ARTICLE IN PRESS

Discrete Optimization ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Discrete Optimization

www.elsevier.com/locate/disopt

Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs☆

Édouard Bonnet[a], Bruno Escoffier[b], Vangelis Th. Paschos[c], Georgios Stamoulis[d,*]

[a] *Department of Computer Science, Middlesex University, London, UK*
[b] *Sorbonne Universités, UPMC Universite Paris 6, CNRS, LIP6 UMR 7606, France*
[c] *Université Paris-Dauphine, PSL* Research University, CNRS UMR 7243, LAMSADE, Paris 75016, France*
[d] *Department of Data Science and Knowledge Engineering, Maastricht University, The Netherlands*

ARTICLE INFO

Article history:
Received 25 January 2016
Received in revised form 29 August 2017
Accepted 1 September 2017
Available online xxxx

Keywords:
Approximation algorithms
Combinatorial algorithms
Non linear program
Graph algorithms
Maximum coverage

ABSTRACT

We study the polynomial time approximation of the **NP**-hard MAX k-VERTEX COVER problem in bipartite graphs and propose purely *combinatorial approximation algorithms*. The main result of the paper is a simple combinatorial algorithm and a computer-assisted analysis of its approximation guarantee giving strong evidence that the worst approximation ratio achieved is bounded below by 0.821. We also study two simpler strategies with provable approximation ratios of $2/3$ and $34/47 \approx 0.72$ respectively that already beat the only such known algorithm, namely the greedy approach which guarantees ratio $(1-1/e) \approx 0.632$. Our principal motivation is to bring a satisfactory answer in the following question: to what extent combinatorial methods for MAX k-VERTEX COVER compete with linear programming ones?

1. Introduction

In MAX k-VERTEX COVER problem, a simple, non-directed graph $G = (V, E)$ is given together with a positive integer $k \leqslant n$. As usual, let $|V| = n$ and $|E| = m$. The goal is to find a subset $K \subseteq V$ with k vertices 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 $K \cap e \neq \emptyset$. In the weighted version of MAX k-VERTEX COVER, positive weights are assigned to the edges and the objective becomes to determine k vertices that maximize the total weight of the edges covered by them. MAX k-VERTEX COVER is **NP**-hard in general graphs (as a generalization of MIN VERTEX COVER) and the **NP**-completeness in bipartite graphs has been only recently established [1,2].

☆ An extended abstract of the paper has been presented in LATIN 2016.

* Corresponding author.
E-mail addresses: edouard.bonnet@lamsade.dauphine.fr (É. Bonnet), bruno.escoffier@lip6.fr (B. Escoffier), paschos@lamsade.dauphine.fr (V.Th. Paschos), georgios.stamoulis@maastrichtuniversity.nl (G. Stamoulis).

http://dx.doi.org/10.1016/j.disopt.2017.09.001
1572-5286/

MAX k-VERTEX COVER is a well-known restriction of MAX k-SET COVER problem where we are given a family of subsets $\mathcal{S}$ over a set of elements C and an integer k and the objective is to determine a subfamily $\mathcal{S}' \subseteq \mathcal{S}$ of cardinality k, covering a maximum number of elements from C. Analogously, in the weighted version of MAX k-SET COVER, the elements of C are provided with positive weights and the objective becomes to determine k sets that maximize the total weight of the covered elements.

Both MAX k-SET COVER and MAX k-VERTEX COVER are well-known problems met in many real-world applications since they model several very natural facility location problems. In particular MAX k-VERTEX COVER is used for modeling problems in areas such as databases, social networks, sensor placement, information retrieval, etc. A non-exhaustive list of references to such applications can be found in [3].

To the best of our knowledge, the approximation of the former has been studied for the first time in the late seventies by Cornuejols et al. [4], where an approximation ratio $1 - (1/e)$ (≈ 0.632) is proved for the natural greedy algorithm, consisting of iteratively choosing the currently largest-cardinality set, until k sets are included in the solution. Obviously, since MAX k-VERTEX COVER is a restriction of MAX k-SET COVER the same ratio is achieved for the former problem also. This ratio is tight in (weighted) bipartite graphs [3]. A more systematic study of the greedy approximation of MAX k-VERTEX COVER can be found in [5]. More recently, it has been proved in [6] that the greedy algorithm also achieves ratio k/n. In the same paper, a very simple randomized algorithm is presented, that achieves approximation ratio $2(k/n) - (k/n)^2$. Using a sophisticated linear programming method, the approximation ratio for MAX k-VERTEX COVER, in general graphs was improved to $3/4$ [7]. A direct reduction from MIN VERTEX COVER shows that the general MAX k-VERTEX COVER cannot admit a polynomial time approximation schema (PTAS), unless $\mathbf{P} = \mathbf{NP}$ [8]. The best approximation ratio in bipartite graphs is now $8/9$ and is based on highly non-trivial linear programming approaches [2]. This improves over the $4/5$-approximation algorithm, based again on LP techniques, from [9]. MAX k-VERTEX COVER is shown to be polynomial time solvable in *line* graphs [10]. This is a remarkable result because it shows that line graphs are instances where both the classical MIN VERTEX COVER and the MAX k-VERTEX COVER problems are solvable in polynomial time. Finally we note that MAX k-VERTEX COVER rather specializes to the dual of MAX k-SET COVER, namely, where sets are stars of vertices.

We note that it is easy to observe that unweighted MAX k-VERTEX COVER is easy on semiregular bipartite graphs (where all the vertices of each color class have the same degree). Indeed, any k vertices in the color class of maximum degree yield an optimal solution. Obviously, if this color class contains less than k vertices, then one can cover *all* the edges.

The principal question motivating this paper is:

> *to what extent combinatorial methods for this problem can compete with linear programming ones.* In other words, *what is the ratio that a purely combinatorial algorithm can guarantee for* MAX k-VERTEX COVER *in bipartite graphs?*

Such research direction is barely a new phenomenon and similar questions for other problems have also motivated several more or less recent research works. Probably the most remarkable and well-known result towards this direction is [11] where a 0.531-ratio combinatorial algorithm is given for the MAX CUT problem. Although a 0.878-approximation guarantee exists due to Goemans–Williamson SDP [12], it was a major open problem if, and how, purely combinatorial methods could potentially achieve something better than $1/2$ in polynomial time. Also very recently, in [13] a similar issue is presented for the Steiner forest problem which is very well understood from LP point of view but not from combinatorial side. For an example to the opposite direction we mention [14] where approximation of the MAX k-DIMENSIONAL MATCHING is handled for $k \geqslant 3$ and it is shown that linear programming methods do not succeed in producing approximation factors achievable by combinatorial methods (based upon local search for the particular problem family). The latter is considered to be a specific part of a much more deep question [15]: *"are LP/SDP techniques capable of simulating local search heuristics, even in quasi-polynomial time?"* The answer is not known but,

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

as the two previous publications suggest, this is an intensive area of current research going very well beyond the scope of simple LP/SDP programs.

In any case, as it is apparent, comparison of different classes of methods with respect to their abilities to solve problems seems to be a very interesting research issue. This may bring new insights to both on the problems handled and the methods themselves. Furthermore, such studies may exhibit interesting and intriguing mathematical problems.

From a practical point of view, the value of combinatorial methods should be obvious: they are in general much more compact, easier to handle and to program and do not depend on manually constructing an LP, by adding all underlying constraints, for each particular instance we would like to solve in practice. But there is probably an even greater value on combinatorial methods: it is not uncommon that such combinatorial insights on a particular problem pave the way for designing even better approximation guarantees. The most famous example to that direction is the SteinerTree problem where the purely combinatorial algorithmic insights [16] were translated into genuine LP arguments [17] to device new approximation guarantees not achievable with prior pure LP or pure combinatorial methods.

Our contribution. As mentioned just above, the main motivation of this paper is to explore the efficiency of combinatorial methods for approximating the weighted max k-vertex cover in bipartite graphs. Our main contribution consists of a purely combinatorial approximation algorithm which computes six distinct and simple solutions and returns the best among them.

Although the algorithm (consisting of 6 greedy solutions) is quite simple and natural, there is a major difficulty in analyzing the performance guarantee of such an algorithm. Indeed, it seems that there is no easy way to compare different solutions and argue globally over all of them. This difficulty motivates us to provide analytic expressions for all the solutions produced, a task 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).

By setting up a suitable non-linear (in fact not even convex) 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 remark that, due to the complicated nature of the non-convex program, our solution is not guaranteed to be optimal. On the other hand, several successive executions with different initial configurations did not provide smaller value for the approximation guarantee. This gives high evidence that the true approximation guarantee is indeed (extremely close to) 0.821. See also the discussion at the end of the corresponding section that provides also further evidence of the possible optimality of our result. Even if this ratio is dominated by the $8/9$-ratio of [2] within a factor of about 8%, it is obtained using only greedy-like combinatorial solutions on the contrary of the latter that is obtained by highly non-trivial linear programming arguments, and already dominates the $4/5$-ratio also linear programming-based approximation algorithm of [9].

We note that similar situations, where a solution could not easily be extracted by the mathematical formulas involved, were faced, for example, in [18] 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 and also, more recently, in [19] for the max bisection problem.

Finally, we devise a simple greedy like algorithm already achieving $2/3$ approximation guarantee and a more involved one achieving approximation ratio at least $34/47 \approx 0.7234$. This result is proven analytically, in contrast with the computer assisted proof of our main result. This should be seen as a restriction of our more involved analytical expressions that they can actually be mathematically manipulated.

