Editorial Manager(tm) for Annals of Operations Research
Manuscript Draft

Manuscript Number: ANOR-122R1

Title: Edge cover by connected bipartite subgraphs

Article Type: Special: In memory of Peter L. Hammer

Keywords: edge cover; connected bipartite subgraph; min cut cover; complexity; constructive heuristic; local search

Corresponding Author: Dr Leo Liberti, PhD

Corresponding Author's Institution: Ecole Polytechnique

First Author: Leo Liberti, PhD

Order of Authors: Leo Liberti, PhD; Laurent Alfandari, PhD; Marie-Christine Plateau, PhD

Abstract: We consider the problem of covering the edge set of an unweighted, undirected graph with the minimum number of connected bipartite subgraphs (where the subgraphs are not necessarily bicliques). We show that this is an NP-hard problem, provide lower bounds through an integer programming formulation, propose several constructive heuristics and a local search, and discuss computational results. Finally, we consider a constrained variant of the problem which we show to be NP-hard, and provide an integer programming formulation for the variant.

Answers to reviewers

Reviewer 1

Major points

1. *Lack of motivation*. We think it is fitting to report these results on an issue devoted to the memory of Peter Hammer, as he first proposed the problem. We believe we surveyed most of the literature on bipartite graphs and their applications, without really finding an explicit reference to a _native_ application of this problem. Notwithstanding, since the problem is equivalent to the MCC and to the Minimum Test Collection Problem, it "inherits" all the applications of the these two problems, as we underlined in a paragraph in the introduction.

2. *Correctness of proof of thm. 5.1*. We think the referee considered the "length of a graph E_k" to be defined as the maximum length of a shortest path in E_k. In fact, we define the length of E_k to be the maximum length of a *simple* path in E_k (not necessarily shortest). We made this definition explicit just before the definition of the MCBGC problem. So, for L=2 the bipartite subgraph is indeed a star (hence the problem is equivalent to the vertex cover problem) and not a complete bipartite graph.

Minor points

1. The numbering of results should be taken care of during production.
2. OK
3. OK
4. OK
5. OK
6. OK
7. OK
8. Since x_{ik} is a decision variable, its value after the

optimization process will encode the definition of V_k (given in the Cut Cover problem definition, page 3).

9. OK ("subgraph of the cover").
10. OK
11. OK
12. OK
13. We explained this in the 2nd paragraph of Sect. 4.3
14. OK
15. We agree unless L is a constant (i.e. not depending on n --- in which case all simple paths can be enumerated --- at worst). We inserted a remark to this effect after Thm. 5.1
16. OK
17. We removed the first sentence of 5.1 because we had already introduced an explicit definition of graph length at the beginning of section 5. We remark that we took out the notion of "induced path" from the paper, replacing by "path contained in a subgraph".
18. OK
19. OK

Reviewer 2

Major points

1. OK
2. We removed \bar{A},\bar{B}, A_1,B_1,A_2,B_2 from the Lemma statement
3. OK - we added the figure
4. OK - But as far as we know the MCC math prog formulation was never introduced previously -- although naturally it's nothing specially difficult so it might have already appeared in some book exercise or similar.
5. We actually eliminated \tilde{\alpha} entirely and defined \alpha straight away
6. We reinforced the explanation of the CTW07 heuristic somewhat and added

an URL where the paper that describes it in detail can be downloaded from.

7. We noticed that, combined with the "budge the edge" local search, heuristic L is good on Torus instances whereas K behaves well on Triangle, but the number of instances in each class (3 or 4) and the size of the instances seems to small to us for stating general and explainable conclusions on this aspect.
8. We inserted a figure explaining the transformation. Thank you for pointing out this mistake: it is the length of the path connecting v^{ij}_{jp} and v^{jl}_{jp} which is equal to 2p. We modified the whole sentence, given that it is not E_k that cannot contains more than 2 edges of $E"$, as the referee pointed out, but any path of E_k. We also redefined X more precisely as X={j_k|k in K'}.

Minor points

1. The reference order is carried out automatically by LaTeX. They will likely be fixed in production according to journal style
2. OK - We inserted some lines in the introduction to this effect.
3. We think our definition - a bipartite graph not required to be connected but spanning V - is OK.
4. OK
5. OK
6. OK
7. OK
8. OK
9. OK
10. OK
11. OK
12. OK
13. OK
14. OK
15. OK
16. OK
17. OK
18. We remarked that we're using v^{ij}_{ip} to mean v^{\{i,j\}}_{ip}

with a slight abuse of notation.

19. Not true in general: think of the case where j is a leaf in graph G
20. OK
21. OK
22. Since V^2 = V x V, and arcs are elements of a subset of V^2, it is formally correct to say (u,v) in V^2 whether (u,v) is an arc in a graph or not.

Edge cover by connected bipartite subgraphs

Leo Liberti[1], Laurent Alfandari[2], Marie-Christine Plateau[3]

[1] *LIX, École Polytechnique, F-91128 Palaiseau, France*
Email:liberti@lix.polytechnique.fr

[2] *ESSEC, Av. Bernard Hirsch, BP105 95021, Cergy Pontoise, France*
Email:alfandari@essec.fr

[3] *CEDRIC, CNAM, 292 rue St. Martin, 75141 Paris, France*
Email:mcplateau@yahoo.fr

November 21, 2008

Abstract

We consider the problem of covering the edge set of an unweighted, undirected graph with the minimum number of connected bipartite subgraphs (where the subgraphs are not necessarily bicliques). We show that this is an **NP**-hard problem, provide lower bounds through an integer programming formulation, propose several constructive heuristics and a local search, and discuss computational results. Finally, we consider a constrained variant of the problem which we show to be **NP**-hard, and provide an integer programming formulation for the variant.

1 Introduction

We consider the following problem.

> Minimum Bipartite Graph Cover (MBGC). Given a simple undirected graph $G = (V, E)$, find a family $\{H_k = (A_k, B_k, E_k) \mid k \leq m\}$ of (not necessarily induced nor complete) connected bipartite subgraphs of G such that $E = \bigcup_{k \leq m} E_k$ and m is minimum.

This problem was proposed by P. Hammer to one of the authors during a discussion on reformulations of pseudo-boolean functions which occurred in Reykjavik during the EURO 2006 conference. The particular subject under discussion was the struction operation on conflict graphs [2, 13] as an alternative to pseudo-boolean reformulations. The question of the significance of the posiform that would occur from considering a bipartite cover instead of a biclique cover arose naturally, for an optimal bipartite cover has fewer subgraphs than an optimal biclique cover, and would therefore lead to simpler posiforms. The focus of the discussion then shifted to the problem of finding a bipartite cover as a necessary step for devising computational experiments. We have no progress yet to report on the main question, i.e. the significance of the modified posiform; this paper presents the progress on the auxiliary topic of covering the edges of a graph by a minimum cardinality set of connected not necessarily induced nor complete bipartite subgraphs.

A related problem is the edge covering where H_k are necessarily induced bicliques (Minimum Biclique Cover (MBC)), which is relevant to the struction operation [2, 13]. The MBC is **NP**-hard and well studied [21, 19, 3, 1], and has many applications (for example the encoding of partial orders by bit vectors [11], heuristic coloring algorithms [9] and continuous relaxation based methods [7]). Another related problem is the Minimum Cut Cover (MCC), where H_k are cutsets, namely not required to be connected but required to have V as vertex set. The MCC is also **NP**-hard [15, 6, 18].

In this paper we show the **NP**-completeness of the decision version of the MBGC by reduction from the MCC; in particular, we also prove the complete equivalence of these two problems with respect to the minimization of cover cardinality (Sect. 2). This implies that the same approximability results found for the MCC hold for the MBGC [18, 12]. Since the direct formulation of the MBGC results in a model whose size makes practical application extremely limited, we provide a Mixed-Integer Linear Programming (MILP) formulation for the MCC instead, which also yields optimal solutions for the MBGC (Sect. 3). We then describe four constructive heuristics and a local search for the MBGC (Sect. 4) and discuss some computational results. Finally, we introduce a constrained variant of the MBGC where the subgraphs in the cover are of limited length, prove that its decision version is also **NP**-complete, and provide a MILP formulation for the variant (Sect. 5).

The complete equivalence of the MBGC with the MCC means that the MBGC is also equivalent to all problems equivalent to the MCC. In particular, MBGC is equivalent to the Minimum Test Collection Problem (MTCP) [12]. This implies that the MBGC inherits all the applications of the MCC and MTCP: testing of electronic boards [15], fault analysis, medical diagnostics, and pattern recognition [12]. Furthermore, there exists a reduction from the (minimum) Vertex Coloring Problem (VCP) [18] whereby the minimum cardinality of a cut cover for a graph G is equal to $\lceil \log \chi(G) \rceil$, where $\chi(G)$ is the chromatic number of G.

Notationwise, we shall indicate a connected bipartite subgraph edge cover by $H = \{H_k \mid k \leq m\}$ for some integer m, where $H_k = (A_k, B_k, E_k)$ for all $k \leq m$. We usually employ m^* to indicate the optimal cover cardinality and $\bar{m}$ to indicate an upper bound on m^*, such as for example the cardinality of a cover found by a heuristic.

2 Worst-case complexity of the MBGC

We show **NP**-completeness of the MBGC by reduction from the MCC. The crucial part of the proof consists in showing that a cut cover can be transformed into a connected bipartite graph cover of the same cardinality. We first prove three technical lemmata.

2.1 Lemma
Consider two connected bipartite subgraphs $H_1 = (V_1, E_1)$ and $H_2 = (V_2, E_2)$ of G such that:

(a) for $k \in \{1, 2\}$, V_k is bipartite;

(b) $|V_1 \cap V_2| = 1$.

Then there is a connected bipartite subgraph $\bar{H} = (\bar{V}, \bar{E})$ of G such that $\bar{V} = V_1 \cup V_2$ and $\bar{E} = E_1 \cup E_2$.

Proof. Let $\bar{E} = E_1 \cup E_2$ for subgraph $\bar{H}$, and $V_1 \cap V_2 = \{v_0\}$. Every new cycle γ created in $\bar{H}$ when joining H_1 and H_2 is necessarily formed of a cycle of H_1 and a cycle of H_2 intersected in v_0. Since these two cycles have even length, then γ has also an even length, so $\bar{H}$ contains no odd cycle and is indeed a connected bipartite subgraph. □

2.2 Lemma
Given two connected bipartite subgraphs $H_1 = (V_1, E_1)$ and $H_2 = (V_2, E_2)$ of G such that $V_1 \cap V_2 = \emptyset$, there is a connected bipartite subgraph $H = (\bar{V}, \bar{E})$ of G such that $\bar{V} \supseteq V_1 \cup V_2$.

