Hitting Diamonds and Growing Cacti[⋆]

Samuel Fiorini[1], Gwenaël Joret[2,⋆⋆], and Ugo Pietropaoli[3,⋆⋆⋆]

[1] Université Libre de Bruxelles (ULB), Département de Mathématique, CP 216, B-1050 Brussels, Belgium
sfiorini@ulb.ac.be

[2] Université Libre de Bruxelles (ULB), Département d'Informatique, CP 212, B-1050 Brussels, Belgium
gjoret@ulb.ac.be

[3] Università di Roma "Tor Vergata", Dipartimento di Ingegneria dell'Impresa, Rome, Italy
pietropaoli@disp.uniroma2.it

Abstract. We consider the following NP-hard problem: in a weighted graph, find a minimum cost set of vertices whose removal leaves a graph in which no two cycles share an edge. We obtain a constant-factor approximation algorithm, based on the primal-dual method. Moreover, we show that the integrality gap of the natural LP relaxation of the problem is $\Theta(\log n)$, where n denotes the number of vertices in the graph.

1 Introduction

Graphs in this paper are finite, undirected, and may contain parallel edges but no loops. We study the following combinatorial optimization problem: given a vertex-weighted graph, remove a minimum cost subset of vertices so that all the cycles in the resulting graph are edge-disjoint. We call this problem the *diamond hitting set problem*, because it is equivalent to covering all subgraphs which are diamonds with a minimum cost subset of vertices, where a *diamond* is any subdivision of the graph consisting of three parallel edges.

The diamond hitting set problem can be thought of as a generalization of the vertex cover and feedback vertex set problems: Suppose you wish to remove a minimum cost subset of vertices so that the resulting graph has no pair of vertices linked by k internally disjoint paths. Then, for $k = 1$ and $k = 2$, this is respectively the vertex cover problem and feedback vertex set problem, while for $k = 3$ this corresponds to the diamond hitting set problem.

[⋆] This work was supported by the "Actions de Recherche Concertées" (ARC) fund of the "Communauté française de Belgique".

[⋆⋆] G. Joret is a Postdoctoral Researcher of the Fonds National de la Recherche Scientifique (F.R.S.–FNRS).

[⋆⋆⋆] This work was done while U.P. was at Département de Mathématique - Université Libre de Bruxelles as a Postdoctoral Researcher of the F.R.S.–FNRS.

© Springer-Verlag Berlin Heidelberg 2010

It is well-known that both the vertex cover and feedback vertex set problems admit constant-factor approximation algorithms[1]. Hence, it is natural to ask whether the same is true for the diamond hitting set problem. Our main contribution is a positive answer to this question.

1.1 Background and Related Work

Although there exists a simple 2-approximation algorithm for the vertex cover problem, there is strong evidence that approximating the problem with a factor of $2 - \varepsilon$ might be hard, for every $\varepsilon > 0$ [8]. It should be noted that the feedback vertex set and diamond hitting set problems are at least as hard to approximate as the vertex cover problem, in the sense that the existence of a ρ-approximation algorithm for one of these two problems implies the existence of a ρ-approximation algorithm for the vertex cover problem, where ρ is a constant.

Concerning the feedback vertex set problem, the first approximation algorithm is due to Bar-Yehuda, Geiger, Naor, and Roth [2] and its approximation factor is $O(\log n)$. Later, 2-approximation algorithms have been proposed by Bafna, Berman, and Fujito [1], and Becker and Geiger [3]. Chudak, Goemans, Hochbaum and Williamson [4] showed that these algorithms can be seen as deriving from the primal-dual method (see for instance [9,7]). Starting with an integer programming formulation of the problem, these algorithms simultaneously construct a feasible integral solution and a feasible dual solution of the linear programming relaxation, such that the values of these two solutions are within a constant factor of each other.