Structure of the manuscript. In the next section we provide all the preliminaries needed for the sequels. We group and provide all the necessary definitions and all other concepts needed in our analysis. We give the main block of our algorithm, and we set up the bipartite abstraction by partitioning the set of edges into each possible region. Then, we provide some expressions involving these sets of edges. Intuitively, each

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

set of edges should correspond to a $[0, 1]$ variable and all the relationships and expressions between these set of edges should provide us with the constraints. In Section 3 we provide the 6 solutions that are part of our algorithm and using the results of Section 2 we provide analytical expressions about their value. The purpose of Sections 4 and 5 is to provide the results and all the details, respectively, of the non-convex program that arises by all these expressions obtained in Section 3. We finish the paper with providing two simpler algorithms: a $2/3$- and a $34/77 \approx 0.7234$-ratio algorithm. The latter can be seen as a simpler restriction of the main algorithm of this paper so any result for this immediately carries over the main algorithm.

2. Preliminaries

Here we will provide all the necessary definitions and concepts and we will also define our bipartite abstraction model. We will prove all the properties needed for the sequel.

Consider an edge-weighted bipartite graph $G = (V_1, V_2, E, \boldsymbol{w})$, where $\boldsymbol{w}$ is weight-vector of dimension $|E|$ and an optimal solution O (i.e., a vertex-set on k vertices covering a maximum weight of edges in E). Denote by O_1 and O_2 the subsets of O lying in the color-classes V_1 and V_2, respectively and suppose that $|O_1| = k_1$ and $|O_2| = k_2$ $(k_1 + k_2 = k)$. Without loss of generality suppose that $k_1 \leqslant k_2$. Let weighted degree of a vertex v denote the quantity:

$$\delta(v) \ = \sum_{e \text{ incident } v} w(e)$$

i.e., the sum of the weights of the edges incident to v. Analogously, we will denote by $\delta(V')$, $V' \subseteq V$, the total weight of the edges covered by V'.

For the rest of the paper, we define and call "best" vertices to be a set of vertices that cover the largest total weight of *uncovered* edges in G. For instance, saying, for some vertex set S (suppose that $S \subset V_1$) "we take S plus the χ best vertices in V_2", this means that we first take S and then χ vertices of highest weighted degree in $G[(V_1 \setminus S), V_2]$.

Now we define the following sets of vertices that we will be using repeatedly in the rest of the paper.

Definition 1 (*Sets S_i, X_i*)**.** For $i \in \{1, 2\}$ let S_i be the set of the q first vertices in V_i after we have sorted the vertices in V_i in a decreasing order according to their weighted degree, for some positive integer q. According to the same ordering in V_i, let now X_i be the set of the q best vertices in the subgraph $G[V \setminus S_i]$.

For a fixed $q \in [k]$, S_i would contain the "best" q vertices from V_i with respect to the sum of the weights of the edges they cover (this sum being called sometimes "coverage potential") and X_i would contain the "second best" disjoint set of vertices with respect to their coverage potential.

2.1. Basic algorithmic idea

The algorithm we are considering for MAX k-VERTEX COVER (called k-`VC_ALGORITHM` in the sequel) is the following:

1. order the vertices of both V_1 and V_2 in decreasing order of weighted degree;
2. for any pair (χ_1, χ_2) such that $\chi_1 + \chi_2 = k$ do:
   (a) take the sets S_i of the χ_i first vertices in V_i, $i = 1, 2$;
   (b) "guess" the cardinalities χ_i' of the intersections $S_i \cap O_i$, $i = 1, 2$;
   (c) for $i = 1, 2$ compute the sets X_i of the $\chi_i - \chi_i'$ best vertices from V_i in the graphs $G[(V \setminus S_1), V_2]$ and $G[V_1, (V_2 \setminus S_2)]$, respectively;
3. choose the best among six solutions built as described in Section 3.

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

In step 2(a), the set S_i is obviously a set of the χ_i largest weighted degree vertices in V_i (breaking ties arbitrarily). The guessing step in 2(b) can be done, as usual, by an exhaustive search over all possible values $\chi'_i = 1, 2, \ldots |S_i|$. This takes of course polynomial time.

Obviously, since the above outlined algorithm is executed for any pair (χ_1, χ_2) of values such that $\chi_1 + \chi_2 = k$, it will be executed for the particular pair corresponding to $\chi_1 = k_1$ and $\chi_2 = k_2$. Furthermore, since at the end the best among the solutions computed is retained, this solution will be at least as good as the one corresponding to pair (k_1, k_2). So, in what follows and for simplicity, we will reason with respect to the solution corresponding to this pair and we will use k_i and k'_i instead of χ_i and χ'_i, $i \in \{1, 2\}$.

In the last step of the algorithm, 6 distinct and "greedy"-like solutions are computed and the one with the maximum value among them is considered. This is the topic of the corresponding section, but the idea is to define 6 different solutions that intuitively will behave different with respect to bad instances in a complementary way (so in total there would not exist a bad instance that is bad for each of the 6 solutions at the same time).

2.2. Bipartite abstraction model

In this section we will provide our abstraction of the bipartite graph. The idea is that the six sets S_i, X_i, O_i form natural sets of vertices in the bipartite graph. Indeed, the sets S_i, X_i and O_i separate each color-class in 6 regions, namely:

1. $S_i \cap O_i$,
2. $S_i \setminus O_i$,
3. $X_i \cap O_i$,
4. $X_i \setminus O_i$,
5. $O_i \setminus (S_i \cup X_i)$ (denoted by $\bar{O}_i$, in what follows) and
6. $V_i \setminus (S_i \cup X_i \cup O_i)$.

So, there exist in total 36 groups of edges (cuts) among them (each group on one side can be connected to each of the 6 groups of the other side), 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 since these edges are neither part of the optimal solution O, nor of any solution built by k-VC_ALGORITHM. So they will not appear in any ratio and can be ignored.

We will use the following notations to refer to the values of the 35 relevant cuts:

B: the total weight of the cut $(S_1 \setminus O_1, S_2 \cap O_2)$;
C: the total weight of the cut $(S_2 \setminus O_2, S_1 \cap O_1)$;
F_1, F_2, F_3: the total weight of 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 total weight of 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 total weight of 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 total weight of 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 total weight of 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 total weight of 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 total weight of 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;

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

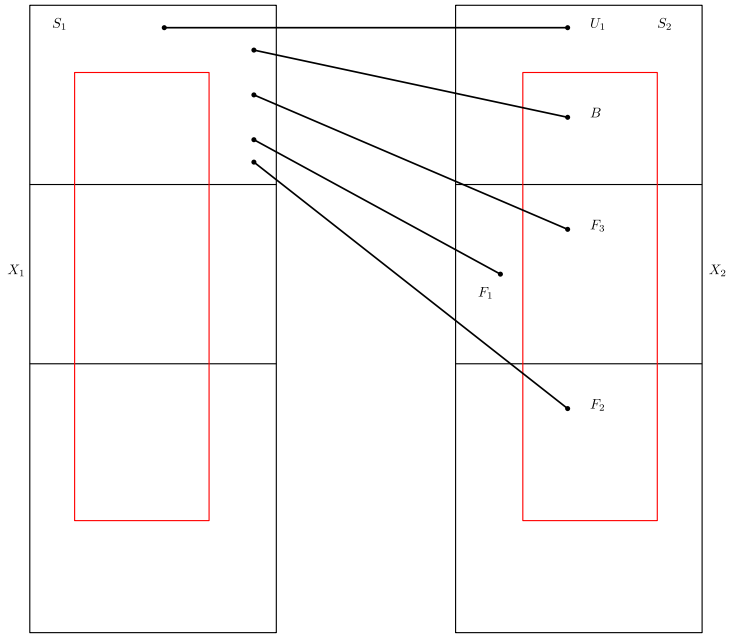

Fig. 1. The cuts between $(S_1 \setminus O_1)$, V_2.

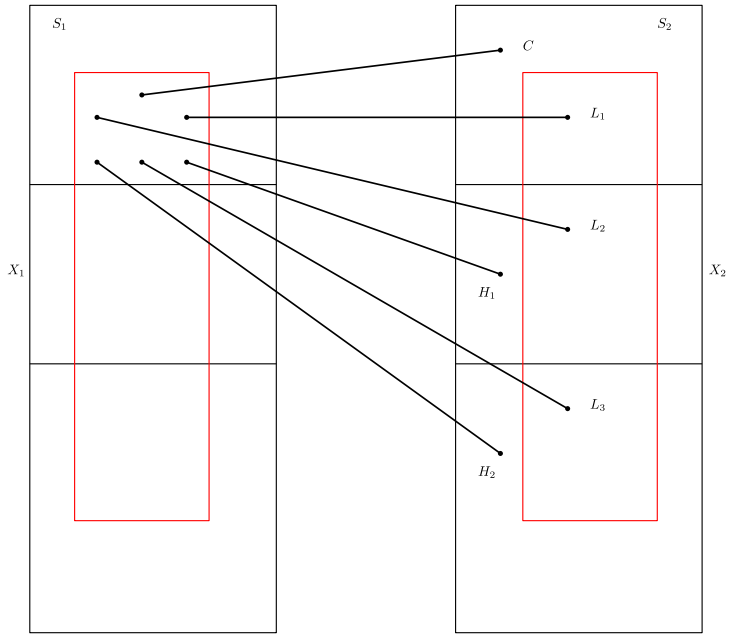

Fig. 2. The cuts between $(S_1 \cap O_1)$, V_2.

U_1, U_2, U_3: the total weight of 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.