Proof. Since G is a connected graph there must exist vertices $v_1 \in V_1$ and $v_2 \in V_2$ connected by a path $p = (v_1, u_1, \ldots, u_q, v_2)$ in G, with $u_l \notin V_1 \cup V_2$ for $l \in \{1, \ldots, q\}$. Since p is a connected bipartite subgraph of G such that its vertex set intersects V_1 (resp. V_2) in $\{v_1\}$ (resp. $\{v_2\}$), by Lemma 2.1 there exists a

connected bipartite subgraph H of G whose vertex set consists of V_1, V_2 and the vertices in p. We remark that H can be derived from H_1, H_2 in polynomial time. □

Let $H^* = \{H_k^* \mid k \leq m^*\}$ (where $H_k^* = (A_k^*, B_k^*, E_k^*)$ for all $k \leq m^*$) be an optimal solution (of cardinality m^*) to the MBGC problem on G.

2.3 Lemma
There exists an optimal solution $H' = \{H'_k \mid k \leq m^*\}$ of the MBGC such that, for all $k \leq m^*$:

(a) H_k^* is a subgraph of H'_k;

(b) H'_k has a maximal edge set;

(c) H'_k spans V.

Proof. First of all notice that any H'_k satisfying (a) and (c) may be enlarged to be maximal with respect to the number of edges by simply inserting as many edges as possible without changing the bipartition property. Now, supposing we can find a maximal spanning H'_k whose edge set contains all the edges in H_k^*, H' is still a feasible solution (it covers E) and has the same cardinality m^* as H^*, so it suffices to show that we can enlarge H_k^* so that its vertex set is V. This can be done as follows, for all k such that $U_k = V \setminus (A_k^* \cup B_k^*) \neq \emptyset$. For all $u \in U_k$ find the shortest path $p = (u, w_1, \ldots, w_l, v)$ in G to the vertex $v \in H_k^*$ closest to u; it is evident that $w_i \in U_k$ for all $i \leq l$, otherwise v would not be closest to u. Now any path is a connected bipartite graph, and $p' = p \setminus \{v\}$ is a connected bipartite graph whose vertex set has no intersection with H_k^*. Thus by Lemma 2.2 we can enlarge H_k^* to a bipartite graph H' whose vertex set contains all of the vertices of p (see Fig. 1). We can now repeat the same construction for all remaining vertices in U_k to obtain the desired H'_k □

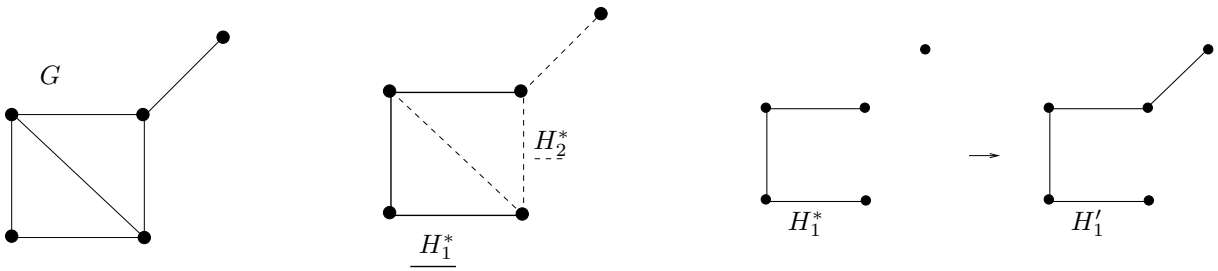

Figure 1: Proof of Lemma 2.3. A graph G, an optimal bipartite graph cover H_1^*, H_2^*, and the way to extend H_1^* so that it stays bipartite and spans V.

We consider the decision problem underlying the MBGC:

> Bipartite Graph Cover (BGC). Given an undirected graph $G = (V, E)$ and a positive integer m, is there a family $\{H_k = (A_k, B_k, E_k) \mid k \leq m\}$ of cardinality m of (not necessarily induced nor complete) connected bipartite subgraphs of G such that $E = \bigcup_{k \leq m} E_k$?

and the decision version of the MCC problem:

> Cut Cover (CC). Given an undirected graph $G = (V, E)$ and a positive integer m, is there a family $\{C_k = \{V_k, V \setminus V_k\} \mid k \leq m\}$ of cardinality m of cutsets of G such that $E = \bigcup_{k \leq m} \{\{i, j\} \mid i \in V_k, j \in V \setminus V_k\}$?

2.4 Theorem
The BGC decision problem is **NP***-complete.*

Proof. First, we remark that BGC is trivially in **NP**: checking that every subgraph in the solution is bipartite, connected and the union of these subgraphs is E can be done in polynomial time. Secondly, we exhibit a polynomial reduction from the CC problem. Consider an instance (G, m) of the CC problem: trivially, it is also an instance of the BGC problem. Assume that (G, m) has a solution w.r.t. the CC problem, namely $C = \{C_k \mid k \leq m\}$ where C_k is a cutset of G defined by a partition $\{V_k, V \setminus V_k\}$ of V. Any cutset of G can be seen as a disconnected graph whose connected components are connected bipartite subgraphs of G. By Lemma 2.2, these can be joined (in polynomial time) into a single connected bipartite subgraph of G. We therefore obtain a family $H = \{H_k \mid k \leq m\}$ of connected bipartite subgraphs of G of cardinality m. Conversely, suppose that $H = \{H_k \mid k \leq m\}$ is a solution of (G, m) w.r.t. the BGC problem. Then by Lemma 2.3 we can derive a family $H' = \{H'_k \mid k \leq m\}$ which is a solution to (G, m) and whose components H'_k are spanning in V. Since any spanning bipartite subgraph of G is also a cutset of G, H' is also a solution to the CC problem of the same cardinality. This concludes the proof. □

As a corollary, the MBGC is **NP**-hard. Notice that Thm. 2.4 is stronger than usual reduction proofs, as the solutions to the two problems have the same cardinality, which implies that an optimal solution to the MBGC reduced from an optimal solution to the MCC also has the same objective function value. For this reason, the MBGC inherits all the approximability results derived on the MCC, namely that it is approximable within $1 + (\log|V| - 3\log\log|V|)/m^*$, and that there is no polynomial time algorithm with relative error less than 1.5 [18]. Furthermore, it is solvable in polynomial time for planar graphs, and $\lceil \log|V| \rceil$ is always a valid upper bound [4] (this is derived from recursively identifying the maximum cut in a complete graph and splitting V in two cardinality balanced subsets).

3 Binary Linear Programming formulation of the MCC problem

A mathematical programming formulation of the MBGC involves connectivity constraints. This can be done in polynomial size by using flow variables [16] but it results in formulations of excessive sizes for all practical purposes. However, since the MBGC is equivalent to the MCC as explained above, an optimal objective function value for the MCC is also optimal for the MBGC. We therefore present an Integer Programming (IP) formulation for the MCC in this section. Let $\bar{m}$ be an upper bound to m^* (for instance $\bar{m} = \lceil \log|V| \rceil$).

We consider three sets of binary variables:

$$\forall k \leq \bar{m} \quad y_k = \begin{cases} 1 & \text{if the } k\text{-th cut is in the cover} \\ 0 & \text{otherwise,} \end{cases}$$

$$\forall i \in V, k \leq \bar{m} \quad x_{ik} = \begin{cases} 1 & \text{if node } i \text{ is in } V_k \\ 0 & \text{otherwise,} \end{cases}$$

$$\forall i \in V, j \in V, k \leq \bar{m} \quad e^k_{ij} = \begin{cases} 1 & \text{if edge } \{i,j\} \text{ is in the } k\text{-th cut} \\ 0 & \text{otherwise,} \end{cases}$$

The objective is to minimize the number of cuts in the cover:

$$\min \sum_{k=1}^{\bar{m}} y_k.$$

We consider edge covering constraints, which ensure that every edge $\{i, j\}$ of E belongs to at least one subgraph of the cover.

$$\forall \{i,j\} \in E \quad \sum_{k=1}^{\bar{m}} e^k_{ij} \geq 1.$$

The following constraints ensure that if node i belongs to V_k then $y_k = 1$.

$$\forall k \leq \bar{m}, \forall i \in V \quad y_k \geq x_{ik}.$$

If cut C_k is not in the cover, then no edges must be assigned to it.

$$\forall k \leq \bar{m} \quad \sum_{\{i,j\} \in E} e_{ij}^k \leq |E| y_k.$$

Lastly, we make sure that all edges in the cover have one vertex in V_k and the other in $V \setminus V_k$:

$$\forall \{i,j\} \in E, \forall k \leq \bar{m} \quad e_{ij}^k = (x_{ik} - x_{jk}) \mod 2. \tag{1}$$

We remark that (1) is a nonlinear (nonconvex) constraint. The set of points

$$Q = \{(\chi, \xi, \zeta) \mid \zeta = |\chi - \xi| \wedge \chi, \xi \in \{0,1\}\}$$

is represented by the large black dots in Fig. 2. By inspection, it is easy to see that all integral points are vertices of the convex envelope of Q (thick lines), which is a polytope. We can therefore replace Q with the set of linear constraints that are facets of the polytope. The representation of the polytope Q

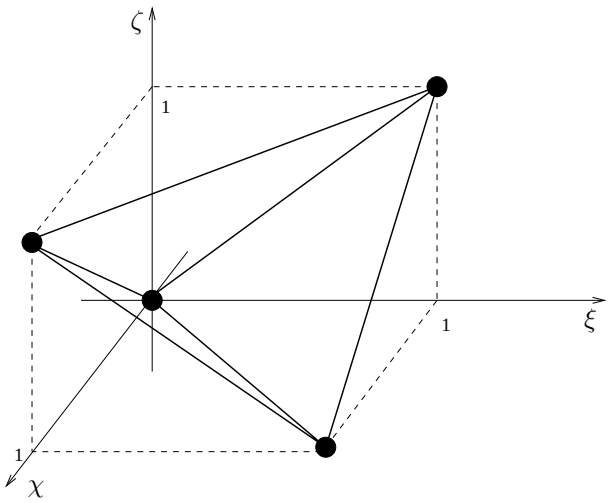

Figure 2: The set Q and its convex envelope.

by inequalities is $\{\zeta \leq \chi + \xi, \zeta \leq 2 - \chi - \xi, \zeta \geq \chi - \xi, \zeta \geq \xi - \chi\}$. We therefore replace (1) with

$$\begin{aligned}
\forall \{i,j\} \in E, \forall k \leq \bar{m} \quad e_{ij}^k &\leq x_{ik} + x_{jk} \\
\forall \{i,j\} \in E, \forall k \leq \bar{m} \quad e_{ij}^k &\leq 2 - x_{ik} - x_{jk} \\
\forall \{i,j\} \in E, \forall k \leq \bar{m} \quad e_{ij}^k &\geq x_{ik} - x_{jk} \\
\forall \{i,j\} \in E, \forall k \leq \bar{m} \quad e_{ij}^k &\geq x_{jk} - x_{ik}.
\end{aligned}$$

The purpose of introducing a mathematical programming formulation for this problem is that of obtaining exact optima on a test set of small but meaningful instances; these are used to benchmark heuristic algorithms. This is discussed in Sect. 4.3.