These algorithms also lead to a characterization of the integrality gap[2] of two different integer programming formulations of the problem, as we now explain. Let $\mathcal{C}(G)$ denote the collection of all the cycles C of G. A natural integer programming formulation for the feedback vertex set problem is as follows:

$$\begin{aligned} \text{Min} \quad & \sum_{v \in V(G)} c_v\, x_v \\ \text{s.t.} \quad & \sum_{v \in V(C)} x_v \geqslant 1 \qquad && \forall C \in \mathcal{C}(G) \\ & x_v \in \{0,1\} && \forall v \in V(G). \end{aligned} \tag{1}$$

(Throughout, c_v denotes the (non-negative) cost of vertex v.) The algorithm of Bar-Yehuda et al. [2] implies that the integrality gap of this integer program is $O(\log n)$. Later, Even, Naor, Schieber, and Zosin [5] proved that its integrality gap is also $\Omega(\log n)$.

[1] A ρ-approximation algorithm for a minimization problem is an algorithm that runs in polynomial time and outputs a feasible solution whose cost is no more than ρ times the cost of the optimal solution. The number ρ is called the approximation factor.

[2] The integrality gap of an integer programming formulation is the worst-case ratio between the optimum value of the integer program and the optimum value of its linear relaxation.

A better formulation has been introduced by Chudak et al. [4]. For $S \subseteq V(G)$, denote by $E(S)$ the set of the edges of G having both ends in S, by $G[S]$ the subgraph of G induced by S, and by $d_S(v)$ the degree of v in $G[S]$. Then, the following is a formulation for the feedback vertex set problem:

$$\begin{aligned} \text{Min} \quad & \sum_{v \in V(G)} c_v\, x_v \\ \text{s.t.} \quad & \sum_{v \in S} (d_S(v) - 1) x_v \geqslant |E(S)| - |S| + 1 \qquad && \forall S \subseteq V(G) : E(S) \neq \varnothing \qquad (2) \\ & x_v \in \{0, 1\} && \forall v \in V(G). \end{aligned}$$

Chudak et al. [4] showed that the integrality gap of this integer program asymptotically equals 2. Constraints (2) derive from the simple observation that the removal of a feedback vertex set X from G generates a forest having at most $|G| - |X| - 1$ edges. Notice that the covering inequalities (1) are implied by (2).

1.2 Contribution and Key Ideas

First, we obtain a $O(\log n)$-approximation algorithm for the diamond hitting set problem, leading to a proof that the integrality gap of the natural LP formulation is $\Theta(\log n)$. Then, we develop a 9-approximation algorithm. Both the $O(\log n)$- and 9-approximation algorithm are based on the primal-dual method.

Our first key idea is contained in the following observation: every simple graph of order n and minimum degree at least 3 contains a $O(\log n)$-size diamond. This directly yields a $O(\log n)$-approximation algorithm for the diamond hitting set problem, in the unweighted case. However, the weighted case requires more work.

Our second key idea is to generalize constraints (2) by introducing ‘sparsity inequalities’, that enable us to derive a constant-factor approximation algorithm for the diamond hitting set problem: First, by using reduction operations, we ensure that every vertex of G has at least three neighbors. Then, if G contains a diamond with at most 9 edges, we raise the dual variable of the corresponding covering constraint. Otherwise, no such small diamond exists in G, and we can use this information to select the right sparsity inequality, and raise its dual variable. This inequality would not be valid in case G contained a small diamond.

The way we use the non-existence of small diamonds is perhaps best explained via an analogy with planar graphs: An n-vertex planar simple graph G has at most $3n - 6$ edges. However, if we know that G has no small cycle, then this upper bound can be much strengthened. (For instance, if G is triangle-free then G has at most $2n - 4$ edges.)

We remark that this kind of local/global trade-off did not appear in the work of Chudak et al. [4] on the feedback vertex set problem, because the cycle covering inequalities are implied by their more general inequalities. In our case, the covering inequalities and the sparsity inequalities form two incomparable classes of inequalities, and examples show that the sparsity inequalities alone are not enough to derive a constant-factor approximation algorithm.