All the relevant sets of edges are depicted in Figs. 1–6, while Fig. 8 on page 30 of the paper is a summary of all these cuts into the whole graph. The description of the above sets of edges is as follows: Fix a positive integer χ_1 as in the description of the pseudocode k-`VC_ALGORITHM`. Fig. 1 corresponds to the set of edges

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

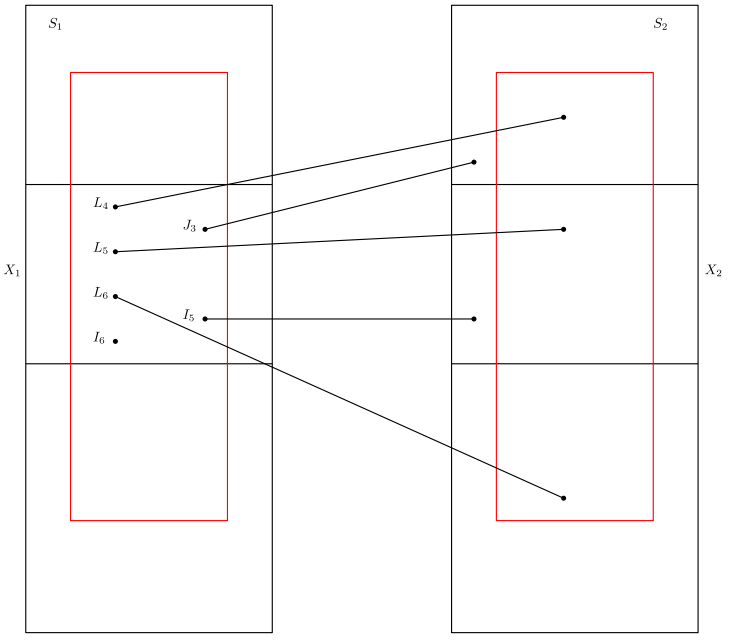

Fig. 3. The cuts between $(X_1 \cap O_1)$, V_2.

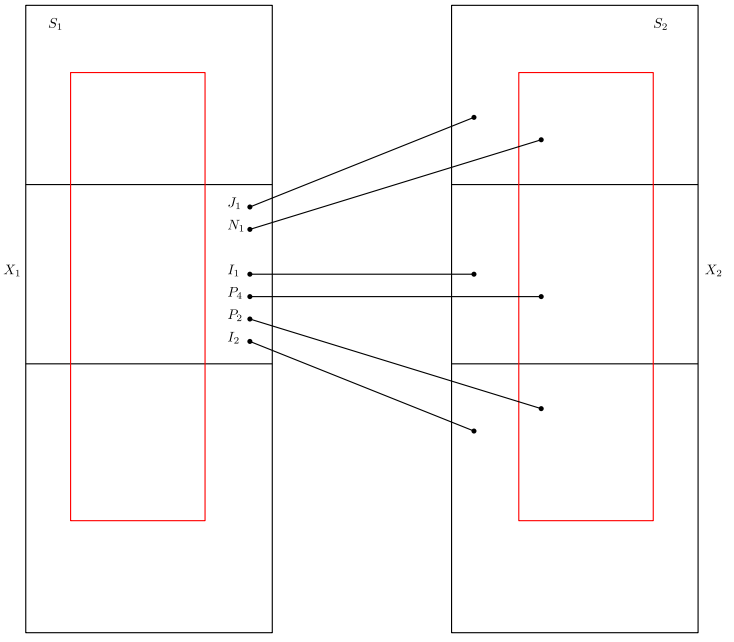

Fig. 4. The cuts between $(X_1 \setminus O_1)$, V_2.

in $S_1 \setminus O_1$, i.e., the edges that are incident to the best χ_1 vertices in V_1 (called S_1) that are *not* part of the vertices in the optimal solution segment O_1 in V_1. Fig. 2 depicts the set of edges from that are incident to vertices in O_1 that are among the best χ_1 vertices defined by S_1. Fig. 3 shows all the edges incident to vertices that are in $X_i \cap O_i$ i.e., vertices in the optimal solution that are among the $\chi_1 - \chi_1'$ best vertices in $V_1 \setminus S_1$. Fig. 4 has the edges incident to vertices in X_1 but do not belong in the optimal solution part O_1.

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

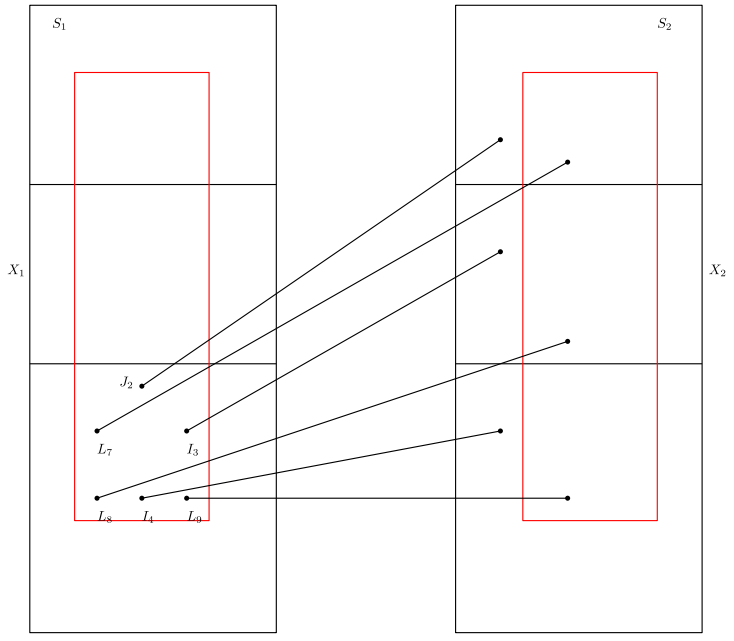

Fig. 5. The cuts between $(O_1 \setminus \{S_1 \cup X_1\})$, V_2.

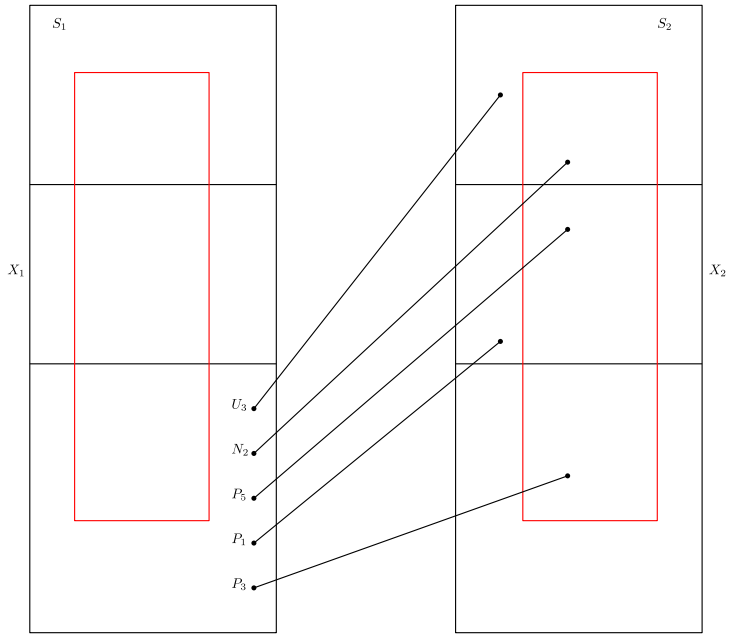

Fig. 6. The cuts between $(V_1 \setminus \{S_1 \cup X_1 \cup O_1\})$, V_2.

Fig. 5 depicts the edges that are incident to vertices in the optimal solution part of V_1, denoted by O_1, that do not belong in the first $|S_1| + |X_1| = 2\chi_1 - \chi'_1$ best vertices of V_1. Similarly, Fig. 6 depicts the rest of the edges.

Based upon the notations above and denoting by $\text{opt}(G)$ the value of an optimal solution (i.e., the total weight covered) for MAX k-VERTEX COVER in the input graph G, the following holds (see also Figs. 1–6):

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

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

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

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

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

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

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

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

As mentioned above, we assume $k_1 \leqslant k_2$ and so we set:

- $k_1 = \mu k_2$ $(\mu \leqslant 1)$; in other words:

$$k = (1 + \mu) \cdot k_2 \tag{8}$$

- $k'_1 = |S_1 \cap O_1| = \nu k_1$ $(0 \leqslant \nu \leqslant 1)$;
- $k'_2 = |S_2 \cap O_2| = \xi k_2$ $(0 \leqslant \xi \leqslant 1)$.

Let us note that, since k'_i vertices lie in the intersections $S_i \cap O_i$, the following inequalities hold for $\bar{O}_i = O_i \setminus (S_i \cup X_i)$, $i = 1, 2$:

$$|\bar{O}_1| = |O_1 \setminus (S_1 \cup X_1)| \leqslant |O_1 \setminus S_1| = (1 - \nu)k_1 = (1 - \nu)\mu k_2 \tag{9}$$

$$|\bar{O}_2| = |O_2 \setminus (S_2 \cup X_2)| \leqslant |O_2 \setminus S_2| = (1 - \xi)k_2. \tag{10}$$

From the definitions of the cuts and using (1) to (6) and the expressions for $|\bar{O}_1|$ and $|\bar{O}_2|$ in (9) and (10), simple average arguments and the assumptions for k_1, k_2, k'_1 and k'_2 just above, the following lemma holds.

