 + U_3) \right] \end{aligned} \tag{26}$$

Z_5 refers to the best $(1-\mu(2\pi-1)+\mu\nu\pi)k_2$ vertices of $S_2 \cup X_2 \cup \bar{O}_2$ having a contribution of:

$$\begin{aligned} Z_5 = &\frac{1-\mu(2\pi-1)+\mu\nu\pi}{3-2\xi} \left[\sum_{i=1}^{5} (1-\pi_i)\, \Pi_i \right. \\ &\left. + (I_3 + J_2 + L_7 + L_8 + L_9 + N_2 + P_1 + P_3 + P_5 + U_3) \right] \end{aligned} \tag{27}$$

2. $\mu(1-2\pi)+\mu\nu\pi \geq 1-\xi$; in this case, the partial solution induced by the π-vertical separation will be completed in such a way that the contribution of the completion is at least $\max\{\Theta_i, i = 1, \ldots, 3\}$, where:
Θ_1 refers to $S_2 \cup X_2$ plus the best $(\mu(1-2\pi)+\mu\nu\pi-(1-\xi))k_2$ vertices of $\bar{O}_2$, all this having a contribution of:

$$\begin{aligned} \Theta_1 = &\sum_{i=1}^{4} (1-\pi_i)\, \Pi_i + (I_3 + J_2 + L_7 + L_8 + N_2 + P_1 + P_5 + U_3) \\ &+ \frac{\mu(1-2\pi)+\mu\nu\pi-(1-\xi)}{1-\xi} [(1-\pi_5)\, \Pi_5 + L_9 + P_3] \end{aligned} \tag{28}$$

Θ_2 refers to $S_2 \cup O_2$ plus the best $(\mu(1-2\pi)+\mu\nu\pi-(1-\xi))k_2$ vertices of $X_2 \setminus O_2$, all this having a contribution of:

$$\begin{aligned}\Theta_2 = &\sum_{i=1}^{3}(1-\pi_i)\,\Pi_i \\ &+(1-\pi_5)\,\Pi_5+(J_2+L_7+L_8+L_9+N_2+P_3+P_5+U_3) \\ &+\frac{\mu(1-2\pi)+\mu\nu\pi-(1-\xi)}{1-\xi}\left[(1-\pi_4)\,\Pi_4+I_3+P_1\right] \end{aligned} \tag{29}$$

Θ_3 refers to the best $(1-\mu(2\pi-1)+\mu\nu\pi)k_2$ vertices of $S_2 \cup X_2 \cup \bar{O}_2$ having a contribution of:

$$\begin{aligned}\Theta_3 = &\frac{1-\mu(2\pi-1)+\mu\nu\pi}{3-2\xi}\left[\sum_{i=1}^{5}(1-\pi_i)\,\Pi_i\right. \\ &\left.+(I_3+J_2+L_7+L_8+L_9+N_2+P_1+P_3+P_5+U_3)\right] \end{aligned} \tag{30}$$

Case 2: $1-\mu(2\pi-1)+\mu\nu\pi < 1$. The partial solution induced by the π-vertical separation will be completed in such a way that the contribution of the completion is at least equal to $\max\{\Phi_i, i=1,\ldots,5\}$, where:
Φ_1 refers to the best $(1-\mu(2\pi-1)+\mu\nu\pi)k_2$ vertices in S_2 with a contribution:

$$\Phi_1 = (1-\mu(2\pi-1)+\mu\nu\pi)\left[\sum_{i=1}^{2}(1-\pi_i)\,\Pi_i+(J_2+L_7+N_2+U_3)\right] \tag{31}$$

Φ_2 refers to the best $(1-\mu(2\pi-1)+\mu\nu\pi)k_2$ vertices in X_2 with a contribution:

$$\Phi_2 = \frac{1-\mu(2\pi-1)+\mu\nu\pi}{1-\xi}\left[\sum_{i=3}^{4}(1-\pi_i)\,\Pi_i+(I_3+L_8+P_1+P_5)\right] \tag{32}$$

Φ_3 refers to the best $(1-\mu(2\pi-1)+\mu\nu\pi)k_2$ vertices in O_2 with a contribution:

$$\begin{aligned}\Phi_3 = &(1-\mu(2\pi-1)+\mu\nu\pi)\left[\sum_{i=2}^{3}(1-\pi_i)\,\Pi_i+(1-\pi_5)\,\Pi_5\right. \\ &\left.+(L_7+L_8+L_9+N_2+P_3+P_5)\right] \end{aligned} \tag{33}$$

Φ_4 refers to the best $(1-\mu(2\pi-1)+\mu\nu\pi)k_2$ vertices in $S_2 \cup X_2$ with a contribution:

$$\begin{aligned}\Phi_4 = &\frac{1-\mu(2\pi-1)+\mu\nu\pi}{2-\xi}\left[\sum_{j=1}^{4}(1-\pi_j)\,\Pi_j\right. \\ &\left.+(I_3+J_2+L_7+L_8+N_2+P_1+P_5+U_3)\right] \end{aligned} \tag{34}$$