4 Heuristics

In this section we discuss two types of heuristics targeting the MBGC problem. The first one (Sect. 4.1) is a greedy constructive heuristic scheme along with four different "next best subgraph" procedures. The second type (Sect. 4.2) is a local search based on iteratively moving edges out of a subgraph in the cover. We use these two heuristic types conjunctively, finding an initial solution first and trying to improve it later.

4.1 Greedy constructive heuristic

We follow a classic greedy constructive meta-algorithm as follows:

1. Accept $G = (V, E)$ as input
2. Let $H = \emptyset$, $k = 1$, $E' = E$
3. Choose a (not necessarily connected) bipartite subgraph $H_k = (A_k, B_k, E_k)$ of G
4. For all $\{u, v\} \in E'$, if u, v do not both belong to A_k or B_k:
   - if $u, v \notin A_k \cup B_k$ then add u to A_k, v to B_k
   - if $u \in A_k$ and $v \notin A_k \cup B_k$ add v to B_k
   - if $u \in B_k$ and $v \notin A_k \cup B_k$ add v to A_k
   - if $v \in A_k$ and $u \notin A_k \cup B_k$ add u to B_k
   - if $v \in B_k$ and $u \notin A_k \cup B_k$ add u to A_k

   finally, add $\{u, v\}$ to E_k
5. If H_k is disconnected, connect it as per Lemma 2.2
6. Let $H = H \cup \{H_k\}$, $E' = E' \setminus E_k$
7. If $E' \neq \emptyset$ let $k = k + 1$ and go to Step 3.
8. Let $\bar{m} = |H|$
9. Return H, $\bar{m}$.

According to different ways of constructing H_k in Step 3, we obtain different greedy constructive heuristics. The usual choice would be "choose H_k that maximizes $|E_k|$", but since this is equivalent to finding a maximum cut in a graph, it is itself an **NP**-hard problem. In the rest of this section we propose four different implementations of Step 3, which attempt to find reasonably dense bipartite subgraphs H_k. We remark that Step 4 ensures that each H_k has a maximal number of edges, and that Step 5 marks the distinction between a greedy heuristic for the MBGC and one for the MCC.

4.1.1 Largest stars order

Notationwise, for a graph $G = (V, E)$ and every $u \in V$ we let $\delta(u) = \{v \in V \mid \{u, v\} \in E\}$. H_k is a spanning tree of G (hence a connected bipartite subgraph) constructed by Breadth-First-Search (BFS) on V and depending on an order on V obtained as follows:

$$\forall u, v \in V \quad u < v \leftrightarrow |\delta(u) \cap E'| < |\delta(v) \cap E'|.$$

The worst-case time complexity of this algorithm is $O(|V|^2 \bar{m})$.

4.1.2 Kruskal's spanning tree

H_k is a spanning tree of G (hence a connected bipartite subgraph) constructed by using Kruskal's algorithm, where the edges are ordered by pooling edges in E' (those still to be covered) before those in $E' \setminus E$ (those already covered). The worst-case time complexity of this algorithm is $O((|E| \log |V| + |V|^2)\bar{m})$, which for dense graphs is $O(|V|^2 \log |V| \bar{m})$ (some smarter implementations of Kruskal's algorithms exist, see [23], Sect. 10.1.2).

4.1.3 Maximum Laplacian eigenvalue

For each connected component of the current "remaining" graph (V', E') (where V' is the set of vertices induced by E'), we identify its approximate maximum cut. H_k is the union of all these cuts. Each maximum cut approximation is obtained by computing the maximum eigenvalue of the graph Laplacian [8].

Let $\{A, B\}$ be a partition of V. For all $i \in V$ consider decision variables $x_i = 1$ if $i \in A$ and $x_i = -1$ if $i \in B$. Then $f(x) = \frac{1}{4}\sum_{\{i,j\}\in E}(x_i - x_j)^2$ counts the number of intercluster edges between A and B. We also note that:

$$\begin{aligned} 4f(x) &= \sum_{\{i,j\}\in E}(x_i^2 + x_j^2) - 2\sum_{\{i,j\}\in E} x_i x_j = \sum_{\{i,j\}\in E} 2 - \sum_{i,j\in V} x_i a_{ij} x_j = \\ &= 2|E| - x^\top M x = \sum_{i\in V} x_i d_i x_i - x^\top M x = x^\top (D - M)x, \end{aligned}$$

where $M = (a_{ij})$ is the adjacency matrix of G and D is the matrix with the degree of vertex i on the diagonal and zero elsewhere. The matrix $L = D - M$ is known as the *graph Laplacian*. The function f can be written as $f(x) = \frac{1}{4}x^\top L x$. Since on average it is more likely to obtain larger cuts out of balanced $\{A, B\}$ partitions (i.e. partitions where $|A|$ and $|B|$ are roughly equal), we also require that $\sum_{i\in V} x_i = 0$ (this obviously only holds for $|V|$ even). In order to find an approximate solution to the following problem:

$$\left.\begin{array}{rll} \max\limits_{x\in\{-1,1\}} & \frac{1}{4}x^\top L x & \\ \text{s.t.} & x^\top \mathbf{1} &= 0, \end{array}\right\} \tag{2}$$

we consider its continuous relaxation ($x \in [-1, 1]$), strengthened by the valid cut $x^\top x = |V|$. In view of the fact that we only consider this relaxation in the following, the requirement that $|V|$ is even is not vital for the approximation to hold.

There is an interesting relation between the solution of (2) in the minimization direction (i.e. the determination of a *minimum* balanced cut, which is itself an **NP**-hard problem) and the second-smallest eigenvalue of L. Let $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ be the ordered eigenvalues of L (where $n = |V|$) and $u_1, \ldots, u_n$ be the corresponding eigenvectors, normalized so that $||u_i|| = \sqrt{n}$ for all $i \le n$. Some established results [17, 5] show that L is symmetric positive semidefinite, that $u_1 = \mathbf{1}$ and that $\lambda_1 = 0$; and, if G is connected, that $\lambda_2 > 0$. By the definition of eigenvalue and eigenvector, we have $Lu_i = \lambda_i u_i \Rightarrow {u_i}^\top L u_i = \lambda_i {u_i}^\top u_i = \lambda_i n$ for all $i \le n$. Because of orthogonality of the eigenvectors, if $i \ge 2$ we have $u_i u_1 = 0$, which implies $u_1 \mathbf{1} = 0$. Lastly, we chose the normalization of the eigenvectors in such a way that ${u_i}^\top u_i = n$. Thus, since $\lambda_1 = 0$, λ_2 yields the smallest objective function value $\frac{n}{4}\lambda_2$ with solution $\bar{x} = u_2$. Normally, the values $\bar{x}$ are not in $\{-1, 1\}$, but by approximating $\bar{x}_i$ to its closest value in $\{-1, 1\}$, breaking ties in such a way as to keep the bisection balanced, one obtains a practically efficient approximation of the minimum balanced cut. It now suffices to notice that the maximum eigenvalue λ_n yields the *maximum* objective function value $\frac{n}{4}\lambda_n$ with solution $x' = u_n$ feasible in the continuous relaxation of (2) [8].

The worst-case time complexity of this algorithm (based on computing eigenvalues and eigenvectors of the graph Laplacian) is $O(|V_1'|^3 + \cdots + |V_\gamma'|^3)$ where γ is the number of connected components of the input graph and V_j' is the set of vertices of the j-th connected component. For complete graphs, $\gamma = 2^{k-1}$, where k is the iteration index in the greedy algorithm, and a reasonable average value for $|V_j'|$ at the k-th iteration is $\frac{|V|}{2^{k-1}}$. We can hence estimate the average time complexity for the k-th iteration of this algorithm to be $\gamma\left(\frac{|V|}{2^{k-1}}\right)^3 = 2^{k-1}\frac{|V|^3}{2^{3(k-1)}} = \frac{4|V|^3}{4^k}$. The number of iterations of the greedy algorithm is $\bar{m}$, which for complete graphs and supposing the approximation exact is $\lceil \log_2 |V| \rceil$. So an estimation of the average time complexity of the whole greedy algorithm is $4|V|^3 \sum_{k\le\lceil\log_2|V|\rceil}\left(\frac{1}{2^k}\right)^2 = 4|V|^3\Gamma$ where $\Gamma = \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{4}\right)^2 + \cdots + \left(\frac{1}{|V|}\right)^2\right)$. Since $\Gamma < \sum_{k\in\mathbb{N}}\frac{1}{2^k} = 1$, the overall complexity is $O(|V|^3)$.

4.1.4 DFS Forest

This algorithm, suggested in [6], consists in setting H_k as a Depth-First-Search (DFS) spanning forest of (V, E'). The worst-case time complexity of this algorithm is $O(|E| + |V|)$ and it is shown in [6] that the overall greedy algorithm worst-case time complexity based on DFS spanning forests is $O((|E| + |V|) \log |V|)$, which for dense graphs is $O(|V|^2 \log |V|)$. This algorithm is guaranteed to find solutions H^* such that $|H^*| \leq \lceil \log |V| \rceil$ (this bound is obtained in [6] by considering the performance on complete graph as the worst case).

4.2 The "budge the edge" local search

We start from an existing edge cover of G by a family of connected, not necessarily induced bipartite subgraphs $H = \{H_k \mid k \leq m\}$ for an integer m (usually $m = \bar{m}$ obtained by some constructive heuristic). Each H_k has bipartition (A_k, B_k) and edge set $E_k = E(H_k)$. For any $k \leq m$ and $e \in E_k$ we let $H_k \smallsetminus e$ be the subgraph of G consisting of all edges of E_k but e. Likewise, for $f \in E \smallsetminus E_k$ we let $H_k \cup f$ be the subgraph of G consisting of all edges of E_k and f. We let $\mathcal{H}$ be the set of all connected bipartite edge covers of G (i.e. all feasible solutions to the MBGC).