Lemma 1. *The following inequalities hold:*

$$\begin{aligned}
&\delta(S_1) && \geqslant \delta(O_1)\\
&\delta(S_2) && \geqslant \delta(O_2)\\
&\delta(X_1) + C + H_1 + H_2 + L_1 + L_2 + L_3 && \geqslant \delta(O_1)\\
&\delta(X_2) + B + N_1 + N_2 + L_1 + L_4 + L_7 && \geqslant \delta(O_2)\\
&\delta(S_1) && \geqslant \left(\frac{1}{1-\nu}\right) \cdot \delta(X_1)\\
&\delta(S_2) && \geqslant \left(\frac{1}{1-\xi}\right) \cdot \delta(X_2)\\
&\delta(S_1) + \delta(X_1) && \geqslant \left(\frac{2-\nu}{1-\nu}\right) \cdot (I_3 + I_4 + J_2 + L_7 + L_8 + L_9)\\
&\delta(S_2) + \delta(X_2) && \geqslant \left(\frac{2-\xi}{1-\xi}\right) \cdot (F_2 + L_3 + L_6 + L_9 + P_2 + P_3)\\
&B + F_1 + F_2 + F_3 + U_1 + U_2 && \geqslant \delta(X_1)\\
&C + J_1 + J_2 + J_3 + U_1 + U_3 && \geqslant \delta(X_2).
\end{aligned} \tag{11}$$

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

Proof. For $i = 1, 2$, the two first inequalities in (11) hold because S_i is the set of k_i highest weighted-degree vertices in V_i.

The third and fourth ones because the left-hand side quantities are the total weights of edges covered by $X_i \cup (S_i \cap O_i)$; each of these sets has cardinality k_i and obviously covers more weight than O_i.

The fifth and sixth inequalities because the average weighted degree of S_i is at least the average weighted 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 weighted degree of vertices in $S_i \cup X_i$ is at least the average weighted degree of vertices in $O_i \setminus (S_i \cup X_i)$.

Finally, for the last two inequalities the sum of weighted degrees of the $k_i - k'_i$ vertices in $S_i \setminus O_i$ is at least the sum of weighted degrees of the $k_i - k'_i$ vertices of X_i. □

Given a solution $\mathrm{SOL}_q(G)$, we denote by $\mathrm{sol}_q(G)$ its value. For the quantities implied in the ratios corresponding to these solutions, one can be referred to Figs. 1–6 and to expressions (1)–(7).

Observe 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 easy remark will be useful for some solutions presented in Section 3, for example in the completion of solution $\mathrm{SOL}_5(G)$.

3. Six solutions for the bipartite max k-vertex cover

In this section we provide the 6 simple greedy solutions that constitute the main ingredient of the "block" algorithm k-VC_ALGORITHM described in previous section. For simplicity, as it has been discussed in Section 2, we only present its execution for the values $(\chi_1, \chi_2, \chi'_1, \chi'_2) = (k_1, k_2, k'_1, k'_2)$ that corresponds to step (3) of the main algorithm of the previous section.

Compute sets S_i of k_i best vertices in V_i, $i = 1, 2$ and 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:

- $\mathbf{SOL_1(G)}$ and $\mathbf{SOL_2(G)}$ 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;
- $\mathbf{SOL_3(G)}$ 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;
- $\mathbf{SOL_4(G)}$ takes S_2 and completes it either with vertices from V_2, or with vertices from both V_1 and V_2;
- $\mathbf{SOL_5(G)}$ 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;
- $\mathbf{SOL_6(G)}$ 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* taking values in $[0, 1]$. Let us also note that the algorithm above, since it runs for any value of χ_1 and χ_2, it will run for $\chi_1 = k$ and $\chi_2 = k$. So, it is optimal for the instances of [3], where the greedy algorithm attains the ratio $(e-1)/e$. Let us also note that, besides their apparent simplicity, one can actually program these solutions in parallel and then simply compare their solutions.

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

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

3.1. Solution $SOL_1(G)$

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:

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

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

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

3.2. Solution $SOL_2(G)$

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:

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

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

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

3.3. Solution $SOL_3(G)$

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:

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

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

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

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 $\mathrm{SOL}_3(G)$ is:

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

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

3.4. Solution $SOL_4(G)$

Once S_2 is 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 the weight of the edges incident to X_2, or a $\mu/2(1-\xi)$ fraction of the weight of the edges incident to $X_2 \cup \bar{O}_2$, i.e., at least an edge-weight $\mathcal{M}_1$, where $\mathcal{M}_1$ is given by:
$$\max\left\{\frac{\mu}{1-\xi}\delta\left(X_2\right), \frac{\mu}{2(1-\xi)}\left(\left(X_2\right) + F_2 + L_3 + L_6 + L_9 + P_2 + P_3\right)\right\} \tag{18}$$
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
   $$\min\left\{1, \frac{\mu - 1 + \xi}{\left|\bar{O}_2\right|}\right\} \geqslant \min\left\{1, \frac{\mu - 1 + \xi}{1 - \xi}\right\}$$
   fraction of the total weight of the 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 the total weight 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 the total weight 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 the total weight of uncovered edges incident to this vertex-set (quantity $\mathcal{M}_5$ in (19));

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

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

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

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

3.5. Vertical separations—solutions $SOL_5(G)$ and $SOL_6(G)$

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 weighted degree) vertices of V'. Then, the following easy claim holds for a vertical separation of $V' \cup V''$ with parameter c.

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

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

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 weighted degree of V' is $\delta(V')/n'$. The average weighted degree of $A(V')$ is $\delta(A(V'))/cn'$. But, from the selection of $A(V')$ as the cn' vertices with highest weighted degree, we have that $\delta(A(V'))/cn' \geqslant \delta(V')/n' \Rightarrow \delta(A(V')) \geqslant c\delta(V')$. Similarly for V'', i.e., $\delta(A(V'')) \geqslant c\delta(V'')$. □

Solutions $\mathrm{SOL}_5(G)$ and $\mathrm{SOL}_6(G)$ 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 us 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 $\mathbf{SOL_5(G)}$. It consists of *separating $S_1 \cup X_1$ with parameter $\pi \in (0, 1/2]$, i.e., 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 weights 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, $j = 1, \ldots, 6$, 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, $i = 1, \ldots, 6$ 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 $\mathrm{sol}_5(G)$. 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. $\mu(1-2\pi)+\mu\nu\pi \leqslant 1-\xi$; 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:

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

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

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:

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

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:

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

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

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:

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

2. $\mu(1-2\pi)+\mu\nu\pi \geqslant 1-\xi$; 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:

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

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:

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

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

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:

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

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:

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

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:

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

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:

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

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

$$\frac{\sum_{i=1}^{6}\pi_i\Pi_i + \begin{cases}\begin{cases} Z^* & \text{if } \mu(1-2\pi)+\mu\nu\pi \leqslant 1-\xi \\ \Theta^* & \text{if } \mu(1-2\pi)+\mu\nu\pi \geqslant 1-\xi\end{cases} & \text{case: } 1-\mu(2\pi-1)+\mu\nu\pi \geqslant 1 \\ \Phi^* & \text{case: } 1-\mu(2\pi-1)+\mu\nu\pi < 1\end{cases}}{\mathrm{opt}(G)}. \tag{36}$$

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

Solution $\mathbf{SOL_6(G)}$. Symmetrically to $\mathrm{SOL}_5(G)$, solution $\mathrm{SOL}_6(G)$ 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:

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

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 weights 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:

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

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, $i = 1, \ldots, 6$ 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 $\mathrm{sol}_6(G)$. We again distinguish two cases.

Case 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:

1. $1-\lambda(2-\xi) \leqslant (1-\nu)\mu$; 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 $\Upsilon^* = \max\{\Upsilon_i, i = 1, \ldots, 5\}$, where:
   Υ_1 refers to S_1 plus the best $(1-\lambda(2-\xi))k_2$ vertices of X_1 having a contribution of:

$$\begin{aligned} \Upsilon_1 = &\sum_{i=1}^{2} (1-\lambda_i)\,\Lambda_i + (H_2 + F_2 + L_3 + U_2) \\ &+ \frac{1-\lambda(2-\xi)}{\mu(1-\nu)} \left[\sum_{i=3}^{4} (1-\lambda_i)\,\Lambda_i + (I_2 + I_6 + L_6 + P_2)\right] \end{aligned} \tag{39}$$

   Υ_2 refers to S_1 plus the best $(1-\lambda(2-\xi))k_2$ vertices of O_1 having a contribution of:

$$\begin{aligned} \Upsilon_2 = &\sum_{i=1}^{2} (1-\lambda_i)\,\Lambda_i + (H_2 + F_2 + L_3 + U_2) + \frac{1-\lambda(2-\xi)}{\mu(1-\nu)}\,[(1-\lambda_3)\,\Lambda_3 \\ &+ (1-\lambda_5)\,\Lambda_5 + (I_4 + I_6 + L_6 + L_9)] \end{aligned} \tag{40}$$

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

Υ_3 and Υ_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:

$$\Upsilon_3 = \frac{\mu+1-\lambda(2-\xi)}{\mu(2-\nu)} \left[\sum_{i=1}^{4} (1-\lambda_i)\, \Lambda_i + (F_2 + H_2 + I_2 + I_6 + L_3 + L_6 + P_2 + U_2) \right] \tag{41}$$

$$\Upsilon_4 = \frac{\mu+1-\lambda(2-\xi)}{\mu(2-\nu)} \left[\sum_{i=1}^{3} (1-\lambda_i)\, \Lambda_i + (1-\lambda_5)\, \Lambda_5 + (F_2 + H_2 + I_4 + I_6 + L_3 + L_6 + L_9 + U_2) \right] \tag{42}$$

Υ_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:

$$\Upsilon_5 = \frac{\mu+1-\lambda(2-\xi)}{\mu(3-2\nu)} \left[\sum_{j=1}^{5} (1-\lambda_j)\, \Lambda_j + (F_2 + H_2 + I_2 + I_4 + I_6 + L_3 + L_6 + L_9 + P_2 + U_2) \right] \tag{43}$$

2. $1-\lambda(2-\xi) \geqslant (1-\nu)\mu$; 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 $\Psi^* = \max\{\Psi_i, i = 1, \ldots, 3\}$, where:
Ψ_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:

$$\Psi_1 = \sum_{j=1}^{4} (1-\lambda_j)\, \Lambda_j + (F_2 + H_2 + I_2 + I_6 + L_3 + L_6 + P_2 + U_2) + \frac{1-\lambda(2-\xi)-\mu(1-\nu)}{\mu(1-\nu)} [(1-\lambda_5)\, \Lambda_5 + I_4 + L_9] \tag{44}$$

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:

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

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:

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

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

Case 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:

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

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

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

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

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

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

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

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:

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

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

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

4. Results

To analyze the performance guarantee of k-VC_ALGORITHM, we set up a non-linear program and solved it. Here, we interpret the values of cuts $B, C, F_i, \ldots$, as *variables* , the expressions in (11) 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 assignment to the set of

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

Table 1
The final results with $\pi = \lambda = 10^{-5}$.