This extended abstract is organized as follows. Preliminaries are given in Section 2. Then, in Section 3, we define some reduction operations that allow us to work mainly with graphs where each vertex has at least three distinct neighbors. Next, in Section 4, we deal with the unweighted version of the diamond hitting set problem and provide a simple $O(\log n)$-approximation algorithm. In Section 5, we turn to the weighted version of the problem. We sketch a $O(\log n)$-approximation algorithm. It turns out that the integrality gap of the natural formulation of the problem is $\Theta(\log n)$. Finally, in Section 6, we introduce the sparsity inequalities, prove their validity and sketch a 9-approximation algorithm. Due to length restrictions, most proofs and details are not included in this extended abstract. A full version of the paper can be found in [6].

2 Preliminaries

A *cactus* is a connected graph where each edge belongs to at most one cycle. Equivalently, a connected graph is a cactus if and only if each of its blocks is isomorphic to either K_2 or a cycle. Thus, a connected graph is a cactus if and only if it does not contain a diamond as a subgraph. A graph without diamonds is called a *forest of cacti* (see Figure 1).

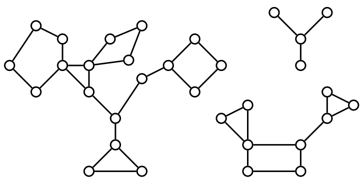

Fig. 1. A forest of cacti

A *diamond hitting set* (or simply *hitting set*) of a graph is a subset of vertices that hits every diamond of the graph. A *minimum (diamond) hitting set* of a weighted graph is a hitting set of minimum total cost, and its cost is denoted by OPT.

Let $\mathcal{D}(G)$ denote the collection of all diamonds contained in G. From the standard IP formulation for a covering problem, we obtain the following LP relaxation for the diamond hitting set problem:

$$\begin{aligned} \text{Min} \quad & \sum_{v \in V(G)} c_v \, x_v & & \\ \text{s.t.} \quad & \sum_{v \in V(D)} x_v \geqslant 1 & \quad & \forall D \in \mathcal{D}(G) \qquad (3) \\ & x_v \geqslant 0 & & \forall v \in V(G). \end{aligned}$$

We call inequalities (3) *diamond inequalities*.

3 Reductions

In this section, we define two reduction operations on graphs: First, we define the 'shaving' of an arbitrary graph, and then introduce a 'bond reduction' operation for shaved graphs.

The aim of these two operations is to modify a given graph so that the following useful property holds: each vertex either has at least three distinct neighbors, or is incident to at least three parallel edges.

3.1 Shaving a Graph

Let G be a graph. Every block of G is either isomorphic to K_1, K_2, a cycle, or contains a diamond. Mark every vertex of G that is included in a block containing a diamond. The *shaving* of G is the graph obtained by removing every unmarked vertex from G. A graph is *shaved* if all its vertices belong to a block containing a diamond. Observe that, in particular, every endblock[3] of a shaved graph contains a diamond.

3.2 Reducing a Bond

A *bond* of a graph G is a connected subgraph $Q \subseteq G$ equipped with two distinguished vertices v, w (called *ends*) satisfying the following requirements:

– Q is a cactus with at least two blocks;
– the block-graph of Q is a path;
– v and w belong to distinct endblocks of Q;
– v and w are not adjacent in Q;
– $Q - \{v, w\}$ is a non-empty component of $G - \{v, w\}$, and
– Q contains all the edges in G between $\{v, w\}$ and $V(Q) - \{v, w\}$.

Observe that Q is "almost" an induced subgraph of G, since Q includes every edge of G between vertices of Q, except those between v and w (if any). The vertices in $V(Q) - \{v, w\}$ are said to be the *internal vertices* of Q. The bond Q is *simple* if Q is isomorphic to a path, *double* otherwise.