The intuitive idea behind the proposed local search is as follows. In order to decrease $|H|$, we consider the subgraph of H with the set of edges of minimum cardinality (call it E_1) and try to "budge" each edge in E_1 to other subgraphs H such that the resulting cover still consists of connected bipartite subgraphs. We aim to drive $|E_1|$ to zero, which effectively means that the cardinality of the cover is decreased by one unit. More precisely, for each $e \in E_1$ we attempt to find a subgraph in H (say H_2, different from H_1) such that $E_2 \cup e$ is a connected bipartite subgraph of G in order to move e from H_1 to H_2. If no such H_2 exists, it is possible that there are distinct graphs H_2, H_3 (both different from H_1) and edges $e_1 \in E_1, e_2 \in E_2$ such that $H_2 \cup e_1 \smallsetminus e_2$ and $H_3 \cup e_2$ are connected bipartite subgraphs of G: in this case we move e_1 from H_1 to H_2 and e_2 from H_2 to H_3.

We now extend this idea to general sequences of indices and edges. Let S be the set of all (finite) pair sequences $\langle (e_j, h_j) \mid \forall j \ (e_j \in E \wedge h_j \leq m) \wedge \forall i \neq j \ (e_i \neq e_j \wedge h_i \neq h_j) \rangle$. Given a sequence $\langle s \rangle \in S$ of length ℓ and $H \in \mathcal{H}$, we let $\alpha(H, \langle s \rangle)$ be the set of subgraphs H'_k of G constructed as follows:

$$\begin{array}{rrcl} & H'_{h_1} & = & H_{h_1} \smallsetminus e_1 \\ \forall j \in \{2, \ldots, \ell - 1\} & H'_{h_j} & = & H_{h_j} \cup e_{j-1} \smallsetminus e_j \\ & H'_{h_\ell} & = & H_{h_\ell} \cup e_{\ell-1} \end{array}$$

(e_ℓ is a "dummy" edge that plays no role since it does not appear in the above definition, and can therefore be chosen at leisure). We now define neighbourhoods $\mathcal{N}(H, \ell)$ in $\mathcal{H}$ with center $H \in \mathcal{H}$ and radius $\ell \in \mathbb{N}$ as follows:

$$\mathcal{N}(H, \ell) = \{\alpha(H, \langle s \rangle) \mid \langle s \rangle \in S \wedge |\langle s \rangle| \leq \ell\} \cap \mathcal{H}.$$

In practice, we suppose the initial cover $H = \{H_1, \ldots, H_m\}$ is ordered by increasing $|H_k|$. We try to "empty" H_1 by iteratively "budging" edges along sequences of other subgraphs in the cover; if H_1 cannot be emptied, we consider H_2, and so on. We therefore need to know what happens when an edge $e \in E$ is added to a bipartite subgraph H_k for some $k \leq m$. We remark that for all $k \leq m$ and $e \in E \smallsetminus E_k$, recognizing whether $H_k \cup e$ is a bipartite connected subgraph is an $O(1)$ operation.

Suppose we now want to budge an edge e out of a subgraph H_k and along a sequence $\langle s \rangle \in S$ of length at most ℓ. One possible way to construct $\langle s \rangle = \langle (e_j, h_j) \rangle$ is as follows. Let $\beta = 0$, $j = 1$, $k \leq m$, $e \in E_k$, $I = \{1, \ldots, m\} \smallsetminus \{k\}$ be ordered by a given ordering $<$. Set $h_j = k$ and $e_j = e$.

1. If $j \geq \ell$, set $q = \ell$ and exit.

2. If $\exists i \in I$ such that $H_i \cup e_j$ is connected and bipartite, let $h_{j+1} = i$ and e_{j+1} be any edge in E; set $\beta = 1$, $q = j$ and exit.

3. Otherwise, choose $i \in I$ such that $H_i \cup e_j$ is connected, not bipartite, and has one single cycle γ of odd length. If no such i exists, set $q = j$ and exit.

4. Increase j, let $h_j = i$, e_j be any edge of γ but e_{j-1}, and $I = I \setminus \{i\}$.

5. Go back to Step 1.

We denote the algorithm above by BUDGE(H, k, e, ℓ); its return value is a pair $(\beta, \langle s \rangle)$. If BUDGE terminates with $\beta = 1$, it means it has found a sequence $\langle s \rangle = \langle (h_1, e_1) \dots, (h_q, e_q) \rangle$ (with $q \leq \ell$) such that $\alpha(H, \langle s \rangle) \in \mathcal{N}(H, q) \subseteq \mathcal{N}(H, \ell)$, otherwise it terminates with $\beta = 0$. The worst-case time complexity of BUDGE is $O(\ell m \omega)$, where ω is the complexity of verifying that a connected graph has exactly one cycle of odd length incident on a given edge. We remark that BUDGE is not designed to explore the whole of $\mathcal{N}(H, \ell)$. Its scope depends on the initial ordering on I, on the choice of e_j in Step 4, and on the fact that for simplicity we do not consider connected but not bipartite graphs where the addition of an edge creates more than one odd cycle. These limitations were imposed in order to keep BUDGE practically functional with respect to CPU time.

The overall local search is given in Alg. 1. The search scope limitations of BUDGE imply that Alg. 1 cannot explore the whole of $\mathcal{H}$. The worst-case time complexity of Alg. 1 is $O(m|E|O(\text{BUDGE})) = O(\ell m^2 \omega |E|)$.

Algorithm 1 The "budge the edge" local search.

```
INPUT: An initial edge cover H and a positive integer ℓ.
OUTPUT: An edge cover in N(H, ℓ) of (hopefully) smaller cardinality.
Let K = {1, ..., m}
Order H = {H_1, ..., H_m} by increasing cardinality of the H_k's (and K accordingly).
while K ≠ ∅ do
  Let k = min_{i∈K} i and E' = E_k.
  for all e ∈ E' do
    Let (β, ⟨s⟩) = BUDGE(H, k, e, ℓ).
    if β = 1 then
      Let H = α(H, ⟨s⟩).
    else
      BREAK
    end if
  end for
  if |E_k| = 0 then
    Let H = H \ {H_k}.
  end if
  Let K = K \ {k}.
end while
```

4.3 Computational results

We generated a collection of random instances of various types, densities and sizes, and solved them using: (a) the MCC formulation of Sect. 3; (b) the heuristics of Sect. 4. All results were obtained on an Intel Core Duo 1.2GHz with 1.5GB RAM running Linux. The MCC formulation was modelled in AMPL [10] and solved by ILOG CPLEX 10.1 [14]; the heuristics were coded in GNU C++ 4.1.2.

We used two test suites. The first one, named SMALL, is composed of 48 small graphs of various types. We remark that the 18 instances listed in the left hand side part of Table 1 have been generated with

different topological properties (hypercubic, mesh-like, tripartite), but in fact they are all bipartite graphs, so the optimal cover consists of the graph itself ($m = 1$). On the instances in the SMALL suite, we were able to obtain guaranteed optimal values by means of the MCC formulation — we used these results to benchmark the heuristics. The second test suite, named LARGE, consists of a set of 30 randomly generated undirected graphs with $100 \leq |V| \leq 900$ where each edge is generated with probability in $\{0.2, 0.5, 0.8\}$. The LARGE test suite is used to assess the scalability of our heuristics. All graphs can be downloaded from `http://www.lix.polytechnique.fr/~liberti/bipgraphcover`; the number of vertices and edges is reported in Table 1.

Name	$\|V\|$	$\|E\|$
hypercube-2.gph	4	4
hypercube-3.gph	8	12
mesh-3.gph	9	12
mesh-4.gph	16	24
mesh-5.gph	25	40
mesh-6.gph	36	60
tripartite-5_0.2.gph	15	5
tripartite-5_0.4.gph	15	20
tripartite-5_0.6.gph	15	37
tripartite-5_0.8.gph	15	39
tripartite-10_0.2.gph	30	49
tripartite-10_0.4.gph	30	90
tripartite-10_0.6.gph	30	106
tripartite-10_0.8.gph	30	168
tripartite-15_0.2.gph	45	75
tripartite-15_0.4.gph	45	179
tripartite-15_0.6.gph	45	282
tripartite-15_0.8.gph	45	361

Name	$\|V\|$	$\|E\|$
complete-5.gph	5	10
complete-6.gph	6	15
complete-7.gph	7	21
complete-8.gph	8	28
torus-3.gph	9	18
torus-4.gph	16	32
torus-5.gph	25	50
torus-6.gph	36	72
triangle-5.gph	15	30
triangle-6.gph	21	45
triangle-7.gph	28	63

Name	$\|V\|$	$\|E\|$
random-6_0.2.gph	6	9
random-6_0.4.gph	6	8
random-6_0.6.gph	6	12
random-6_0.8.gph	6	14
random-8_0.2.gph	8	16
random-8_0.4.gph	8	13
random-8_0.6.gph	8	20
random-8_0.8.gph	8	25
random-10_0.2.gph	10	18
random-10_0.4.gph	10	24
random-10_0.6.gph	10	33
random-10_0.8.gph	10	39
random-12_0.2.gph	12	22
random-12_0.4.gph	12	35
random-12_0.6.gph	12	38
random-12_0.8.gph	12	55
random-14_0.2.gph	14	29
random-14_0.4.gph	14	48
random-14_0.6.gph	14	66

Table 1: Problem statistics for the SMALL test suite: on the left, "hidden" bipartite graphs (for which $m = 1$); in the center, other structured graphs; on the right, small random graphs.

Table 2 reports computational results on the SMALL test suite, comparing exact values (columns labelled "CPLEX", which also reports the time taken to find the exact optimum), values obtained using the constructive heuristic in [22] (labelled "CTW07"), and each of the four greedy constructive heuristics proposed in: Sect. 4.1.1 (labelled "L"), Sect. 4.1.2 (labelled "K"), Sect. 4.1.3 (labelled "E"), Sect. 4.1.4 (labelled "D"). The first column contains two sub-columns reporting the cardinality $\bar{m}$ of the cover (left) and seconds of user CPU time (right). Each other column only contains the cover cardinality $\bar{m}$, because the CPU time is negligible. For each heuristic $h \in \{\text{CTW07,L,K,E,D}\}$ and instance $i \in \text{SMALL}$, the average approximation ratio $\Delta_h = \frac{1}{|\text{SMALL}|} \sum_{i \in \text{SMALL}} \frac{\bar{m}_i^h - m_i^*}{m_i^*}$, where $\bar{m}_i^h$ is the cover cardinality obtained by heuristic h on instance i, and m_i^* is the minimum cover cardinality determined by CPLEX, is reported on the last line. The best results in terms of approximation ratio are obtained by CTW07 and D, followed closely by L, K and E. User CPU times taken by all the heuristics was mostly 0.00s, with 0.01s in a few cases.