Φ_5 refers to the best $(1-\mu(2\pi-1)+\mu\nu\pi)k_2$ vertices in $S_2 \cup X_2 \cup \bar{O}_2$ with a contribution:

$$\begin{aligned}\Phi_5 = &\frac{1-\mu(2\pi-1)+\mu\nu\pi}{3-2\xi}\left[\sum_{j=1}^{5}(1-\pi_j)\,\Pi_j\right. \\ &\left.+(I_3+J_2+L_7+L_8+L_9+N_2+P_1+P_3+P_5+U_3)\right] \end{aligned} \tag{35}$$

Setting $Z^* = \max\{Z_i : i = 1, \dots 5\}$, $\Theta^* = \max\{\Theta_i : i = 1, 2, 3\}$ and $\Phi^* = \max\{\Phi_i : i = 1, \dots 5\}$, and putting (21) and (22) together with expressions (23) to (35), we get for ratio r_5:

$$\frac{\sum_{i=1}^{6} \pi_i \Pi_i + \begin{cases} \left\{\begin{array}{l} Z^* \text{ if } \mu(1-2\pi)+\mu\nu\pi \leq 1-\xi \\ \Theta^* \text{ if } \mu(1-2\pi)+\mu\nu\pi \geq 1-\xi \end{array}\right\} & \text{case: } 1-\mu(2\pi-1)+\mu\nu\pi \geq 1 \\ \Phi^* & \text{case: } 1-\mu(2\pi-1)+\mu\nu\pi < 1 \end{cases}}{\operatorname{opt}(B)} \tag{36}$$

Solution $\mathbf{SOL_6(B)}$. Symmetrically to $\mathrm{SOL}_5(B)$, solution $\mathrm{SOL}_6(B)$ consists of *separating $S_2 \cup X_2$ with parameter λ, of taking a λ fraction of the best vertices of S_2 and X_2 in the solution and of completing it with the adequate vertices from V_1*. Here, we need that:

$$\lambda\left(k_2 + k_2 - k_2'\right) \leqslant k \Rightarrow \lambda(2-\xi)k_2 \leqslant (1+\mu)k_2 \Rightarrow \lambda \leqslant \frac{1+\mu}{2-\xi} \Rightarrow \lambda \in \left(0, \frac{1+\mu}{2-\xi}\right]$$

A λ-vertical separation of $S_2 \cup X_2$ introduces in the solution $\lambda(2-\xi)k_2$ vertices of V_2, which are to be completed with:

$$k - \lambda(2-\xi)k_2 = (1+\mu)k_2 - \lambda(2-\xi)k_2 = (1+\mu-\lambda(2-\xi))k_2$$

vertices from V_1.

Observe that such a separation implies the cuts with corresponding cardinalities B, C, F_1, F_3, H_1, I_1, I_3, I_5, J_i, $i = 1, 2, 3$, L_1, L_2, L_4, L_5, L_7, L_8, N_1, N_2, P_1, P_4, P_5, U_1 and U_3. We group these cuts in the following way:

$$\begin{array}{l} \Lambda_1 = B + F_1 + F_3 + U_1 \\ \Lambda_2 = C + H_1 + L_1 + L_2 \\ \Lambda_3 = J_3 + I_5 + L_4 + L_5 \\ \Lambda_4 = I_1 + J_1 + N_1 + P_4 \\ \Lambda_5 = I_3 + J_2 + L_7 + L_8 \\ \Lambda_6 = N_2 + P_1 + P_5 + U_3 \end{array} \tag{37}$$

Group Λ_1 refers to $S_1 \setminus O_1$, Λ_2 to $S_1 \cap O_1$, Λ_3 to $X_1 \cap O_1$, Λ_5 to $\bar{O}_1$ and Λ_4 to $X_1 \setminus O_1$. Assume, as previously, that a $\lambda_i < 1$ fraction of each group Λ_i, $i = 1, \dots 6$ contributes in the λ vertical separation of $S_2 \cup X_2$. Then, a λ-vertical separation of $S_2 \cup X_2$ will contribute with a value:

$$\sum_{i=1}^{6} \lambda_i \Lambda_i \geqslant \lambda \sum_{i=1}^{6} \Lambda_i \tag{38}$$

to $\mathrm{sol}_6(B)$. We again distinguish two cases.

1. $(1+\mu-\lambda(2-\xi))k_2 \geqslant \mu k_2$, i.e., $1+\mu-\lambda(2-\xi) \geqslant \mu$. Here we have the two following subcases:

(a) $1-\lambda(2-\xi) \leq (1-\nu)\mu$; then, the partial solution induced by the λ-vertical separation will be completed in such a way that the contribution of the completion is at least equal to $\varUpsilon^* = \max\{\varUpsilon_i, i=1,\ldots,5\}$, where:
$\varUpsilon_1$ refers to S_1 plus the best $(1-\lambda(2-\xi))k_2$ vertices of X_1 having a contribution of:

$$\varUpsilon_1 = \sum_{i=1}^{2} (1-\lambda_i)\,\varLambda_i + (H_2 + F_2 + L_3 + U_2) + \frac{1-\lambda(2-\xi)}{\mu(1-\nu)} \left[\sum_{i=3}^{4} (1-\lambda_i)\,\varLambda_i + (I_2 + I_6 + L_6 + P_2)\right] \quad (39)$$