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						

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 [20]) program. The reasons this method is selected are presented in Section 5, as well as a more detailed description of the implementation. GRG is a generalization of the classical *Reduced Gradient* method [21] for solving (concave) quadratic problems so that it can handle higher degree polynomials and incorporate non-linear constraints. Table 1 in the following Section 5 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.

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

Remark. As we note in Section 5, 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.

5. A computer assisted analysis of the approximation ratio of k-VC_ALGORITHM

In this section we provide all the implementation details and a brief discussion about all the issues regarding the optimizer and about the set-up of our non-convex program.

5.1. 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 values (total weights) 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. Let us note that, this is not a convex program, so we cannot hope to formulate it as a semi-definite program.

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.

An important issue regarding the set-up of our program is that 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. $\exists r_i \geqslant r^*$. 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^*).

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

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.

5.2. 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 the GRG (Generalized Reduced Gradient) solver [20,22] (which is an extension of the famous Reduced Gradient method of Wolfe & Frank [21]. See also the book [23] for a modern and pedagogically nice treatment of this and related topics. 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 $\delta(S_1) \leqslant \delta(O_1)$ 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 (11):

$$\begin{array}{ll} \min & F(C) = \max_{i=1}^{6} \{r_i\} \\ \text{s.t.} & (11). \end{array}$$

5.3. 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 Figs. 1–6 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. correspond to $\pi = \lambda = 0.05$. In Table 2, we report results for various different choices of values for parameters λ and π.

Constraints. Expression (11) 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. This 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).

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

Table 2
Results according to the different values of parameters π and λ.

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

Differencing method. In order to numerically compute the partial derivative of a given configuration, we use the *Central Differencing* method [24,25]. 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):

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

5.4. Values of the several variables and parameters

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 2). In Table 1, the final results with $\pi = \lambda = 10^{-5}$ are given.

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

6. Some simpler approximation results for bipartite max k-vertex cover

The purpose of this final section of the paper is to provide two simple combinatorial approximation algorithms for the MAX k-VERTEX COVER. The first, and simpler one, has an approximation guarantee of $2/3$ and the second, more complicated, a guarantee of ≈ 0.7234. The first one already beats the natural greedy, which up to date, was the best combinatorial algorithm for MAX k-VERTEX COVER. The second one can be seen as a restriction of the main algorithm of this paper. In particular, when we simplify the solutions involved and their expressions, we can much more easily algebraically argue about them and prove that they provide a more or less satisfactory solution.

6.1. An easy $2/3$-approximation algorithm

We first propose a $2/3$-approximation algorithm for MAX k-VERTEX COVER, which goes as follows. Consider a bipartite graph $G = (V_1, V_2, E)$ and the integer k.

1. For k_1 from 0 to k:
   - ▷ Set $k_2 = k - k_1$;
   - ▷ let S_1 be the set of k_1 largest weighted degree vertices in V_1, and S_2' be the set of k_2 largest weighted degree vertices in V_2 if S_1 is removed from V_1; set $\mathrm{SOL}_1 = S_1 \cup S_2'$;

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

▷ let S_2 be the set of k_2 largest weighted degree vertices in V_2, and S'_1 be the set of k_1 largest weighted degree vertices in V_1 if S_2 is removed from V_2; set $\mathrm{SOL}_2 = S_2 \cup S'_1$;

2. output $\mathrm{sol}(G)$ the best among the solutions SOL_1 and SOL_2 computed above.

Theorem 1. *The above algorithm is a $2/3$-approximation algorithm for* MAX k-VERTEX COVER *in bipartite graphs.*

Proof. Considering an optimal solution O (i.e., a set of k vertices covering a maximum number of edges in E), let O_1 and O_2 be the parts of O lying in color-classes V_1 and V_2, respectively. Consider the iteration of the algorithm where $k_1 = |O_1|$ (hence $k_2 = |O_2|$). In SOL_1 S'_2 is the best choice of choosing k_2 vertices from V_2 once S_1 is chosen. Consequently, SOL_1 is at least as good as $S_1 \cup O_2$, and at least as good as $S_1 \cup S_2$. From the first argument we get:

$$\mathrm{sol}(G) \geqslant \delta(S_1) + \delta(O_2) - \delta(S_1, O_2) \tag{54}$$

and the same argument for SOL_2 gives:

$$\mathrm{sol}(G) \geqslant \delta(S_2) + \delta(O_1) - \delta(S_2, O_1). \tag{55}$$

Denote by x the number of edges between $S_1 \cap O_1$ and $S_2 \cap O_2$. Then $S_1 \cup S_2$ covers at least $\delta(S_1, O_2) + \delta(S_2, O_1) - x$ edges, so

$$\mathrm{sol}(G) \geqslant \delta(S_1, O_2) + \delta(S_2, O_1) - x. \tag{56}$$

Considering that $\delta(S_1) \geqslant \delta(O_1)$ and $\delta(S_2) \geqslant \delta(O_2)$, summing up inequalities (54), (55) and (56) gives:

$$3\mathrm{sol}(G) \geqslant 2\delta(O_1) + 2\delta(O_2) - x.$$

Since $x \leqslant \delta(O_1, O_2)$ we get the result. □

We now show that in the case where $O_1 \cap S_1 = \emptyset$ and $O_2 \cap S_2 = \emptyset$ at iteration $k_1 = |O_1|$ (i.e., when $\nu = \xi = 0$; notice that this case is not polynomially detectable), a $4/5$-approximation ratio is achieved by the algorithm above.

As said before, the output solution is at least as good as $S_1 \cup S_2$. Since S_2 and O_2 are assumed to be disjoint, we get:

$$\mathrm{sol}(G) \geqslant \delta(S_2) + \delta(S_1, O_2). \tag{57}$$

Now, consider the solution when the algorithm takes the set S_1 of k best vertices in V_1. Since $k_1 \leqslant k/2$ and O_1 and S_1 are disjoint, it holds in this case that:

$$\mathrm{sol}(G) \geqslant \delta(S_1) + \delta(O_1) \tag{58}$$

Now, sum up (54), (57) and (58) with coefficients 2, 2 and 1, respectively. Then:

$$5 \cdot \mathrm{sol}(G) \geqslant 3 \cdot \delta(S_1) + \delta(O_1) + 2 \cdot \delta(S_2) + 2 \cdot \delta(O_2).$$

Note that $\mathrm{opt}(G) \leqslant \delta(O_1) + \delta(O_2)$. The results follow since by the choice of S_1 and S_2 we have $\delta(S_1) \geqslant \delta(O_1)$ and $\delta(S_2) \geqslant \delta(O_2)$.

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

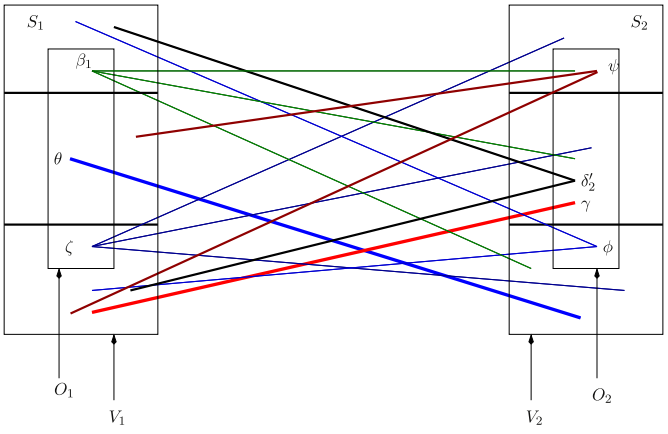

Fig. 7. The edge-sets induced by the several "Greek" parameters.