CTW07 Heuristic. This heuristic is described in detail in [22], which can be found at `http://www.lix.polytechnique.fr/~liberti/bgc-ctw07.pdf`. It progressively selects the vertex v with highest star degree, disconnected from A_k but whose star intersects B_k; v is added to A_k, its vertex star to B_k and its edge star to E_k; when no further vertices may be added to A_k, k is increased and the procedure is repeated with a smaller edge set $E = E \setminus E_k$; when $E = \emptyset$, the cover is complete. We remark that although this heuristic performs well on the graphs in the SMALL instance set, it is the worst-performing (quality-wise) of the tested heuristics.

Table 3 reports computational results on the SMALL test suite, listing cover cardinalities obtained with each constructive heuristic $h \in \{\text{CTW07,L,K,E,D}\}$ followed by an application of the "budge-the-edge" local search heuristic. Because of the relatively high time complexity measure of the latter, we set $\ell = 1$; as this also implies that we never perform Step 3 of BUDGE (because $j = \ell$ at the next iteration anyway), the worst-case time complexity of Alg. 1 is reduced to $O(|E|\bar{m}^2)$. We emphasized in boldface the cardinality of the covers found by heuristics improved by the local search. The improved average

	CPLEX		CTW07	L	K	E	D
hypercube-2	1	0.00	1	2	2	2	1
hypercube-3	1	0.01	1	2	2	3	1
mesh-3	1	0.02	1	2	2	3	1
mesh-4	1	0.18	1	2	2	4	1
mesh-5	1	1.24	1	2	2	4	1
mesh-6	1	1.16	1	2	2	4	1
tripartite-5_0.2	1	0.01	2	1	1	2	2
tripartite-5_0.4	1	0.22	1	2	2	4	1
tripartite-5_0.6	1	0.06	1	2	3	4	1
tripartite-5_0.8	1	0.06	1	2	3	5	1
tripartite-10_0.2	1	1.19	1	2	2	4	3
tripartite-10_0.4	1	0.32	1	2	4	4	2
tripartite-10_0.6	1	0.43	1	2	4	5	2
tripartite-10_0.8	1	0.62	1	2	6	5	1
tripartite-15_0.2	1	0.50	1	2	2	5	3
tripartite-15_0.4	1	1.08	1	2	5	6	3
tripartite-15_0.6	1	1.42	1	2	7	6	2
tripartite-15_0.8	1	1.72	1	2	9	6	1
complete-5	3	0.16	4	4	4	3	3
complete-6	3	1.10	5	5	5	3	3
complete-7	3	2.54	6	6	6	3	3
complete-8	3	7.75	7	7	7	4	3
torus-3	2	1.58	2	3	3	4	2
torus-4	1	0.12	1	2	3	3	1
torus-5	2	1397.36	2	3	3	4	2
torus-6	1	15.29	1	2	3	4	1
triangle-5	2	7.54	2	4	3	4	3
triangle-6	2	40.53	4	4	3	3	3
triangle-7	2	113.80	4	5	3	4	3
random-6_0.2	2	0.16	2	3	2	3	2
random-6_0.4	2	0.07	3	3	2	3	2
random-6_0.6	2	0.32	3	4	3	3	3
random-6_0.8	3	0.91	4	5	3	3	3
random-8_0.2	2	0.89	2	3	3	3	3
random-8_0.4	2	0.68	2	3	2	3	2
random-8_0.6	2	1.76	3	4	3	3	3
random-8_0.8	3	11.67	5	6	5	3	3
random-10_0.2	2	5.74	2	3	3	4	2
random-10_0.4	2	2.87	3	4	4	4	3
random-10_0.6	3	22.34	4	5	5	3	3
random-10_0.8	3	86.32	6	6	6	4	4
random-12_0.2	2	3.70	4	4	3	4	3
random-12_0.4	2	15.10	4	4	4	4	4
random-12_0.6	3	287.44	4	5	4	4	3
random-12_0.8	3	43.56	6	7	6	4	3
random-14_0.2	2	40.69	2	3	3	4	2
random-14_0.4	2	32.49	4	5	4	4	4
random-14_0.6	3	662.29	6	6	7	4	4
Δ_h	0.00		0.33	0.88	1.30	1.63	0.34

Table 2: Results for the SMALL test suite — constructive heuristics. Δ_h is the average approximation ratio computed over the heuristic h. The instance names are self-explanatory, and `complete-`n stands for the complete graph K_n.

approximation ratios are on the bottom line. The best results are found by CTW07 and L, closely followed by D and K. E is last. Again, user CPU times were mostly 0.00s and occasionally 0.01s.

Table 4 reports values on the LARGE test suite, comparing constructive heuristic performance in terms of cover cardinality (left) and seconds of user CPU time (right). Since in this case we do not have exact values, we cannot rely on the approximation ratio to estimate relative performance. Instead, we rely on the average cover cardinality μ_h of the sample for $h \in \{\text{CTW07,L,K,E,D}\}$ (last line). In the LARGE instance set, computation time becomes an issue: we compare by looking at the cumulative user CPU time τ_h spent for solving the whole test suite. The best performing heuristic is D (values marked by *), followed by E.

Table 5 includes results on the LARGE test suite obtained by running the constructive heuristic followed by the "budge-the-edge" local search with $\ell = 1$. User CPU times are cumulative (i.e. they include both the time taken for carrying out the constructive heuristic and the local search itself) and improved values are emphasized in boldface. Again, the best performing heuristic is D (values marked by *), followed by

	CTW07	L	K	E	D
hypercube-2	1	**1**	2	2	1
hypercube-3	1	**1**	2	3	1
mesh-3	1	**1**	2	3	1
mesh-4	1	**1**	2	**3**	1
mesh-5	1	**1**	2	**3**	1
mesh-6	1	**1**	2	**3**	1
tripartite-5_0.2	**1**	1	1	2	2
tripartite-5_0.4	1	**1**	2	**3**	1
tripartite-5_0.6	1	**1**	**1**	**3**	1
tripartite-5_0.8	1	**1**	**1**	**3**	1
tripartite-10_0.2	1	**1**	2	**3**	3
tripartite-10_0.4	1	**1**	**1**	4	2
tripartite-10_0.6	1	**1**	**1**	**4**	2
tripartite-10_0.8	1	**1**	**1**	**4**	1
tripartite-15_0.2	1	**1**	2	**4**	3
tripartite-15_0.4	1	**1**	**1**	**5**	3
tripartite-15_0.6	1	**1**	**1**	**5**	2
tripartite-15_0.8	1	**1**	**1**	**5**	1
complete-5	**3**	**3**	**3**	3	3
complete-6	**3**	**3**	**3**	3	3
complete-7	**4**	**4**	**4**	3	3
complete-8	**4**	**4**	**4**	4	3
torus-3	2	**2**	**2**	**3**	2
torus-4	1	**1**	3	3	1
torus-5	2	**2**	**2**	**3**	2
torus-6	1	**1**	3	4	1
triangle-5	2	**3**	**2**	**3**	3
triangle-6	**3**	**3**	**2**	3	3
triangle-7	**3**	**3**	**2**	**3**	3
random-6_0.2	2	**2**	2	3	2
random-6_0.4	**2**	**2**	2	**2**	2
random-6_0.6	3	**3**	3	3	3
random-6_0.8	4	**4**	3	3	3
random-8_0.2	2	**2**	**2**	**2**	3
random-8_0.4	2	**2**	2	3	2
random-8_0.6	3	**3**	3	3	3
random-8_0.8	5	**5**	**4**	3	3
random-10_0.2	2	**2**	3	**3**	2
random-10_0.4	3	**3**	4	**3**	3
random-10_0.6	4	**4**	5	3	3
random-10_0.8	**5**	**5**	**5**	4	4
random-12_0.2	**3**	**3**	**2**	**3**	3
random-12_0.4	**3**	**3**	4	**3**	4
random-12_0.6	4	**4**	4	4	3
random-12_0.8	**5**	**6**	6	4	3
random-14_0.2	2	**2**	3	**3**	2
random-14_0.4	4	**4**	**4**	4	4
random-14_0.6	6	**5**	**5**	4	4
Δ_h	**0.19**	**0.20**	**0.47**	**1.22**	0.34

Table 3: Results for the SMALL test suite — chained constructive and "budge the edge" heuristics. Improvements w.r.t. constructive heuristics are in boldfaces.

E. In both cases, the effect of the local search is minor.

These computational results suggest that CTW07 and L+(local search) manage to identify a bipartite structure (high number of covers with cardinality 1) but perform poorly, both quality- and CPU time-wise, on large random graphs. The greedy D heuristic is the clear winner on large random graphs both quality- and CPU time-wise, closely followed (quality-wise) by the E heuristic. Fig. 3 shows the performance of the different heuristics (chained with the local search) expressed in cover cardinality vs. $|V|$. Fig. 4 shows the corresponding CPU-time performance. It appears clear that quality-wise, the performance of CTW07 grows linearly with $|V|$, whereas the performance of the other heuristics grows logarithmically with $|V|$. CPU time-wise, the kind of growth is superlinear on average. The success of CTW07 and L+(local search) on graphs that are already bipartite may be explained by the fact that these heuristics are based on a vertex-based greedy choice with respect to the corresponding vertex stars.