$\varUpsilon_2$ refers to S_1 plus the best $(1-\lambda(2-\xi))k_2$ vertices of O_1 having a contribution of:

$$\varUpsilon_2 = \sum_{i=1}^{2} (1-\lambda_i)\,\varLambda_i + (H_2 + F_2 + L_3 + U_2) + \frac{1-\lambda(2-\xi)}{\mu(1-\nu)} [(1-\lambda_3)\,\varLambda_3 + (1-\lambda_5)\,\varLambda_5 + (I_4 + I_6 + L_6 + L_9)] \quad (40)$$

$\varUpsilon_3$ and $\varUpsilon_4$ refer to the best $(1+\mu-\lambda(2-\xi))k_2$ vertices of $S_1 \cup X_1$ and $S_1 \cup O_1$ having, respectively, contributions:

$$\varUpsilon_3 = \frac{\mu+1-\lambda(2-\xi)}{\mu(2-\nu)} \left[\sum_{i=1}^{4} (1-\lambda_i)\,\varLambda_i + (F_2 + H_2 + I_2 + I_6 + L_3 + L_6 + P_2 + U_2)\right] \quad (41)$$

$$\varUpsilon_4 = \frac{\mu+1-\lambda(2-\xi)}{\mu(2-\nu)} \left[\sum_{i=1}^{3} (1-\lambda_i)\,\varLambda_i + (1-\lambda_5)\,\varLambda_5 + (F_2 + H_2 + I_4 + I_6 + L_3 + L_6 + L_9 + U_2)\right] \quad (42)$$

$\varUpsilon_5$ refers to the best $(1+\mu-\lambda(2-\xi))k_2$ vertices of $S_1 \cup X_1 \cup \bar{O}_1$ having a contribution of:

$$\varUpsilon_5 = \frac{\mu+1-\lambda(2-\xi)}{\mu(3-2\nu)} \left[\sum_{j=1}^{5} (1-\lambda_j)\,\varLambda_j + (F_2 + H_2 + I_2 + I_4 + I_6 + L_3 + L_6 + L_9 + P_2 + U_2)\right] \quad (43)$$

(b) $1-\lambda(2-\xi) \geq (1-\nu)\mu$; in this case, the partial solution induced by the λ-vertical separation will be completed in such a way that the contribution of the completion is at least $\varPsi^* = \max\{\varPsi_i, i=1,\ldots,3\}$, where:
$\varPsi_1$ refers to $S_1 \cup X_1$ plus the best $(1-\lambda(2-\xi)-(1-\nu))k_2$ vertices of $\bar{O}_1$, all this having a contribution of:

$$\varPsi_1 = \sum_{j=1}^{4} (1-\lambda_j)\,\varLambda_j + (F_2 + H_2 + I_2 + I_6 + L_3 + L_6 + P_2 + U_2) + \frac{1-\lambda(2-\xi)-\mu(1-\nu)}{\mu(1-\nu)} [(1-\lambda_5)\,\varLambda_5 + I_4 + L_9] \quad (44)$$

Ψ_2 refers to $S_1 \cup O_1$ plus the best $(1 - \lambda(2 - \xi) - (1 - \nu))k_2$ vertices of $X_1 \setminus O_1$, all this having a contribution of:

$$\Psi_2 = \sum_{j=1}^{3} (1 - \lambda_j)\,\Lambda_j + (1 - \lambda_5)\,\Lambda_5$$
$$+ (F_2 + H_2 + I_4 + I_6 + L_3 + L_6 + L_9 + U_2)$$
$$+ \frac{1 - \lambda(2 - \xi) - \mu(1 - \nu)}{\mu(1 - \nu)} \left[(1 - \lambda_4)\,\Lambda_4 + (I_2 + P_2)\right] \quad (45)$$

Ψ_3 refers to the best $(\mu + 1 - \lambda(2 - \xi))k_2$ vertices of $S_1 \cup X_1 \cup \bar{O}_1$ having a contribution of:

$$\Psi_3 = \frac{\mu + 1 - \lambda(2 - \xi)}{\mu(3 - 2\nu)} \left[\sum_{j=1}^{5} (1 - \lambda_j)\,\Lambda_j \right.$$
$$\left. + (F_2 + H_2 + I_2 + I_4 + I_6 + L_3 + L_6 + L_9 + P_2 + U_2)\right] \quad (46)$$

2. $1 + \mu - \lambda(2 - \xi) \leqslant \mu$. The partial solution induced by the λ-vertical separation will be completed in such a way that the contribution of the completion is at least equal to $\Omega^* = \max\{\Omega_i, i = 1, \ldots, 5\}$, where:
Ω_1 refers to the best $(1 + \mu - \lambda(2 - \xi))k_2$ vertices in S_1 with a contribution:

$$\Omega_1 = \frac{1 + \mu - \lambda(2 - \xi)}{\mu} \left[\sum_{j=1}^{2} (1 - \lambda_j)\,\Lambda_j + (F_2 + H_2 + L_3 + U_2)\right] \quad (47)$$

Ω_2 refers to the best $(1 + \mu - \lambda(2 - \xi))k_2$ vertices in X_1 with a contribution:

$$\Omega_2 = \frac{1 + \mu - \lambda(2 - \xi)}{\mu} \left[\sum_{j=3}^{4} (1 - \lambda_j)\,\Lambda_j + (I_2 + I_6 + L_6 + P_2)\right] \quad (48)$$