6.2. A 34/47-ratio combinatorial algorithm

In this section, we will use the following additional notations and conventions with respect to those of Section 2 (we assume that vertices in both V_1 and V_2 are ordered in decreasing weighted degree order), where all the Greek letters (but δ) used, imply parameters that are all smaller than, or equal to, 1: then previous sections):

- for conciseness we set $\delta(O_1) = \alpha \cdot \text{opt}$;
- $\beta_1 \cdot \delta(O_1) = \beta_1 \cdot \alpha \cdot \text{opt}$: the total weight of edges covered by $S_1 \cap O_1$;
- $\delta'(O_2)$: the total weight of *private* edges covered by O_2, i.e., the edges already covered by O_1 are not counted up to $\delta'(O_2)$; obviously, $\delta'(O_2) = (1-\alpha) \cdot \text{opt}$;
- $\psi \cdot \delta'(O_2) = \psi \cdot (1-\alpha) \cdot \text{opt}$: the total weight of *private* edges covered by $S_2 \cap O_2$;
- $\theta \cdot \delta(O_1)$: the total weight of edges (if any) from O_1 that go "below" O_2 (recall V_1 and V_2 are ordered in decreasing weighted degree order);
- $\gamma \cdot \delta'(O_2)$: symmetrically, it denotes the total weight of edges of O_2 that go below the vertices of O_1;
- $\zeta \cdot \delta(O_1)$: suppose that after taking the k best vertices of V_1, there still remain, say, k'_1 vertices of O_1 that have not been encountered yet; then, $\zeta \cdot \delta(O_1)$ is the total weight of edges covered by those vertices;
- $\phi \cdot \delta'(O_2)$: this is the symmetric of the quantity $\zeta \cdot \delta(O_1)$ for the pair (V_2, O_2) (supposing that the number of vertices in O_2 that have not been encountered is k'_2).

In Fig. 7, the edge-sets defined by the parameters above are illustrated (for simplicity we use in the figure the same notation for a cut and its weight). Heavy lines within rectangles V_1 and V_2 represent the borders of S_1 and S_2 (the upper ones) and those of the k best vertices (the lower ones). Edges from O_1 are not shown in the figure. They can go everywhere in V_2. Private edges of O_2 are shown as heavy black and red lines (the sets of edges with total weights δ'_2 and ψ, respectively). They can go everywhere in $V_1 \setminus O_1$.

The algorithm. The basic idea of the algorithm, that is a simplified version of the algorithm analyzed in Section 3, is the following.

Consider an optimal solution O (i.e., a vertex-set on k vertices covering a maximum number of edges in E). Denoting as previously by O_1 and O_2 the subsets of an optimal solution O lying in the color-classes V_1 and V_2 and supposing that $|O_1| = k_1$ and $|O_2| = k_2$ ($k_1 + k_2 = k$), for any pair (χ_1, χ_2) such that $\chi_1 + \chi_2 = k$, it builds the five solutions specified just below and returns the best among them. As previously we only specify its stem where $(\chi_1, \chi_2) = (k_1, k_2)$. This step goes as follows:

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

build the following solutions:

SOL$_1$: S_1 plus the k_2 remaining best vertices from V_2;
SOL$_2$: S_2 plus the k_1 remaining best vertices from V_1;
SOL$_3$: the k best vertices of V_1;
SOL$_4$: the k best vertices of V_2;
SOL$_5$: the best $2 \cdot k_1$ vertices of V_1 plus the remaining $k - 2 \cdot k_1$ best vertices of V_2;

return the best among the five solutions computed.

It is easy to see that the complexity of the overall algorithm is $O(n^3)$.

Lemmata 2–5 give expressions for the ratios achieved by the five solutions built by the algorithm. These expressions are functions of some of the parameters specified just above. Lemma 2 handles the approximation ratios achieved by solutions SOL$_1$, SOL$_2$ and SOL$_4$ specified just above.

Lemma 2. *The approximation ratios achieved by the solutions* SOL$_1$, SOL$_2$ *and* SOL$_4$ *are, respectively, at least:*

$$1 - \alpha + \beta_1 \cdot \alpha \tag{59}$$
$$\alpha + \psi \cdot (1 - \alpha). \tag{60}$$
$$(1 - \alpha)(1 + \mu \cdot (1 - \psi)). \tag{61}$$

Furthermore, if either S_1 and O_1, or S_2 and O_2 coincide (i.e., $S_1 \cap O_1 = S_1$, or $S_2 \cap O_2 = S_2$), then either SOL$_1$, *or* SOL$_2$ *is optimal.*

Proof. For (59), S_1 contributes, by the first line (11), with a total edge-weight more than $\delta(O_1) = \alpha \cdot \text{opt}$. Decompose this weight into a weight X for the edges covered by $S_1 \setminus (S_1 \cap O_1)$ and a weight $\beta_1 \cdot \alpha \cdot \text{opt}$ concerning the edges covered by $S_1 \cap O_1$, i.e., $\delta(S_1) = |X| + \beta_1 \cdot \alpha \cdot \text{opt}$. On the other hand, the k_2 best vertices in V_2 have a contribution at least equal to that of the k_2 vertices in O_2, that ensure a weight more than $(1 - \alpha) \cdot \text{opt} - X$, qed. The proof for (60) is similar.

If S_1 and O_1 coincide, SOL$_1$ will cover $\alpha \cdot \text{opt} + (1 - \alpha) \cdot \text{opt} = \text{opt}$ edges. The same holds when S_2 and O_2 coincide.

We now prove (61). In fact, SOL$_4$ consists of first taking S_2 and of completing it with the k_1 best vertices below this set. The contribution of S_2 in this solution is, by the second line of (11), more than $\delta(O_2) = (1 - \alpha) \cdot \text{opt}$. But, there still exists an uncovered weight of $(1 - \alpha) \cdot (1 - \psi) \cdot \text{opt}$ due to the edges covered by the optimum; these edges are covered by less than k_2 vertices lying below S_2 and SOL$_4$ takes the k_1 best vertices below S_2. A simple average argument immediately concludes that these vertices contribute with a weight that is at least $({}^{k_1}/_{k_2}) \cdot (1 - \alpha) \cdot (1 - \psi) \cdot \text{opt} = \mu \cdot (1 - \psi) \cdot (1 - \alpha) \cdot \text{opt}$. □

Before studying the ratios of SOL$_3$ and SOL$_5$, let us note that we assume k'_1 vertices of O_1, ensuring a total weight of $\zeta \cdot \delta(O_1)$ that remain outside (below) the k best vertices of V_1. At the end of the proof of Theorem 2, the case $k'_1 = \zeta = 0$ will be separately handled.

Consider solution SOL$_3$. The k vertices taken from V_1 can be seen as the union of $\lfloor {}^{k}/_{k_1} \rfloor$ consecutive k_1-groups (called clusters in what follows), eventually completed by some vertices just below the last cluster. By the fact that $k_1 = \mu \cdot k_2$ and (8),

$${}^{k}/_{k_1} = {}^{(1+\mu)}/_{\mu} \Rightarrow \lfloor {}^{k}/_{k_1} \rfloor \geqslant ({}^{(1+\mu)}/_{\mu}) - 1 = {}^{1}/_{\mu}.$$

Assume now that the $k - k'_1$ vertices of O_1 encountered among the k best vertices of V_1 are included in the π first clusters. Denote by κ_i the number of vertices of O_1 in the ith cluster, $i = 1, \ldots, \pi$, and suppose that the "optimal" κ_i vertices of cluster i cover $\beta_i \cdot \delta(O_1) = \beta_i \cdot \alpha \cdot \text{opt}$ edges. Note also that $\pi \leqslant \lfloor {}^{k}/_{k_1} \rfloor$.

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

Lemma 3. *Consider cluster $i \in [1,\pi]$ and denote by $\bar{O}_{1,i}$ the part of O_1 not captured by clusters $1,2,\ldots,i-1$ (so, $|\bar{O}_{1,1}| = |O_1| = \sum_{j=1}^{\pi}\kappa_j + k'_1$ and $|\bar{O}_{1,i}| = \sum_{j=i}^{\pi}\kappa_j + k'_1$). Then, the vertices of cluster i will cover edges of a total weight that is at least $(1 - \sum_{j=0}^{i-1}\beta_j)\cdot\alpha\cdot\mathrm{opt}$ ($\beta_0 = 0$). Moreover, any of the remaining $\lfloor k/k_1\rfloor - \pi$ clusters in SOL_3 will cover edges of a total weight that is more than $\zeta\cdot\alpha\cdot\mathrm{opt}$.*

Proof. Observe that $\delta(O_1) = \sum_{i=1}^{\pi}\beta_i\cdot\alpha\cdot\mathrm{opt} + \zeta\cdot\alpha\cdot\mathrm{opt}$ and that the part of $\delta(O_1)$ covered by $\bar{O}_{1,i}$ is:

$$\delta\left(\bar{O}_{1,i}\right) = \delta\left(O_1\right) - \sum_{j=0}^{i-1}\beta_j\cdot\delta\left(O_1\right) = \left(1 - \sum_{j=0}^{i-1}\beta_j\right)\cdot\alpha\cdot\mathrm{opt}$$

$$= \left(\sum_{j=i}^{\pi}\beta_j + \zeta\right)\cdot\alpha\cdot\mathrm{opt}. \tag{62}$$

The edges whose total weight is $\delta(\bar{O}_{1,i})$ are covered by $\sum_{j=i}^{\pi}\kappa_j + k'_1 = k_1 - (\sum_{j\leqslant i-1}\kappa_j) \leqslant k_1$ vertices ($\kappa_0 = 0$), while cluster i contains exactly k_1 vertices, κ_i of them belonging to O_1 while the $k_1 - \kappa_i$ remaining ones having weighted degrees at least as large as those of the vertices in $\bar{O}_{1,i+1}$. An easy average argument derives then that the total weight of edges covered by the vertices of cluster i will be least:

$$k_1\cdot\frac{\left(1 - \sum_{j=0}^{i-1}\beta_j\right)\cdot\alpha\cdot\mathrm{opt}}{k_1 - \left(\sum_{j\leqslant i-1}\kappa_j\right)} \geqslant \left(1 - \sum_{j=0}^{i-1}\beta_j\right)\cdot\alpha\cdot\mathrm{opt}. \tag{63}$$