In conclusion, for large graphs the D heuristic is the clear winner. One last interesting remark is provided by looking at the relative CPU time performance of the D and E heuristic: the ratio of the CPU times obtained by E divided by the CPU times provided by D against $|V|$, limited to graphs in

	CTW07		L		K		E		D	
random-100_0.2	10	0.00	9	0.01	6	0.02	7	0.04	6	0.01
random-100_0.5	21	0.01	11	0.02	10	0.04	8	0.06	7	0.02
random-100_0.8	36	0.02	21	0.08	15	0.08	8	0.11	7	0.03
random-200_0.2	18	0.03	12	0.07	7	0.07	8	0.16	6	0.04
random-200_0.5	32	0.07	16	0.16	12	0.12	8	0.26	8	0.11
random-200_0.8	59	0.16	25	0.58	19	0.31	9	0.42	8	0.20
random-300_0.2	20	0.08	13	0.14	8	0.12	9	0.34	7	0.12
random-300_0.5	47	0.38	15	0.60	12	0.45	9	0.81	8	0.40
random-300_0.8	87	0.70	27	3.68	18	0.97	10	1.40	9	0.76
random-400_0.2	24	0.28	14	0.41	8	0.29	9	0.78	7	0.26
random-400_0.5	60	0.96	18	2.84	14	1.06	10	1.90	9	0.94
random-400_0.8	109	1.70	28	11.23	21	2.22	10	3.15	9	1.76
random-500_0.2	30	0.62	16	1.25	10	0.59	10	1.52	8	0.57
random-500_0.5	68	1.89	19	7.89	14	1.90	10	3.82	9	1.80
random-500_0.8	131	3.30	30	24.83	27	4.36	11	5.76	10	3.11
random-600_0.2	34	1.13	15	2.90	10	1.04	10	2.75	8	1.00
random-600_0.5	79	3.27	18	11.09	14	3.24	10	6.00	9	2.97
random-600_0.8	154	5.72	35	51.78	24	7.34	11	8.78	10	4.85
random-700_0.2	39	1.86	16	5.92	9	1.71	10	4.51	8	1.70
random-700_0.5	91	5.18	20	23.63	14	4.78	11	9.47	10	4.70
random-700_0.8	177	8.98	35	80.19	26	10.58	11	13.64	10	7.78
random-800_0.2	42	2.84	15	8.51	10	2.62	10	6.42	8	2.58
random-800_0.5	102	7.74	21	36.03	14	6.60	11	12.32	10	6.35
random-800_0.8	197	13.38	38	132.83	27	16.41	11	20.25	10	11.03
random-900_0.2	46	4.06	17	15.33	10	3.24	11	8.85	9	3.57
random-900_0.5	113	11.20	23	59.19	15	9.45	11	18.59	10	9.42
random-900_0.8	216	18.96	35	172.12	29	22.78	12	27.65	10	14.41
(μ_h, τ_h)	75.62	94.52	20.81	653.31	14.93	102.39	9.81	159.76	8.52*	80.49*

Table 4: Results for the LARGE test suite — constructive heuristics.

	CTW07		L		K		E		D	
random-100_0.2	**9**	0.01	**6**	0.02	6	0.01	**6**	0.04	**5**	0.01
random-100_0.5	**20**	0.04	**8**	0.03	10	0.04	8	0.08	7	0.04
random-100_0.8	**35**	0.14	**18**	0.09	**14**	0.10	8	0.11	7	0.05
random-200_0.2	**15**	0.06	**7**	0.08	7	0.07	8	0.14	6	0.06
random-200_0.5	32	0.24	**12**	0.22	12	0.16	8	0.26	8	0.14
random-200_0.8	59	1.13	**22**	0.68	**17**	0.42	9	0.47	8	0.23
random-300_0.2	**19**	0.15	**8**	0.20	8	0.14	**8**	0.36	7	0.14
random-300_0.5	47	1.63	**11**	0.78	12	0.58	9	0.90	8	0.44
random-300_0.8	87	9.04	**26**	5.16	18	1.78	10	1.79	9	0.97
random-400_0.2	24	0.54	**9**	0.50	8	0.34	9	0.86	7	0.28
random-400_0.5	**59**	6.24	**11**	3.50	14	1.86	10	2.43	9	1.63
random-400_0.8	**108**	26.51	**25**	13.71	21	4.06	10	3.82	9	1.90
random-500_0.2	**29**	1.67	**9**	1.54	10	0.90	**9**	1.66	8	1.04
random-500_0.5	68	12.61	**14**	9.21	14	3.08	10	4.46	9	2.44
random-500_0.8	131	57.50	**29**	30.64	**26**	9.24	11	7.74	10	4.82
random-600_0.2	**33**	3.35	**10**	3.55	10	1.48	10	3.51	8	1.82
random-600_0.5	79	23.88	**13**	13.29	14	4.74	10	6.67	9	3.59
random-600_0.8	**153**	112.09	**32**	61.01	**23**	12.33	**10**	9.44	10	6.10
random-700_0.2	**38**	5.83	**11**	8.50	9	2.02	10	5.49	8	2.16
random-700_0.5	91	44.18	**15**	28.60	14	6.62	11	12.02	10	6.10
random-700_0.8	177	197.54	**31**	93.65	26	18.42	11	17.37	10	8.61
random-800_0.2	42	9.16	**10**	10.51	10	3.38	10	7.48	8	2.89
random-800_0.5	102	71.37	**14**	40.34	14	9.85	**10**	13.33	10	7.79
random-800_0.8	**196**	305.86	**32**	149.51	**26**	27.54	**11**	23.24	10	11.94
random-900_0.2	46	14.71	**11**	18.03	10	4.02	10	9.70	9	6.41
random-900_0.5	113	107.14	**15**	65.98	15	14.00	11	21.09	10	12.89
random-900_0.8	216	469.57	**31**	205.60	**28**	40.63	12	34.39	10	15.52
(μ_h, τ_h)	75.11	1482.19	16.29	764.93	14.66	167.81	9.59	188.85	8.48*	100.01*

Table 5: Results for the LARGE test suite — chained constructive and "budge the edge" heuristics. Improvements w.r.t. constructive heuristics are in boldface.

LARGE with density 0.8, is shown in Fig. 5 (left). The sample average is 1.89 and the standard deviation (0.16) is low enough to suggest an asymptotically constant, or even decreasing, time complexity ratio. This behaviour somehow appears also in Fig. 5 (right), which plots the ratio of the CPU times of D and E over complete graphs ranging from 500 to 1400 vertices against the number of vertices. Although we were not able to test larger graphs due to a limitation in RAM size on the test hardware, this behaviour would lead one to suppose that for larger and larger graphs, the time complexity of E becomes favourable

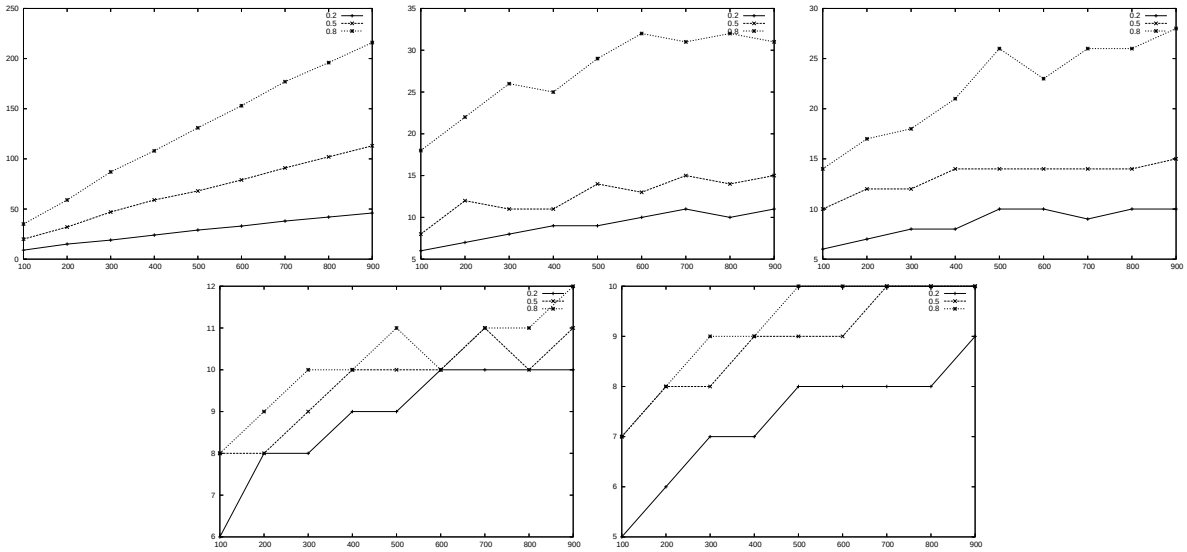

Figure 3: Plot of cover cardinality against $|V|$ on LARGE. From the top left: CTW07, L, K, E, D.

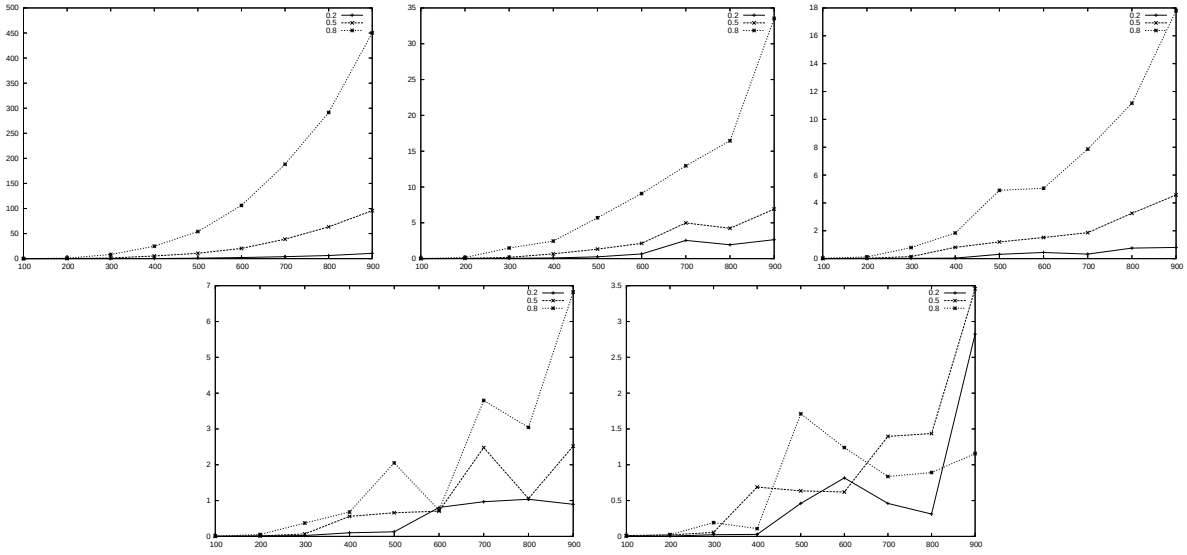

Figure 4: Plot of user CPU time against $|V|$ on LARGE. From the top left: CTW07, L, K, E, D.

with respect to the time complexity of D. This contrasts with the time complexity analysis provided in Sect. 4.1.3: a possible explanation may lie in the fact that finding eigenvalues and eigenvectors of an $n \times n$ matrix is in general an $O(n^3)$ task, but this figure does not keep into account the special structure of the graph Laplacian; another explanation may be that Γ (defined in Sect. 4.1.3) is close to $O(\frac{1}{|V|})$ or some other decreasing function of $|V|$

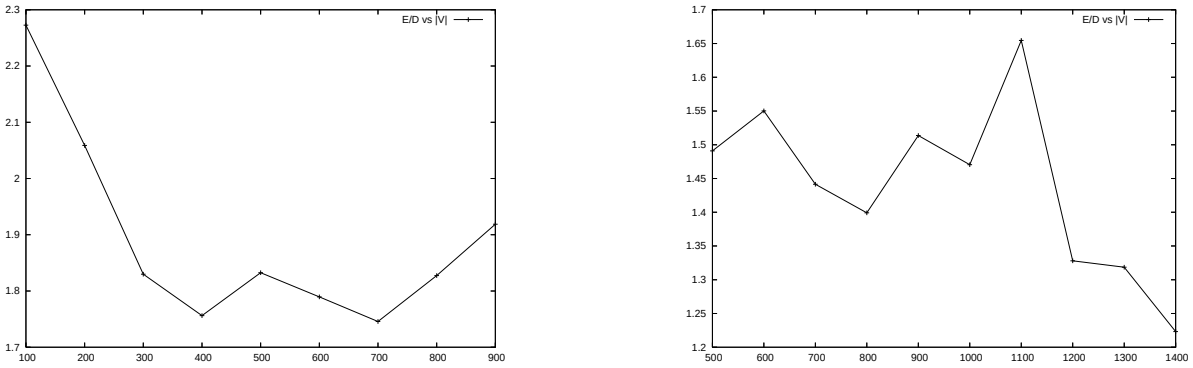

Figure 5: Plot of ratio between user CPU time of heuristic E and D against $|V|$ on dense instances (0.8) in LARGE (left) and on complete graphs ranging from 500 to 1400 vertices (right).