Ω_3 refers to the best $(1 + \mu - \lambda(2 - \xi))k_2$ vertices in O_1 with a contribution:

$$\Omega_3 = \frac{1 + \mu - \lambda(2 - \xi)}{\mu} \left[\sum_{j=2}^{3} (1 - \lambda_j)\,\Lambda_j + (1 - \lambda_5)\,\Lambda_5 \right.$$
$$\left. + (H_2 + I_4 + I_6 + L_3 + L_6 + L_9)\right] \quad (49)$$

Ω_4 refers to the best $(1 + \mu - \lambda(2 - \xi))k_2$ vertices in $S_1 \cup X_1$ with a contribution:

$$\Omega_4 = \frac{1 + \mu - \lambda(2 - \xi)}{\mu(2 - \nu)} \left[\sum_{j=1}^{4} (1 - \lambda_j)\,\Lambda_j \right.$$
$$\left. + (F_2 + H_2 + I_2 + I_6 + L_3 + L_6 + P_2 + U_2)\right] \quad (50)$$

Ω_5 refers to the best $(1+\mu-\lambda(2-\xi))k_2$ vertices in $S_1 \cup X_1 \cup \bar{O}_1$ with a contribution:

$$\Omega_5 = \frac{1+\mu-\lambda(2-\xi)}{\mu(3-2\nu)} \left[\sum_{j=1}^{5} (1-\lambda_j)\, \Lambda_j \right.$$
$$\left. + (F_2 + H_2 + I_2 + I_4 + I_6 + L_3 + L_6 + L_9 + P_2 + U_2) \right] \qquad (51)$$

Putting (37) and (38) together with expressions (39) to (51), we get:

$$r_6 = \frac{\sum_{i=1}^{6} \lambda_i \Lambda_i + \begin{cases} \begin{Bmatrix} \Upsilon^* \text{ if } 1-\lambda(2-\xi) \le (1-\nu)\mu \\ \Psi^* \text{ if } 1-\lambda(2-\xi) > (1-\nu)\mu \end{Bmatrix} & \text{case: } \mu+1-\lambda(2-\xi) \ge \mu \\ \Omega^* & \text{case: } \mu+1-\lambda(2-\xi) < \mu \end{cases}}{\mathrm{opt}(B)} \qquad (52)$$

5 Results

To analyze the performance guarantee of k-VC_ALGORITHM, we set up a non-linear program and solved it to optimality. Here, we interpret the set of edges $B, C, F_i, \ldots,$ as *variables* , the expressions in (8) as *constraints* and the *objective function* is $\min r (\equiv \max_{j=1}^{6} r_j)$. In other words, we try to find a value assignments to the set of variables such that the maximum among all the six ratios defined is minimized. This value would give us the desired approximation guarantee of k-VC_ALGORITHM.

Towards this goal, we set up a GRG (Generalized Reduced Gradient [12]) program. The reasons this method is selected are presented in Section 6, as well as a more detailed description of the implementation. GRG is a generalization of the classical *Reduced Gradient* method [13] for solving (concave) quadratic problems so that it can handle higher degree polynomials and incorporate non-linear constraints. Table 2 in the following Section 6 shows the results of the GRG program about the values of variables and quantities. The values of ratios $r_1 \div r_6$ computed for them are the following:

$$\begin{aligned} r_1 &= 0.81806 \\ r_2 &= 0.81797 \\ r_3 &= 0.79280 \\ r_4 &= 0.79657 \\ \mathbf{r_5} &= \mathbf{0.82104} \\ r_6 &= 0.82103 \end{aligned}$$

These results correspond to the cycle that outputs the *minimum* value for the approximation factor and this is 0.821, given by solution SOL_5.

Remark. As we note in Section 6, the GRG solver does not guarantee the global optimal solution. The 0.821 guarantee is the minimum value that the solver *returns after several runs from different initial starting points. However, successive re-executions of the algorithm, starting from this minimum value, were unable to find another point with smaller value. In each one of these successive re-runs, we tested the algorithm on 1000 random different starting points (which is greater than the estimation of the number of local minima) and the solver did not find value worse that the reported one.*

6 A computer assisted analysis of the approximation ratio of k-VC_ALGORITHM

6.1 Description of the method

In this section we give details of the implementation of the solutions of the previous sections (as captured by the corresponding ratios) and we explain how these ratios guarantee a performance ratio of 0.821, i.e., that there is always a ratio among the ones described that is within a factor of 0.821 of the optimal solution value for the bipartite MAX k-VERTEX COVER.

Our strategy can be summarized as follows. We see the cardinalities of all cuts defined in Section 2 as *variables*. These quantities represent how many edges go from one specific part of the bi-partition to any other given part of the other side of the bipartition. Counting these edges gives the value of the desired solution. By a proper scaling (i.e., by dividing every variable by the maximum among them) we guarantee that all these variables are in $[0, 1]$. Our goal is to find a particular configuration (which means a value assignment on the variables) such that the *maximum* among all the different ratios that define the solutions of the previous section is as low as possible. This will give the performance guarantee.

This boils down to an optimization problem which can be, more formally, described as follows:

$$\min r^* \quad \text{such that} \quad \max_i \{r_i\} \leqslant r^* \tag{53}$$

Unfortunately, given the nature of the constraints captured by (53), this is *not* a linear problem even though each variable appears as a monomial on the numerator and denominator of each constraint. This is because the numerators of r_3 (17), r_4 (20), r_5 (36) and r_6 (52) are polynomials of degree 3 or 4. Otherwise we could easily set up and solve to optimality this optimization problem, with our favorite linear solver.