Using the same average argument, vertices in any of the remaining $\lfloor k/k_1\rfloor - \pi$ clusters ensure a weight that is more than $k_1\cdot\zeta\cdot\delta(O_1)/k'_1 \geqslant \zeta\cdot\delta(O_1) \geqslant \zeta\cdot\alpha\cdot\mathrm{opt}$. □

Lemma 4. *The approximation ratio achieved by SOL_3 is at least:*

$$\left(\sum_{i=0}^{\pi-1}\left(1 - \sum_{j=0}^{i}\beta_j\right)\right)\cdot\alpha. \tag{64}$$

Proof. By (62) and (63), the k_1 vertices of cluster i, $i\in[1,\pi]$, will contribute with total weight more than:

$$\frac{k_1}{k_1 - \sum_{j\leqslant i-1}\kappa_j}\cdot\left(\sum_{j=i}^{\pi}\beta_j + \zeta\right)\cdot\delta\left(O_1\right) \geqslant \left(1 - \sum_{j=0}^{i-1}\beta_j\right)\cdot\delta\left(O_1\right) \tag{65}$$

while, by Lemma 3, the rest of the vertices in the $\lfloor k/k_1\rfloor - \pi$ remaining clusters will contribute with a total weight more than $(\lfloor k/k_1\rfloor - \pi)\cdot\zeta\cdot\delta(O_1)$. Summing (65) for $i = 0$ to $\pi - 1$ and setting $\beta_0 = 0$, we get:

$$\mathrm{sol}_3(G) \geqslant \left(\sum_{i=0}^{\pi-1}\left(1 - \sum_{j=0}^{i}\beta_j\right) + \left(\left\lfloor\frac{k}{k_1}\right\rfloor - \pi\right)\cdot\zeta\right)\cdot\delta\left(O_1\right)$$

This quantity is always greater than its first term since $\lfloor k/k_1\rfloor - \pi$ is positive. □

The following (last) lemma handles approximation ratio of solution SOL_5.

Lemma 5. *The approximation ratio achieved by solution SOL_5 is at least:*

$$(1-\mu) - (1 - 3\cdot\mu)\cdot\alpha + (1-\mu)\cdot(\beta_1 + \beta_2)\cdot\alpha - \mu\cdot\beta_1\cdot\alpha. \tag{66}$$

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

Proof. Rewriting Lemma 3 for $\pi = 2$, the $2 \cdot k_1$ best vertices from V_1 ensure a total edge-weight at least $(\beta_1 + \beta_2) \cdot \delta(O_1)$ of the optimum and more than $[(1 - \beta_1) + (1 - \beta_1 - \beta_2)] \cdot \delta(O_1) = (2 - 2 \cdot \beta_1 - \beta_2) \cdot \alpha \cdot \text{opt}$ additional weight, i.e., a total of at least:

$$[(\beta_1 + \beta_2) + (2 - 2 \cdot \beta_1 - \beta_2)] \cdot \alpha \cdot \text{opt}. \tag{67}$$

The best $k - 2 \cdot k_1$ vertices of V_2 will ensure then a weight that is at least equal to the contribution of the $k - 2 \cdot k_1$ best vertices of O_2, that ensure an additional coverage weight of at least:

$$\begin{aligned} & \frac{k - 2 \cdot k_1}{k_2} \cdot [\delta'(O_2) - (2 - 2 \cdot \beta_1 - \beta_2) \cdot \delta(O_1)] \\ = \; & (1 - \mu) \cdot [(1 - \alpha) - (2 - 2 \cdot \beta_1 - \beta_2) \cdot \alpha] \cdot \text{opt}. \end{aligned} \tag{68}$$

Summing (67) and (68), we get (66). □

We are ready to prove the final result of this subsection:

Theorem 2. *Weighted* MAX k- VERTEX COVER *in bipartite graphs is combinatorially* $34/47$*- approximable in polynomial time.*

Proof. We propose an exhaustive parameter-elimination that has the advantage to be quite simple. It consists of subsequently eliminating parameters from the ratios proved in Lemmata 2, 4 and 5, until ratios that are only functions of μ are obtained.

Expressions (60) and (61) have opposite monotonies with respect to ψ, so equalizing them one can eliminate this parameter. Some easy algebra leads to:

$$\alpha + \psi \cdot (1 - \alpha) = (1 - \alpha)(1 + \mu \cdot (1 - \psi)) \Rightarrow \psi = \frac{1 - \alpha \cdot \frac{2+\mu}{1+\mu}}{1 - \alpha} \tag{69}$$

and embedding (69) to either (60) or (61) results in ratio:

$$1 - \frac{\alpha}{1 + \mu}. \tag{70}$$

In what follows, we distinguish two cases following the value of parameter μ, namely, $\mu \leqslant 1/2$ and $\mu > 1/2$. For the former we show that the ratio achieved by the algorithm is $3/4$, while for the latter we obtain a lower bound of $34/47$.

Case 1: $\mu \leqslant 1/2$. In this case, the number of clusters needed in order to capture the k best vertices of V_1 is at least 3, and (64) in Lemma 4 leads to the ratio:

$$(3 - 2 \cdot \beta_1 - \beta_2) \cdot \alpha. \tag{71}$$

Ratios (66) and (71) have opposite monotonies with respect to the term $(\beta_1 + \beta_2)$; so, equalizing them leads after some easy algebra to:

$$(\beta_1 + \beta_2) = \frac{-(1 - \mu) + (4 - 3 \cdot \mu) \cdot \alpha - (1 - \mu) \cdot \beta_1 \cdot \alpha}{(2 - \mu) \cdot \alpha}. \tag{72}$$

Embedding (72) in either one of (66) and (71), one gets:

$$\frac{(1 - \mu) + 2 \cdot \alpha - \beta_1 \cdot \alpha}{2 - \mu} \tag{73}$$

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

which, equalized with ratio in (59) in order to eliminate parameter β_1, leads to:

$$\beta_1 = \frac{-1 + (4-\mu)\cdot\alpha}{(3-\mu)\cdot\alpha}$$

and then to ratio:

$$\frac{2-\mu+\alpha}{3-\mu}. \tag{74}$$

Now, ratios (70) and (74) have opposite monotonies with respect to α and equality of them occurs when $\alpha = {}^{(1+\mu)}/_4$ that leads to a ratio equal to ${}^3/_4$.

Case 2: $\mu > {}^1/_2$. Here, the k best vertices of V_1 will be contained in at most three clusters. If the number of clusters needed to capture those vertices is equal to three, the analysis of the previous case remains valid and the ratio achieved is again ${}^3/_4$. Hence, suppose that the k best vertices of V_1 have been captured by the first two clusters (with eventually a certain number of them –less than k_1 –below the second cluster; denote by X this set of vertices). Then, by Lemma 3, the $2\cdot k_1$ vertices of the two first (full) clusters will contribute with a total weight that is at least $(2-\beta_1)\cdot\delta(O_1)$, while the contribution of set X is at least $(1-\beta_1-\beta_2-\zeta)\cdot\delta(O_1)$. In all, it holds in this case that:

$$\mathrm{sol}_3(G) \geqslant (3 - 2\cdot\beta_1 - \beta_2 - \zeta)\cdot\delta(O_1)$$

that induces an approximation ratio:

$$(3 - 2\cdot\beta_1 - \beta_2 - \zeta)\cdot\alpha. \tag{75}$$

On the other hand, by the discussion at the end of the proof of Lemma 3, any of the two first clusters covers a total weight more than $\zeta\cdot\delta(O_1)$; hence SOL_3 also guarantees ratio $2\cdot\zeta\cdot\alpha$. Combination of this ratio with (75) and elimination of ζ, leads to ratio:

$$\frac{2}{3}\cdot(3 - 2\cdot\beta_1 - \beta_2)\cdot\alpha. \tag{76}$$

We now proceed exactly as in the previous case. We first equalize (66) with (76), to eliminate the term $(\beta_1+\beta_2)$; this, after some easy though tedious algebra leads to:

$$(\beta_1+\beta_2) = \frac{-3\cdot(1-\mu) + 9\cdot(1-\mu)\cdot\alpha - (2-3\cdot\mu)\cdot\beta_1\cdot\alpha}{(5-3\cdot\mu)\cdot\alpha}$$

which embedded to either one of (66), or (76), leads to ratio:

$$\frac{\left(2-5\cdot\mu+3\cdot\mu^2\right) + \left(4+9\cdot\mu-9\cdot\mu^2\right)\cdot\alpha - \left(2+2\cdot\mu-3\cdot\mu^2\right)\cdot\beta_1\cdot\alpha}{5-3\cdot\mu}. \tag{77}$$

Equality of ratios (59) and (77) (in order to eliminate β_1) gives:

$$\beta_1 = \frac{-\left(3+2\cdot\mu-3\cdot\mu^2\right) + \left(9+6\cdot\mu-9\cdot\mu^2\right)\cdot\alpha}{(7-\mu-3\cdot\mu^2)\cdot\alpha}$$

and leads to ratio:

$$\frac{4-3\cdot\mu+\left(2+7\cdot\mu-6\cdot\mu^2\right)\cdot\alpha}{7-\mu-3\cdot\mu^2}. \tag{78}$$

Equality of ratios (78) and (70) occurs when:

$$\alpha = \frac{(1+\mu)\cdot\left(3+2\cdot\mu-3\cdot\mu^2\right)}{9+8\cdot\mu-2\cdot\mu^2-6\cdot\mu^3}.$$

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

Embedding now this value for α to, say, (70), derives ratio:

$$\frac{6+6\cdot\mu+\mu^2-6\cdot\mu^3}{9+8\cdot\mu-2\cdot\mu^2-6\cdot\mu^3}. \tag{79}$$

Ratio in (79) increases with μ and for $\mu = 1/2$ becomes greater than $34/47 \cong 0.7234$, that concludes the study of case $\mu > 1/2$.