Let G be a shaved graph. A vertex u of G is *reducible* if u has exactly two neighbors in G, and there are at most two parallel edges connecting u to each of its neighbors. The *bond reduction* operation is defined as follows. Let u be a reducible vertex and let Q_u be an inclusion-wise maximal bond of G containing u, with ends v and w. (Observe that such a bond exists by our hypothesis on u; moreover, it might not be unique.) Then, remove from G every internal vertex of Q_u, and add one or two edges between v and w, depending on whether Q_u is simple or double. In the latter case, the two new parallel edges are said to be *twins*. See Figure 2 for an illustration of the operation. Observe that the resulting graph is also a shaved graph.

[3] We recall that the block-graph of G has the blocks of G and the cutvertices of G as vertices, a block and a cutvertex are adjacent if the former contains the latter. This graph is always acyclic. An endblock of G is a vertex of the block-graph with degree at most one.

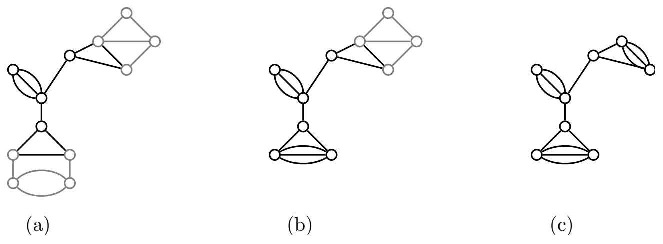

Fig. 2. (a) A shaved graph G with two maximal bonds (in grey). (b) Reduction of the first bond. (c) Reduction of the second bond. The graph is now reduced.

A crucial property of the bond reduction operation is that, when applying it iteratively, we never include in the bond to be reduced any edge coming from previous bond reductions [6, Lemma 3.1].

A *reduced graph* $\widetilde{G}$ of G is any graph obtained from G by iteratively applying a bond reduction, as long as there is a reducible vertex (see Figure 2). We remark that there is not necessarily a unique reduced graph of G (consider for instance K_3 where two edges are doubled).

4 A $O(\log n)$-Approximation Algorithm in the Unweighted Case

As a first step, we show that every reduced graph contains a diamond of size $O(\log n)$ [6, Lemmas 4.1 and 4.4].

Lemma 1. *Every simple n-vertex graph with minimum degree at least 3 contains a diamond of size at most $6\log_{3/2} n + 8$. Moreover, such a diamond can be found in polynomial time. The same holds for reduced graphs.*

Our algorithm for the diamond hitting set problem on unweighted graphs is described in Algorithm 1.

Algorithm 1. A $O(\log n)$-approximation algorithm for unweighted graphs.

```
– $X \leftarrow \varnothing$
– While $X$ is not a hitting set of $G$, repeat the following steps:
    • Compute a reduced graph $\widetilde{G}$ of $G - X$
    • Find a diamond $\widetilde{D}$ in $\widetilde{G}$ of size at most $6\log_{3/2}|\widetilde{G}| + 8$      (using Lemma 1)
    • Include in $X$ all vertices of $\widetilde{D}$
```

The algorithm relies on the simple fact that every hitting set of a reduced graph $\widetilde{G}$ of a graph G is also a hitting set of G itself. The set of diamonds computed by the algorithm yields a collection $\mathcal{D}$ of pairwise vertex-disjoint diamonds in G. In particular, the size of a minimum hitting set is at least $|\mathcal{D}|$. For each diamond in $\mathcal{D}$, at most $6\log_{3/2} n + 8$ vertices were added to the hitting set X. Hence, the approximation factor of the algorithm is $O(\log n)$.

5 A $O(\log n)$-Approximation Algorithm

The present section is devoted to a $O(\log n)$-approximation algorithm for the diamond hitting set problem in the weighted case, which is based on the primal-dual method. We start by defining, in Section 5.1, the actual LP relaxation of the problem used by the algorithm, together with its dual. Then, in Section 5.2, we sketch our approximation algorithm. Details and proofs can be found in the full version of this paper [6, Section 5].