To the best of our knowledge, there are no commercial solvers for solving polynomial optimization problems to find the *global* optimal solution. All solvers for such polynomial systems stuck on local optima. The task then is to run the solver many times, with different starting points and different parameters, and to apply knowledge and intuition about the “ballpark” of the optimal solution value together with the respective configuration of the values of the variables, to be sure (given an error ϵ unavoidable in such situations) that the optimal (or an almost optimal) solution of (53) is reached.

We note here that a promising although, as we will shortly argue, unsuccessful approach would be to set up a *Mathematica*® program and would solve it exploiting the command `solve` which solve to optimality a system of polynomial equations using Gröbner basis approach. Unfortunately, this is a solver that *solves* a system of polynomial equations, and not an optimizer. In other words, given such a system as an input on the `solve` environment, this will either report that no feasible solution in the domain exists, or report a solution (value on the variables) that satisfy the system. Another, more serious, limitation is the following: we do not seek a configuration of the variable that satisfies all constraints (ratios). But we seek a configuration of minimum value such that there exists at least one constraint with value greater than the value of the configuration. In other words, if we look more carefully on the constraints, we see that these are of the form $\min r^*$ s.t. $\exists r_i \geq r^*$. It is far from obvious how, and if, such a system could be set up on such solvers (in which some constraints might be "violated" i.e., be less than the target value of r^*).

Another way to understand the above is to define the objective function value F of a given configuration (values) C for all the variables included. Given $C \in [0,1]^X$ where X is the set of variables, let r_i be the values of the ratios corresponding to the particular solutions. Then $F(C) = \max\{r_i\}$. Our goal is to minimize this objective function value, i.e., to find a configuration on the variables such that $F(C)$ is as small as possible. Observe that for a particular C it might very well be the case that all but one r_is are less than $F(C)$. The objective value is given by the maximum value of all these ratios. This complexity of the objective function is precisely the reason why it is difficult to apply the `solve` environment. There are more complications that arise of technical nature (such as the use of conditions and cases), that will be discussed shortly.

6.2 Selection of the optimizer

So we have to settle with polynomial optimizers that may stuck on local optima and then, applying external knowledge and with the help of repetitive experiments, we try to reach a global optimal solution. For this reason we used two widely used polynomial (non-linear) solvers: The GRG (Generalized Reduced Gradient) solver and the DEPS (Differential Evolution and Particle Swarm Optimization) solver developed in SUN labs.

We will describe in more detail the GRG method and the technical details of the program we set up to achieve the 0.821-approximation guarantee (The DEPS optimizer gave better results). The GRG method allows us to solve non-linear and even non-smooth problems. It has many different options that we exploit in our way to to find a global optimal solution. The GRG algorithm is the convex analog of the simplex method where we allow the constraints to be arbitrary nonlinear functions, and we also allow the variables to possibly have lower and upper bounds. It's general form is the following:

$$\begin{aligned} \max\ (\min)\ & f(\boldsymbol{x}) \\ s.t.\quad & h_i^T(\boldsymbol{x}) = 0\ \forall i \in [m], \boldsymbol{L} \leq \boldsymbol{x} \leq \boldsymbol{U} \end{aligned}$$

where $\boldsymbol{x}$ is the n-dimensional variable vector, h_i is the i-th constraint, and $\boldsymbol{L}$, $\boldsymbol{U}$ are n-dimensional vectors representing lower and upper bounds of the variables. For simplicity we assume that $\mathbf{h}$ is a matrix with m rows (the constraints) and n columns (variables) with rank m (i.e., m linear independent constraints). The GRG method assumes that the set X of variables can be partitioned into two sets (α, β) (let $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ be the corresponding vectors) such that:

1. $\boldsymbol{\alpha}$ has dimension m and $\boldsymbol{\beta}$ has dimension $n - m$;
2. the variables in α strictly respect the given bounds represented by $\boldsymbol{L}_\alpha$ and $\boldsymbol{U}_\alpha$; in other words, $\forall x_i \in \alpha$, $\boldsymbol{L}_{x_i} \leq x_i \leq \boldsymbol{U}_{x_i}$.
3. $\nabla_\alpha h(\boldsymbol{x})$ is non-singular (invertible) at $X = (\alpha, \beta)$. From the Implicit Function Theorem, we know that for any given $\beta \subseteq X$, $\exists \alpha = X \setminus \beta$ such that $h(\boldsymbol{\alpha}, \boldsymbol{\beta}) = 0$. This immediately implies that ${}^{\mathrm{d}\boldsymbol{\alpha}}/_{\mathrm{d}\boldsymbol{\beta}} = (\nabla_{\boldsymbol{\alpha}} h(\boldsymbol{x}))^{-1} \nabla_{\boldsymbol{\beta}} h(\boldsymbol{x})$.