To conclude the proof, it is easy to see that, when $\zeta = 0$, the analysis of the case $\mu \leqslant 1/2$ remains valid. So, for $\zeta = 0$ the ratio achieved by the algorithm is $3/4$. □

Acknowledgments

The very pertinent comments and suggestions of two anonymous reviewers are gratefully acknowledged.

A first version of the paper have been achieved while the first author was with the Institute for Computer Science and Control, at the Hungarian Academy of Sciences (MTA SZTAKI).

The last author acknowledges the support of the Swiss National Research Foundation Early Post-Doc mobility grant P1TIP2_152282 while the author was member of LAMSADE, Paris-Dauphine University, and a NWO TOP2 grant.

Appendix. Description of the GRG method

Here, we will describe in a little more detail the GRG method with the sole purpose of making the paper a bit more self-contained. Of course, the interesting reader is pointed to the original publications or to the standard textbook presentations mentioned in the corresponding section.

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 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. The general form of the method is the following:

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

where $\boldsymbol{x}$ is the n-dimensional variable vector, h_i is the ith 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 α and β be the corresponding vectors) such that:

1. α has dimension m and β 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, $\forall x_i \in \alpha$, $\boldsymbol{L}_{x_i} \leqslant x_i \leqslant \boldsymbol{U}_{x_i}$.
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 $h(\alpha, \beta) = 0$. This immediately implies that $\mathrm{d}\alpha/\mathrm{d}\beta = (\nabla_\alpha h(\boldsymbol{x}))^{-1}\nabla_\beta h(\boldsymbol{x})$.

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) β to be the reduced gradient as follows:

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

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

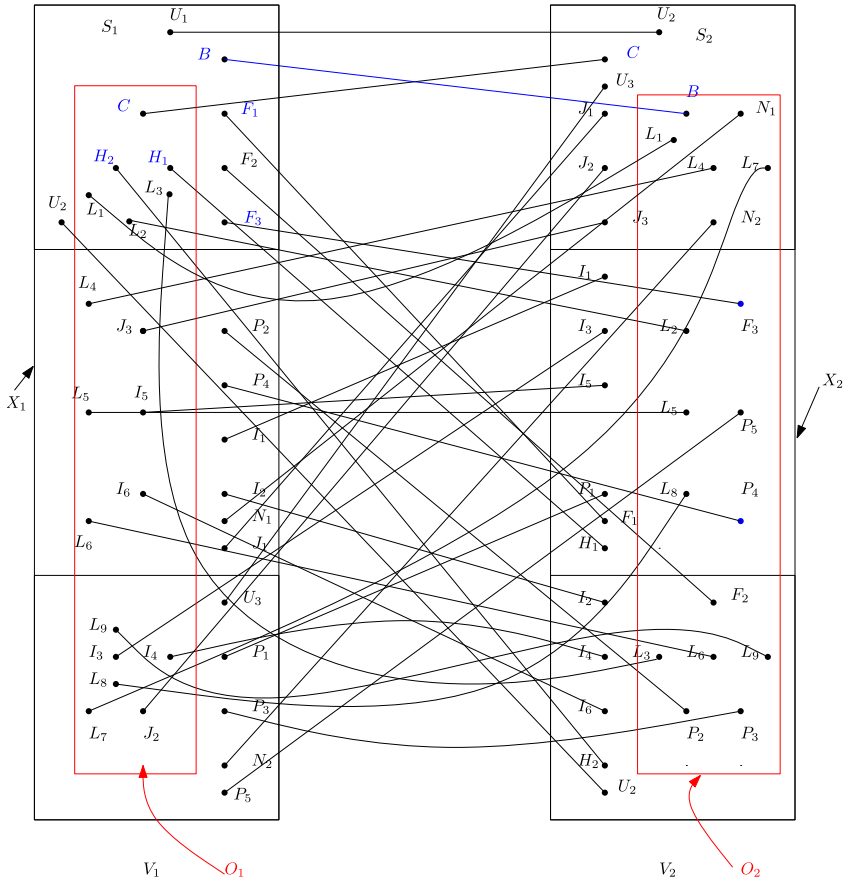

Fig. 8. Sets S_i, O_i, X_i $i = 1, 2$ and cuts between them.

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 [26] of optimality of non-linear systems.

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

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

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

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

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.

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.

References

[1] N. Apollonio, B. Simeone, The maximum vertex coverage problem on bipartite graphs, Discrete Appl. Math. 165 (2014) 37–48.
[2] B. Caskurlu, V. Mkrtchyan, O. Parekh, K. Subramani, On partial vertex cover and budgeted maximum coverage problems in bipartite graphs, in: Theoretical Computer Science - 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Rome, Italy, September 1–3, 2014. Proceedings, 2014, pp. 13–26.
[3] A. Badanidiyuru, R. Kleinberg, H. Lee, Approximating low-dimensional coverage problems, in: Tamal K. Dey, Sue Whitesides (Eds.), Symposuim on Computational Geometry 2012, SoCG '12, Chapel Hill, NC, USA, June 17-20, 2012, ACM, 2012, pp. 161–170.
[4] G. Cornuejols, M.L. Fisher, G.L. Nemhauser, Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms, Manage. Sci. 23 (8) (1977) 789–810.
[5] D.S. Hochbaum, A. Pathria, Analysis of the greedy approach in problems of maximum k-coverage, Naval Res. Logist. 45 (1998) 615–627.
[6] Q. Han, Y. Ye, H. Zhang, J. Zhang, On approximation of max-vertex-cover, European J. Oper. Res. 143 (2002) 342–355.
[7] A.A. Ageev, M. Sviridenko, Approximation algorithms for maximum coverage and max cut with given sizes of parts, in: G. Cornuéjols, R.E. Burkard, G.J. Woeginger (Eds.), Proc. Conference on Integer Programming and Combinatorial Optimization, IPCO'99, in: Lecture Notes in Computer Science, vol. 1610, Springer-Verlag, 1999, pp. 17–30.
[8] E. Petrank, The hardness of approximation: gap location, Comput. Complexity 4 (1994) 133–157.
[9] N. Apollonio, B. Simeone, Improved approximation of maximum vertex coverage problem on bipartite graphs, SIAM J. Discrete Math. 28 (3) (2014) 1137–1151.
[10] N. Apollonio, A. Sebö, Minconvex factors of prescribed size in graphs, SIAM J. Discrete Math. 23 (3) (2009) 1297–1310.
[11] L. Trevisan, Max cut and the smallest eigenvalue, in: M. Mitzenmacher (Ed.), Proc. STOC'09, 2009, pp. 263–272.
[12] M.X. Goemans, D.P. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, J. Assoc. Comput. Mach. 42 (6) (1995) 1115–1145.
[13] A. Gupta, A. Kumar, Greedy algorithms for steiner forest, in: R.A. Servedio and R. Rubinfeld (Ed.), Proc. STOC'15, 2015, pp. 871–878.
[14] Y.H. Chan, L. Chi Lau, On linear and semidefinite programming relaxations for hypergraph matching, Math. Program. 135 (1–2) (2012) 123–148.
[15] B. Barak, Sum of Squares Upper Bounds, Lower Bounds, and Open Questions, in: Lecture Notes, Princeton University, 2014.
[16] G. Robins, A. Zelikovsky, Tighter bounds for graph steiner tree approximation, SIAM J. Discrete Math. 19 (1) (2005) 122–134.
[17] J. Byrka, F. Grandoni, T. Rothvoß, L. Sanità, Steiner tree approximation via iterative randomized rounding, J. Assoc. Comput. Mach. 60 (1) (2013) 6.
[18] U. Feige, M. Karpinski, M. Langberg, Improved approximation of max-cut on graphs of bounded degree, J. Algorithms 43 (2) (2002) 201–219.
[19] P. Austrin, S. Benabbas, K. Georgiou, Better balance by being biased: a 0.8776-approximation for max bisection. in: S. Khanna (Ed.), Proc. Symposium on Discrete Algorithms, SODA'13, 2013, pp. 277–294.
[20] J. Abadie, J. Carpentier, Generalization of the Wolfe reduced gradient method to the case of non-linear constraints, in: J. Abadie, J. Carpentier (Eds.), Optimization, Academic Press, 1969.
[21] M. Frank, P. Wolfe, An algorithm for quadratic programming, Naval Res. Logist. Q. 3 (1–2) (1956) 95–110.
[22] J. Abadie, J. Carpentier, Généralisation de la méthode du gradient réduit de wolfe au cas de contraintes nonlinéaires, in: Proc. IFORS Congress, 1966.
[23] M.S. Bazaraa, H.D. Sherali, C.M., Nonlinear Programming : Theory and Algorithms, third ed., in: Wiley-Interscience series in discrete mathematics and optimization, John Wiley & Sons, New York (NY), Chichester, Brisbane, 2006.
[24] R.E. Mickens, Difference Equations: Theory and Applications, Chapman and Hall/CRC, 1991.
[25] H. Michiel (Ed.), Finite-Difference Calculus, in: Encyclopedia of Mathematics, Springer, 2001.
[26] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, New York, 2004.

Please cite this article in press as: É. Bonnet, et al., Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs, Discrete Optimization (2017), http://dx.doi.org/10.1016/j.disopt.2017.09.001.