5.1 The Working LP and Its Dual

Our approximation algorithm for the weighted case is based on the natural LP relaxation for the diamond hitting set problem, given on page 194. To simplify the presentation, we do not directly resort to that LP relaxation but to a possibly weaker relaxation that is constructed during the execution of the algorithm, that we call the *working LP*. At each iteration, an inequality is added to the working LP. These inequalities, that we name *blended diamond inequalities*, are all implied by diamond inequalities (3). The final working LP reads:

$$\begin{array}{llll} \text{(LP)} & \text{Min} & \displaystyle\sum_{v \in V(G)} c_v\, x_v & \\ & \text{s.t.} & \displaystyle\sum_{v \in V(G)} a_{i,v}\, x_v \geqslant \beta_i & \forall i \in \{1, \ldots, k\} \\ & & x_v \geqslant 0 & \forall v \in V(G), \end{array}$$

where k is the total number of iterations of the algorithm. The dual of (LP) is:

$$\begin{array}{llll} \text{(D)} & \text{Max} & \displaystyle\sum_{i=1}^{k} \beta_i\, y_i & \\ & \text{s.t.} & \displaystyle\sum_{i=1}^{k} a_{i,v}\, y_i \leqslant c_v & \forall v \in V(G) \\ & & y_i \geqslant 0 & \forall i \in \{1, \ldots, k\}. \end{array}$$

The algorithm is based on the primal-dual method. It maintains a boolean primal solution x and a feasible dual solution y. Initially, all variables are set to 0. Then the algorithm enters its main loop, that ends when x satisfies all diamond inequalities. At the ith iteration, a violated inequality $\sum_{v \in V} a_{i,v}\, x_v \geqslant \beta_i$ is added to the working LP and the corresponding dual variable y_i is increased. In order to preserve the feasibility of the dual solution, we stop increasing y_i whenever some dual inequality becomes tight. That is, we stop increasing when $\sum_{j=1}^{i} a_{j,v}\, y_j = c_v$ for some vertex v, that is said to be *tight*. (Actually, we should also stop increasing y_i in case a 'collision' occurs [6, Section 5.2.4].) All tight vertices v (if any) are then added to the primal solution. That is, the corresponding variables x_v are increased from 0 to 1. The current iteration then ends and we check whether x satisfies all diamond inequalities. If so, then we exit the loop, perform a reverse delete step, and output the current primal solution.

The precise way the violated blended diamond inequality is chosen depends among other things on the *residual cost* (or slack) of the vertices. The residual cost of vertex v at the ith iteration is the number $c_v - \sum_{j=1}^{i-1} a_{j,v}\, y_j$. Note that the residual cost of a vertex is always nonnegative, and zero if and only if the vertex is tight.

5.2 The Algorithm

The algorithm (rather, a simplified version of the algorithm) is described in Algorithm 2.

Algorithm 2. A $O(\log n)$-approximation algorithm for weighted graphs.

```
– $X \leftarrow \varnothing$;   $y \leftarrow 0$;   $i \leftarrow 0$
– While $X$ is not a hitting set of $G = (V, E)$, repeat the following steps:
    • $i \leftarrow i + 1$
    • Let $H$ be the graph obtained by shaving $G - X$
    • Find a reduced graph $\widetilde{H}$ of $H$
    • Find a diamond $\widetilde{D}$ in $\widetilde{H}$ of size at most $6 \log_{3/2} |\widetilde{H}| + 8$   (using Lemma 1)
    • Compute a violated blended diamond inequality $\sum_{v \in V} a_{i,v}\, x_v \geqslant \beta_i$
      based on $\widetilde{D}$, and the residual costs; add it to (LP)
    • Increase $y_i$ until some vertex becomes tight
    • Add all tight vertices $v$ to $X$, in a carefully chosen order [6, Section 5.2.6]
– $k \leftarrow i$
– Perform a reverse delete step on $X$
```

We remark that the set X naturally corresponds to a primal solution x, obtained by setting x_v to 1 if $v \in X$, to 0 otherwise, for every $v \in V(G)$. This vector x satisfies the diamond inequalities (3) exactly when we exit the *while* loop of the algorithm, that is, when X becomes a hitting set.