The main idea behind GRG is to select the direction of the independent variables (which are the analog of the non-basic variables of the SIMPLEX method) $\boldsymbol{\beta}$ to be the reduced gradient as follows:

$$\nabla_{\boldsymbol{\beta}} \left(f\left(\boldsymbol{x}\right) - y^T h\left(\boldsymbol{x}\right)\right), \text{ where } \boldsymbol{y} = \frac{\mathrm{d}\boldsymbol{\alpha}}{\mathrm{d}\boldsymbol{\beta}} = \left(\nabla_\alpha h\left(\boldsymbol{x}\right)\right)^{-1} \nabla_{\boldsymbol{\beta}} h\left(\boldsymbol{x}\right)$$

Then, the step size is chosen and a correction procedure applied to return to the surface $h(\boldsymbol{x}) = 0$. The intuition is fairly simple: if, for a given configuration of the values of the variables, a partial derivative has large absolute value, then the GRG would try to change the value of the variable appropriately and observe how its partial derivative changes. The goal is to arrive at a point where all partial derivatives are zero. This can happen to any local or global optimal point. In a few words, the GRG method is viewed as a sequence of steps through feasible points $\boldsymbol{x}^j$ such that the final vector of this sequence satisfied the famous KKT conditions of optimality of non-linear systems.

In order to derive these conditions, we first take the Langrangean of the above problem:

$$\mathcal{L}\left(\boldsymbol{x}, \boldsymbol{\ell}\right) = f\left(\boldsymbol{x}\right) + \sum_{j \in [m]} \ell_j h^j\left(\boldsymbol{x}\right) - \sum_{i \in [n]} L_i\left(x_i - L_i\right) + \sum_{i \in [n]} U_i\left(x_i - U_i\right)$$

At the optimum point $\boldsymbol{x}^*$ the KKT conditions would yield that:

$$\nabla \mathcal{L} = \nabla f\left(\boldsymbol{x}^*\right) + \sum_{j \in [m]} \ell_j \nabla h^j\left(\boldsymbol{x}^*\right) - \left(\boldsymbol{L} - \boldsymbol{U}\right) = 0$$

coupled with the standard constraints derived from the complementary slackness conditions. This is the stopping criterion of an iteration, meaning that we hit a local minimum.

As mentioned above, by setting the objective function value for a given configuration C on the variables X to be $F(C) = \max\{r_i\}$, our goal is to find a feasible C that minimizes $F(C)$. An important thing here is to explain what we mean by “feasible”. Typically, not every assignment of values to variables counts as feasible, because it might violate some obvious restrictions i.e., it might be the case that under a given assignment of values we have $\delta(S_1) \leq \delta(O_1)$ which is of course impossible (remember that S_1 is the set of the k_1 vertices of the *highest* degree in V_1 and so, by definition, they cover more edges than the vertices in the part of the optimum in V_1). So, in order to complete our program, we couple it with all the constraints from block (8):

$$\begin{aligned} &\min F(C) = \max_{i=1}^{6} \{r_i\} \\ &\text{s.t. } (8) \end{aligned}$$

6.3 Implementation

We set up a GRG program with the following details:

Variables. We have one binary variable for each set of edges as depicted in Figure 1 plus π_i, λ_i, $i = 1, \ldots, 5$, plus μ, ν, ξ. Let X be this set of variables. We have $|X| = 48$.

Parameters. We note that in the π-fraction and in the λ-fraction of the solutions SOL_5, and SOL_6, the numbers π and λ are *not* variables, but rather *parameters* that we are free to choose. For the purpose of our experiments, we tried several different values for λ, π. In Table 1, we report results for various different choices of values for parameters λ and π.

Constraints. Expression (8) in Section 2.

Further details. In order to be certain about the optimality of the results, we employ a 2-step strategy. First, we apply a "multistart" on the optimizer. Roughly speaking, the multistart works as follows. We provide a random seed to the optimizer, together with a parameter X, which is a positive integer. Then, we partition the feasible region of the variables (which is a subset of the n-dimensional hypercube $[0, 1]^n$, n = number of variables) into X segments. The selection of X feasible starting points inside the hypercube is done *randomly*. We try to identify the local minimum in the neighborhood of each starting point. The output of the algorithm is the minimum among all these local minima. The intuition is simple: there might be several minima and by selecting randomly different starting points we significantly increase the chance to hit the global optimum. Typical size of X in our experiments is 1000 (which is much greater than the number of different local optima in any case). In other words, after one "cycle" finish (hit of some local minimum) another running immediately starts from a different starting point chosen randomly (which is basically a feasible configuration of the variables).
We run the algorithm 100 independent times. Also, in each iteration, we start the first cycle at a different starting point by selecting a different random seed. The purpose of the random seed is to initiate the algorithm at a random point (feasible or not). This also means that the starting point of the other cycles would be also determined accordingly.

Differencing method. In order to numerically compute the partial derivative of a given configuration, we use the *Central Differencing* method: in order to compute the derivative we use two different configurations on the variables, in the opposite direction of each other, as opposed to the method of forward differencing which uses a single point that is slightly different from the current point to compute the derivative. In more detail, in order to compute the first derivative at point $x_0 \in [0,1]^n$ we use the following (where h is the precision, or the "spacing": typical values of h in our applications are $< 0,00001$):

$$\partial_c f\left(x_0\right) = f\left(x_0 + \frac{1}{2}h\right) - f\left(x_0 - \frac{1}{2}h\right)$$

The central differencing method we used, although more time-consuming since it needs more calculations, is more accurate since, when f is twice differentiable, the term $\partial_c f(x_0)$ divided by the precision h, incurs an error of $O(h^2)$ as opposed to error $O(h)$ that we would have if we were using forward (or backward) differencing. Of course this comes at a cost of time consumption reflected by the more calculations needed to approximate the derivatives, but precision is more important than time in our application.