5 The constrained MBGC

In Network Design, covering a graph by short subgraphs is a classical issue, we refer for example to the k-hop spanning tree problem or the minimum bounded-diameter spanning forest problem where paths between any node and the root of the subgraph tree must have limited length. So, we study and obtain some interesting results on the following length constrained variant of the problem. We define the length of a graph G to be the maximum number of edges of a simple path in G.

> MINIMUM CONSTRAINED BIPARTITE GRAPH COVER (MCBGC). Given an undirected graph $G = (V, E)$ and a positive integer L, find a family $\{H_k = (A_k, B_k, E_k) \mid k = 1, \ldots, m\}$ of (not necessarily induced nor complete) connected bipartite subgraphs of G such that $\bigcup_{k \leq m} E_k = E$, the length of every subgraph H_k is at most L, and m is minimum.

In this section we prove **NP**-hardness of the MCBGC and provide a MILP formulation.

5.1 Theorem
*MCBGC is **NP**-hard for any length bound $L = 2 + 4p$, $p \in \mathbb{N}$.*

Proof. The reduction is from the Minimum Vertex Cover problem (Min-VC — this consists in finding a subset $X \subseteq V$ such that $|X|$ is minimum and for all $\{i, j\} \in E$, either $i \in X$ or $j \in X$ or both). For a length bound $L = 2$ ($p = 0$), the bipartite subgraph is a star, so in that case Min-CBGC consists in covering edges of G by a minimum number of stars, which is exactly the NP-hard Vertex Cover problem. For a lenght bound $L = 2 + 4p$, $p \geq 1$, we transform an instance $G = (V, E)$ of Min-VC into a graph $G' = (V', E')$ the following way (see Fig. 5). For every pair $\{i, j\} \in E$, we create $2p$ new vertices $v^{ij}_{i,1}, \ldots, v^{ij}_{i,p}$ and $v^{ij}_{j,1}, \ldots, v^{ij}_{j,p}$ in G' (with a slight abuse of notation we take v^{ij}_{ip} to mean $v^{\{i,j\}}_{ip}$), so that:

$$
\begin{aligned}
V' &= V \cup \{v^{ij}_{i,1}, \ldots, v^{ij}_{i,p} \mid i \in V, \{i, j\} \in E\} \\
E' &= \bigcup_{\{i,j\} \in E} E'_{ij} \\
\text{with } E'_{ij} &= \{(i, v^{ij}_{i,1}), (v^{ij}_{i,1}, v^{ij}_{i,2}), \ldots, (v^{ij}_{i,p-1}, v^{ij}_{i,p}), (v^{ij}_{i,p}, v^{ij}_{j,p}), (v^{ij}_{j,p}, v^{ij}_{j,p-1}), \ldots, (v^{ij}_{j,2}, v^{ij}_{j,1}), (v^{ij}_{j,1}, j)\}.
\end{aligned}
$$

In the following, we write e^{ij} for $(v^{ij}_{ip}, v^{ij}_{jp})$ to ease notation. We now show that every vertex cover $X \subset V$ can be transformed into a L-length bipartite cover of size $|X|$ in G' and vice-versa.

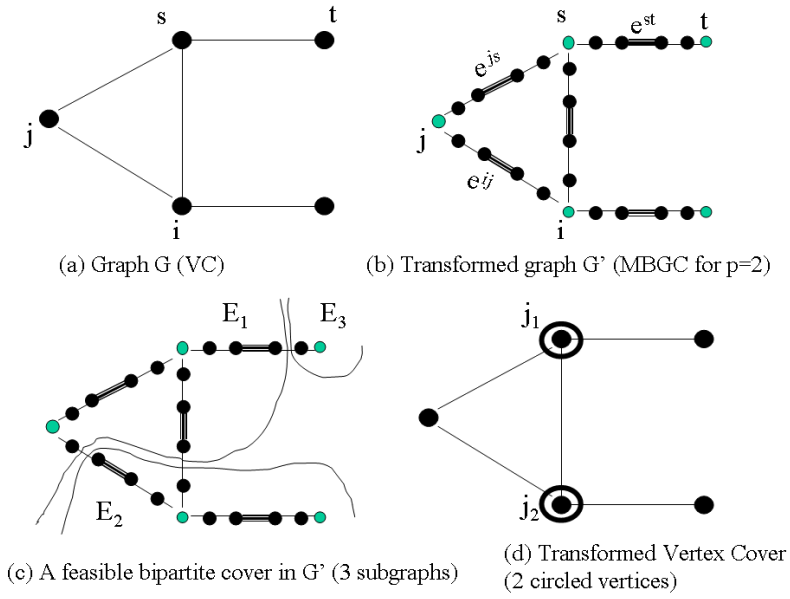

Figure 6: Illustration of a technical point in the proof of Thm. 5.1 ($e^{ij} = (v^{ij}_{ip}, v^{ij}_{jp})$).

(i) Indeed, for $j \in X$, take in the bipartite cover

$$E_j = \bigcup_{\{i,j\} \in E} E'_{ij}$$

As X is a vertex cover in G, $V = X \cup \{i \in V \mid \exists j \in X, \{i,j\} \in E\}$, then $\cup_{j \in X} E_j = E'$ so $\{E_j : j \in X\}$ is indeed a cover of E'. E_j is a tree so it is connected and bipartite. Its length is at most twice the length of the path induced by any edge subset E'_{ij} inside E_j, i.e., at most $2(2p+1) = 4p+2 = L$, so we obtain a L-length bipartite cover of size at most $|X|$.

(ii) Conversely, suppose that $\{H_1, H_2, \ldots, H_m\}$, with $H_k = (A_k, B_k, E_k)$ for $k = 1, \ldots, m$, is a L-length bipartite cover of G'. Let $E'' = \{e^{ij} \mid \{i,j\} \in E\}$, and $K' = \{k \in \{1, \ldots, m\} \mid E_k \cap E'' \neq \emptyset\}$. Since for any edges e^{ij}, e^{js}, e^{st} the length of the unique path starting with the last edge is exactly $3+2(2p) = 4p+3 > L$, then no path in E_k for $k \in K'$ can contain more than 2 edges of E''. So, for $k \in K'$, E_k is a tree (as all cycles in G' contain at least 3 edges of E'') and its restriction to edges of E'' is a star, i.e., there exists a vertex $j_k \in V$ and a subset $I_k \subset V \setminus \{j_k\}$ such that

$$E_k \cap E'' = \{e^{ij_k} \mid i \in I_k\}$$

Then, if we take

$$\{ \bigcup_{(i,j_k) \in E} E'_{i,j_k} \mid k \in K' \}$$

we still have a bipartite cover of size m and length bounded by twice the length of any subpath E'_{i,j_k}, i.e., bounded by $2(2p+1) = L$. We finally obtain that $X = \{j_k \mid k \in K'\}$ is indeed a vertex cover for G of the same size m as the one of the initial bipartite cover for G', which ends the proof. □

We remark that the MCBGC is in **NP** if and only if L is a constant, as computing longest paths in a bipartite graph is **NP**-hard [20].

5.1 Mixed-Integer Linear Programming formulation of the MCBGC problem

We consider the arc set $\mathscr{A} = \{(i,j),(j,i) : (i,j) \in E\}$, i.e. edges are replaced by two antiparallel arcs. Let $\bar{m}$ be an upper bound to m, for example $\bar{m} = \lceil \frac{n}{2} \rceil$. We consider four sets of binary variables and a set of continuous multicommodity flow variables:

$$\begin{aligned}
\forall k \le \bar{m} \quad y_k &= \begin{cases} 1 & \text{if the } k\text{-th bipartite subgraph is in the cover} \\ 0 & \text{otherwise,} \end{cases} \\
\forall i \in V, j \in V, k \le \bar{m} \quad e_{ij}^k &= \begin{cases} 1 & \text{if edge } \{i,j\} \text{ belongs to the } k\text{-th bipartite subgraph} \\ 0 & \text{otherwise,} \end{cases} \\
\forall i \in V, k \le \bar{m} \quad a_i^k &= \begin{cases} 1 & \text{if node } i \text{ is in } A_k \\ 0 & \text{otherwise,} \end{cases} \\
\forall i \in V, k \le \bar{m} \quad b_i^k &= \begin{cases} 1 & \text{if node } i \text{ is in } B_k \\ 0 & \text{otherwise,} \end{cases} \\
\forall (i,j) \in \mathscr{A}, k \le \bar{m}, \qquad & u \in V, v \in V : u \ne v \quad f_{ij}^{uvk} \in [0,1],
\end{aligned}$$

where the continuous flow variables f identify a path connecting u and v in the k-th bipartite subgraph in order to ensure that bipartite subgraphs are connected, i.e., $f_{ij}^{uvk} = 1$ if $\{i,j\} \in E$ and either (i,j) or $(j,i) \in \mathscr{A}$ belongs to the path connecting u and v, 0 otherwise.