6.4 Results

In this section we report the results of the GRG program. First, we summarize the results according to the different values of parameters π and λ. One can see that as these values decrease, the approximation guarantee increases. Also, for convenience, we include the approximation guarantee returned by including only the four first rations (excluding $\mathrm{SOL}_5, \mathrm{SOL}_6$ corresponding to the two vertical cuts on V_1 and V_2 respectively; first line in Table 1).

Value of π	Value of λ	Ratio
-	-	0.723269
0.4	0.4	0.754895
0.2	0.00001	0.776595
0.1	0.1	0.780161
0.05	0.1	0.795602
0.0001	0.5	0.807453
0.0001	0.0001	0.805927
0.00001	0.00001	0.821044

Table 1. Results according to the different values of parameters π and λ.

In Table 2, the final results with $\pi = \lambda = 10^{-5}$ are given.

Let us conclude noticing that the non-linear program that we set up, not only computes the approximation ratio of k-`VC_ALGORITHM` but it also provides an experimental study over families of graphs. Indeed, a particular configuration on the variables (i.e., a feasible value assignments on the variables that represent the set of edges $B, C, \ldots$) corresponds to a particular family of bipartite graphs with similar structural properties (characterized by the number of edges belonging to the several cut considered). Given such a configuration, it is immediate to find the ratio of k-VC_ALGORITHM, because we can simply substitute the values of the variables in the corresponding ratios and output the largest one. We can view our program as an *experimental analysis* over all families of bipartite graphs, trying to find the particular family that implements the worst case for the approximation ratio of the algorithm. Our program not only finds such a configuration, but also provides data about the range of approximation factor on other families of bipartite graphs. Experimental results show that the approximation factor for the *absolute majority* of the instances is very close to 1 i.e., ≥ 0.95. Moreover, our program is *independent* on the size of the instance. We just need a particular configuration on the relative value of the variables $B, C, \dots,$ thus providing a compact way of representing families of bipartite graphs sharing common structural properties.

We run the program on a standard $C++$ implementation of the GRG algorithm on a 64-bit Intel Core i7-3720QM@2.6GHz, with 16GB of RAM at 1600MHz running Windows 7 x64 *and* Ubuntu 9.10 x32.

Acknowledgement. The work of the author was supported by the Swiss National Science Foundation Early Post-Doc mobility grand P1TIP2_152282.

References

1. Apollonio, N., Simeone, B.: The maximum vertex coverage problem on bipartite graphs. Discrete Appl. Math. **165** (2014) 37–48
2. Caskurlu, B., Mkrtchyan, V., Parekh, O., Subramani, K.: On partial vertex cover and budgeted maximum coverage problems in bipartite graphs. Proc. TCS'14, LNCS 8705, Springer (2014) 13–26
3. Hochbaum, D.S., Pathria, A.: Analysis of the greedy approach in problems of maximum k-coverage. Naval Research Logistics **45** (1998) 615–627
4. Badanidiyuru, A., Kleinberg, R., Lee, H.: Approximating low-dimensional coverage problems. Proc. SoCG'12, ACM (2012) 161–170
5. Ageev, A.A., Sviridenko, M.: Approximation algorithms for maximum coverage and max cut with given sizes of parts. Proc. IPCO'99, LNCS 1610, Springer (1999) 17–30
6. Galluccio, A., Nobili, P.: Improved approximation of maximum vertex cover. Oper. Res. Lett. **34** (2006) 77–84
7. Feige, U., Langberg, M.: Approximation algorithms for maximization problems arising in graph partitioning. J. Algorithms **41** (2001) 174–211
8. Han, Q., Ye, Y., Zhang, H., Zhang, J.: On approximation of max-vertex-cover. European J. Oper. Res. **143** (2002) 342–355
9. Petrank, E.: The hardness of approximation: gap location. Computational Complexity **4** (1994) 133–157
10. Feige, U., Karpinski, M., Langberg, M.: Improved approximation of max-cut on graphs of bounded degree. J. Algorithms **43** (2002) 201–219
11. Bonnet, E., Escoffier, B., Paschos, V.T., Stamoulis, G.: On the approximation of maximum k-vertex cover in bipartite graphs. CoRR **abs/1409.6952** (2014)

12. Abadie, J., Carpentier, J.: Generalization of the wolfe reduced gradient method to the case of non-linear constraints. In Abadie, J., Carpentier, J., eds.: Optimization. Academic Publishers (1969)
13. Frank, M., Wolfe, P.: An algorithm for quadratic programming. Naval Research Logistics Quarterly **3** (1956) 95–110

Variables	Values	Groups	Values	π, λ	Values	Ratios	Values
B	1	$\delta(S_1)$	5.28490	π	0.00001	r_1	0.81806
C	0.9944	$\delta(S_2)$	5.90033	π_1	0.08471	r_2	0.81797
$F1$	0.0002	$\delta(X_1)$	2.78398	π_2	0.13072	r_3	0.79280
$F2$	0.4954	$\delta(X_2)$	3.09961	π_3	0.97865	r_4	0.79657
$F3$	0.4457	$\delta(O_1)$	5.26489	π_4	0.19364	r_5	**0.82104**
$H1$	0.8449	$\delta(O_2)$	5.88331	π_5	0.38861	r_6	0.82103
$H2$	0.0623	$\delta(OPT)$	10.5589				
$I1$	0			λ	0.00001		
$I2$	0			λ_1	0.14995		
$I3$	0.9986			λ_2	0.76660		
$I4$	0			λ_3	0.15362		
$I5$	0.0577			λ_4	1		
$I6$	0.3740			λ_5	1		
$J1$	0.2386						
$J2$	0.9824						
$J3$	0.3612						
$N1$	1						
$N2$	0.6005						
$P1$	0						
$P2$	0						
$P3$	1						
$P4$	0.7525						
$P5$	0						
$L1$	0.1932						
$L2$	0						
$L3$	0.3960						
$L4$	0						
$L5$	0						
$L6$	0						
$L7$	0						
$L8$	0						
$L9$	0						
$U1$	0.5330						
$U2$	0.3198						
$U3$	0						
μ	0.809						
ν	0						
ξ	0						

Table 2. The final results with $\pi = \lambda = 10^{-5}$.

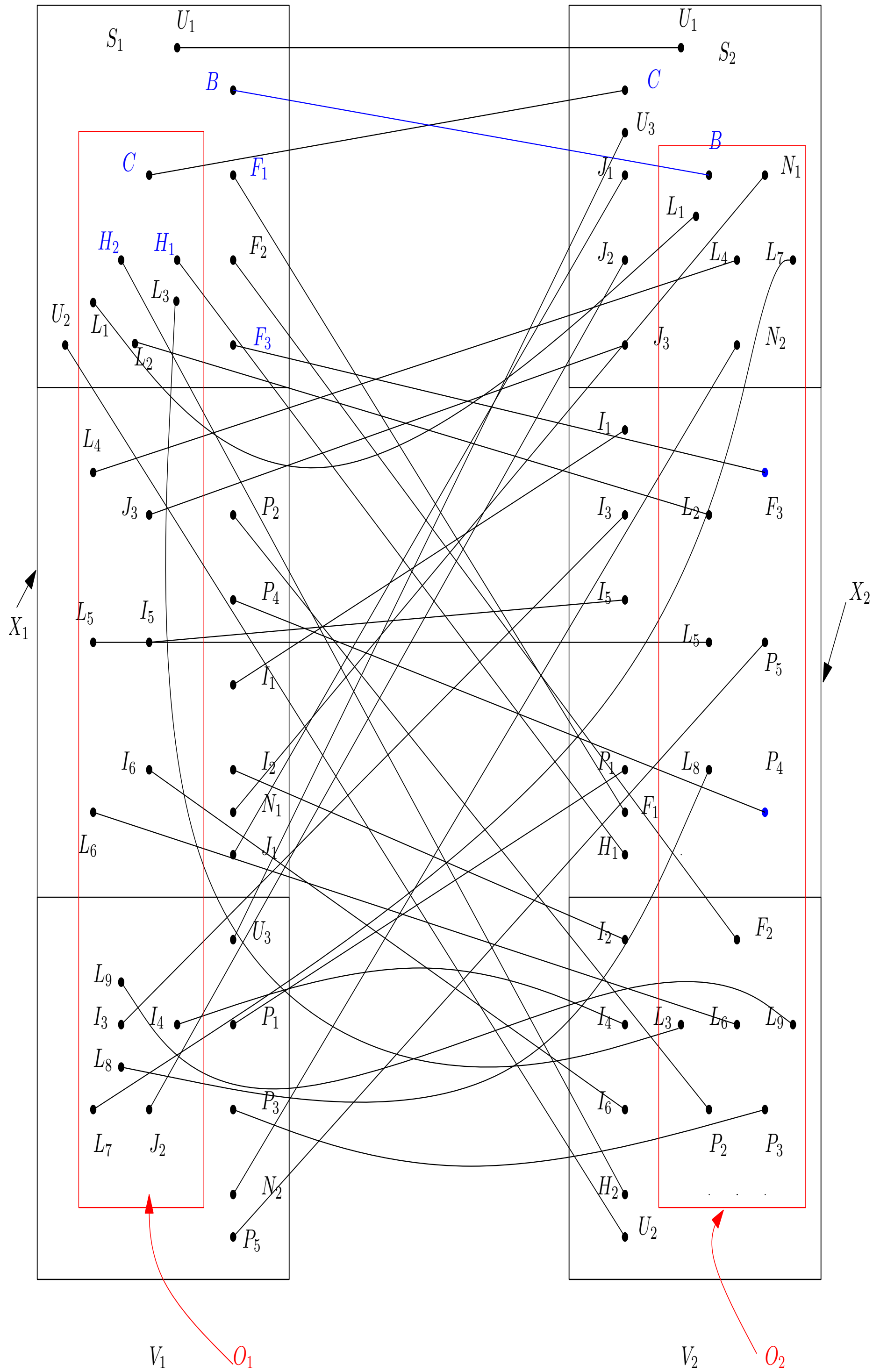

Fig. 2. Sets S_i, O_i, X_i $i = 1, 2$ and cuts between them.