5.2 Lemma
The following constraints:

$$a_i^k + b_i^k \le y_k \qquad \forall\, k = 1,\dots,\bar{m}, i \in V \tag{3}$$
$$a_i^k + a_j^k \le 2 - e_{ij}^k \qquad \forall\, k = 1,\dots,\bar{m}, \{i,j\} \in E \tag{4}$$
$$b_i^k + b_j^k \le 2 - e_{ij}^k \qquad \forall\, k = 1,\dots,\bar{m}, \{i,j\} \in E \tag{5}$$
$$a_i^k + b_i^k + a_j^k + b_j^k \ge 2e_{ij}^k \quad \forall\, k = 1,\dots,\bar{m}, \{i,j\} \in E \tag{6}$$

ensure that subgraph $H_k = (A_k, B_k, E_k)$, where $A_k = \{i \in V : a_i^k = 1\}$, $B_k = \{i \in V : b_i^k = 1\}$, $E_k = \{(i,j) \in \mathscr{A} : i < j \wedge e_{ij}^k = 1\}$, is bipartite.

Proof. If subgraph H_k is non-empty then $y_k = 1$, so by constraints (3) every node in subgraph k is either in A^k or B^k but not both. Constraints (4)-(6) ensure that if $e_{ij}^k = 1$, then we have whether $a_i^k = 1, b_i^k = 0, a_j^k = 0, b_j^k = 1$ or $a_i^k = 0, b_i^k = 1, a_j^k = 1, b_j^k = 0$. This eliminates odd-length cycles as these cycles would have an edge $e_{ij}^k = 1$ with $a_i^k = a_j^k = 1$ or $b_i^k = b_j^k = 1$, so subgraph k is bipartite. □

5.3 Lemma
The following constraints:

$$e_{ij}^k \ge f_{ij}^{uvk} \qquad \forall\, k = 1,\dots,\bar{m}, (i,j) \in \mathscr{A}, (u,v) \in V^2 : u \ne v \tag{7}$$
$$e_{ij}^k = e_{ji}^k \qquad \forall; (i,j) \in \mathscr{A} : i < j \tag{8}$$
$$\sum_{j \in V:(u,j) \in \mathscr{A}} f_{uj}^{uvk} \ge a_u^k + b_u^k + a_v^k + b_v^k - 1 \quad \forall\, k = 1,\dots,\bar{m}, (u,v) \in V^2 : u \ne v \tag{9}$$
$$\sum_{i \in V:(i,u) \in \mathscr{A}} f_{uj}^{uvk} = 0 \qquad \forall\, k = 1,\dots,\bar{m}, (u,v) \in V^2 : u \ne v \tag{10}$$
$$\sum_{i \in V:(i,v) \in \mathscr{A}} f_{iv}^{uvk} \ge a_u^k + b_u^k + a_v^k + b_v^k - 1 \quad \forall\, k = 1,\dots,\bar{m}, (u,v) \in V^2 : u \ne v \tag{11}$$
$$\sum_{j \in V:(v,j) \in \mathscr{A}} f_{vj}^{uvk} = 0 \qquad \forall\, k = 1,\dots,\bar{m}, (u,v) \in V^2 : u \ne v \tag{12}$$
$$\sum_{i \in V:(i,j) \in \mathscr{A}} f_{ij}^{uvk} = \sum_{l \in V:(j,l) \in \mathscr{A}} f_{jl}^{uvk} \qquad \forall\, k = 1,\dots,\bar{m}, (u,v,j) \in V^3 : u \ne v \ne j \tag{13}$$

ensure that subgraph H_k is connected.

Proof. We have to show that for every pair of distinct nodes u and v in subgraph k there is a path wholly contained in subgraph k connecting u and v, represented by flow variables f_{ij}^{uvk}. First, by constraints (7), the flow variable associated with an edge is positive only if this edge belongs to the subgraph; so if these flow variables altogether represent a path, this path is contained in subgraph k. Now, if both nodes u and v are in subgraph k then in constraints (9) and (11) the right-hand term $a_u^k + b_u^k + a_v^k + b_v^k - 1$ is 1 (given constraints (3)). So, at least one flow unit leaves u and finally (by flow conservation constraints (13)) arrives at node v, hence defining a path connecting u and v. Constraints (10) and (12) ensure that the flow is not composed of two disconnected cycles, one passing through u and the other one through v. □

Consider now the following MIP formulation:

$$
\begin{aligned}
& \min \sum_{k=1}^{\bar{m}} y_k \\
\text{s.t.} \quad & (3) - (13) \\
& \sum_{(i,j)\in\mathscr{A}} f_{ij}^{uvk} \le L && \forall\, k = 1,\dots,\bar{m}, (u,v) \in V^2 : u \neq v \\
\text{(MIP)} \quad & \sum_{k=1}^{\bar{m}} e_{ij}^k \ge 1 && \forall\, (i,j) \in \mathscr{A} : i < j \\
& y_k, a_i^k, b_i^k, e_{ij}^k \in \{0,1\},\ f_{ij}^{uvk} \ge 0
\end{aligned}
$$

5.4 Theorem
The above Mixed Integer Program (MIP) is a valid formulation for Min-CBGC.

Proof. By Lemmata 5.2 and 5.3, subgraphs $H_k = (A_k, B_k, E_k)$ where $A_k = \{i \in V \mid a_i^k = 1\}$, $B_k = \{i \in V : b_i^k = 1\}$, $E_k = \{(i,j) \in \mathscr{A} : i < j \wedge e_{ij}^k = 1\}$, are connected bipartite subgraphs. The last block of constraints in the above MIP formulation ensures that the length of the path connecting any two distinct vertices u and v in subgraph H_k, represented by flow variables f_{ij}^{uvk}, is at most L. If the connecting path is not a simple path, we can obtain one of smaller length by removing cycles, so its length is smaller than L. The remaining constraints with variables e_{ij}^k are the global covering constraints, ensuring that every edge of E is covered, i.e., belongs to at least one subgraph. By constraints (6), if $A_k = B_k = \emptyset$ then $E_k = \emptyset$ so in that case no edge in E_k contributes to the covering constraints, and $y_k = 0$ by the minimization of the sum of the y-variables. As by constraints (3) $y_k = 1$ as soon as $A_k \neq \emptyset$ or $B_k \neq \emptyset$, the objective function $\sum_{1\le k\le\bar{m}} y_k$ counts indeed the number of non-empty subgraphs in the cover, which completes the proof. □

6 Conclusion

We considered the problem of covering the edges of an undirected graph G by a minimum cardinality set of not necessarily induced (nor complete) bipartite connected subgraphs of G. We proved that this problem is **NP**-hard by reduction from the Minimum Cut Cover problem. We provided an IP formulation for the latter in order to derive bounds for the former. We proposed four greedy constructive heuristics and a local search, all of which are comparatively discussed though some computational results. Finally, we introduced a constrained variant of the problem, proved its **NP**-hardness for appropriate values of a certain parameter, and provided a MILP formulation.

Acknowledgements

We would like to remember Peter Hammer, who posed the problem, and thank Dr. D. Gonçalves for his contribution on the **NP**-completeness proof of the BGC.

References

[1] G. Alexe, S. Alexe, Y. Crama, S. Foldes, P. Hammer, and B. Simeone. Consensus algorithms for the generation of all maximal bicliques. *Discrete Applied Mathematics*, 145:11–21, 2004.

[2] G. Alexe, P.L. Hammer, V.V. Lozin, and D. de Werra. Struction revisited. *Discrete Applied Mathematics*, 132:27–46, 2004.

[3] J. Amilhastre, M.C. Vilarem, and P. Janssen. Complexity of minimum biclique cover and minimum biclique decomposition for bipartite domino-free graphs. *Discrete Applied Mathematics*, 86:125–144, 1998.

[4] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation — Combinatorial optimization problems and their approximability properties*. Springer, New York, 1999.

[5] B. Bollobás. *Modern Graph Theory*. Springer, New York, 1998.

[6] M. Bussieck. The minimal cut cover of a graph. Technical Report TR-94-02, Pennsylvania State University, 1994.

[7] Cornaz and Fonlupt. Chromatic characterizations of biclique covers. *Discrete Mathematics*, 306(5):495-507, 2006.

[8] C. Delorme and S. Poljak. Laplacian eigenvalues and the maximum cut problem. *Mathematical Programming*, 62:557–574, 1993.

[9] D. Eppstein, M.T. Goodrich and J. Yu Meng. Confluent layered drawings *Algorithmica*, 47(4):439-452, 2007.

[10] R. Fourer and D. Gay. *The AMPL Book*. Duxbury Press, Pacific Grove, 2002.

[11] M. Habib, L. Nourine, O. Raynaud, and E. Thierry. Computational aspects of the 2-dimension of partially ordered sets. *Theoretical Computer Science*, 312:401-431, 2004.

[12] B.V. Halldórsson, M.M. Halldórsson, and R. Ravi. On the approximability of the minimum test collection problem. In F. Meyer auf der Heide, editor, *ESA*. LNCS **2161**:158-169. Springer, Heidelberg, 2001.

[13] P. Hammer. The conflict graph of a pseudo-boolean function. Technical report, Bell Labs, West Long Branch, NJ, 1978.

[14] ILOG. *ILOG CPLEX 10.1 User's Manual*. ILOG S.A., Gentilly, France, 2006.

[15] R. Loulou. Minimal cut cover of a graph with an application to the testing of electronic boards. *Operations Research Letters*, 12(5):301–305, 1992.

[16] N. Maculan. Integer programming problems using a polynomial number of variables and constraints for combinatorial optimization problems in graphs. In N. Mladenović and Dj. Dugošija, editors, *SYM-OP-IS Conference Proceedings, Herceg-Novi*, pages 23–26, Beograd, September 2003. Mathematical Institute, Academy of Sciences.

[17] R. Merris. Laplacian matrices of graphs: A survey. *Linear Algebra and its Applications*, 198:143–176, 1994.

[18] R. Motwani and J.S. Naor. On exact and approximate cut covers of graphs. Technical Report STAN-CS-TN-94-11, Dept. of Computer Science, Stanford University, 1994.

[19] H. Müller. On edge perfectness and classes of bipartite graphs. *Discrete Mathematics*, 149:159–187, 1996.

[20] H. Müller. Hamiltonian circuits in chordal bipartite graphs. *Discrete Mathematics*, 156:291–298, 1996.

[21] J. Orlin. Contentment in graph theory: Covering graphs with cliques. *Proceedings of the Koninklijke Nederlandse Akademie van Weteschappen, Series A*, 80(5):406–424, 1977.

[22] M.C. Plateau, L. Liberti, and L. Alfandari. Edge cover by bipartite subgraphs. In J.L. Hurink, W. Kern, G.F. Post, and G.J. Still, editors, *Proceedings of the 6th Cologne-Twente Workshop on Graphs and Combinatorial Optimization*, Enschede, 2007. University of Twente.

[23] K.H. Rosen, editor. *Handbook of Discrete and Combinatorial Mathematics*. CRC Press, New York, 2000.