The reverse delete step consists in considering the vertices of X in the reverse order in which they were added to X and deleting those vertices v such that $X - \{v\}$ is still a hitting set. Observe that, because of this step, the hitting set X output by the algorithm is inclusion-wise minimal.

Theorem 1. *Algorithm 2 yields a $O(\log n)$-approximation for the diamond hitting set problem.*

We can prove that the integrality gap of the natural LP relaxation for the problem (see page 194) is $\Theta(\log n)$. This result is obtained using expander graphs with large girth [6, Section 5.4].

6 A 9-Approximation Algorithm

In this section we give a primal-dual 9-approximation algorithm for the diamond hitting set problem. We start with a sketch of the algorithm in Section 6.1. The algorithm makes use of the sparsity inequalities. In order to describe them, we first bound the number of edges in a forest of cacti in Section 6.2; using this bound, in Sections 6.3 and 6.4 we introduce the sparsity inequalities and prove their validity. Once again, missing proofs and details can be found in the full version of this paper [6, Section 6].

In the whole section, q is a global parameter of our algorithm which is set to 5. (We remark that the analysis below could be adapted to other values of q, but this would not give an approximation factor better than 9.)

6.1 The Algorithm

Our 9-approximation algorithm for the diamond hitting set problem is very similar to the $O(\log n)$-approximation algorithm. The main difference is that we use a larger set of inequalities to build the working LP relaxation. See Algorithm 3 for a description of (a simplified version of) the algorithm.

Algorithm 3. A 9-approximation algorithm.

```
– $X \leftarrow \varnothing$;   $y \leftarrow 0$;   $i \leftarrow 0$
– While $X$ is not a hitting set of $G = (V, E)$, repeat the following steps:
    • $i \leftarrow i + 1$
    • Let $H$ be the graph obtained by shaving $G - X$
    • Find a reduced graph $\widetilde{H}$ of $H$
    • If, in $\widetilde{H}$, no two cycles of size at most $q$ share an edge then let
      $\sum_{v \in V} a_{i,v}\, x_v \geqslant \beta_i$ be the extended sparsity inequality with support $V(H)$
    • Otherwise, $\widetilde{H}$ contains a diamond $\widetilde{D}$ with at most $2q - 1$ edges and let
      $\sum_{v \in V} a_{i,v}\, x_v \geqslant \beta_i$ be a blended diamond inequality deduced from $\widetilde{D}$ and
      the residual costs
    • Add the inequality $\sum_{v \in V} a_{i,v}\, x_v \geqslant \beta_i$ to the working LP
    • Increase $y_i$ until some vertex becomes tight
    • Add all tight vertices to $X$, in a carefully chosen order
– $k \leftarrow i$
– Perform a reverse delete step on $X$
```

Our main result is as follows.

Theorem 2. *Algorithm 3 yields a 9-approximation for the diamond hitting set problem.*

6.2 Bounding the Number of Edges in a Forest of Cacti

The following lemma provides a bound on the number of edges in a forest of cacti. For $i \in \{2, \ldots, q\}$, we denote by $\gamma_i(G)$ the number of cycles of length i of a graph G.

Lemma 2. *Let F be a forest of cacti with k components and let $q \geqslant 2$. Then*

$$||F|| \leqslant \frac{q+1}{q}(|F|-k) + \sum_{i=2}^{q} \frac{q-i+1}{q}\,\gamma_i(F).$$

Proof. Denote by $\gamma_{>q}(F)$ the number of cycles of F whose length exceeds q. We have

$$||F|| = |F| - k + \sum_{i=2}^{q} \gamma_i(F) + \gamma_{>q}(F). \tag{4}$$

In the right hand side, the first two terms represent the number of edges in a spanning forest of F, while the last terms give the number of edges that should be added to obtain the forest of cacti F.

Because every two cycles in F are edge disjoint, we have

$$||F|| \geqslant \sum_{i=2}^{q} i\,\gamma_i(F) + (q+1)\,\gamma_{>q}(F).$$

Combining this with (4), we get

$$\gamma_{>q}(F) \leqslant \frac{1}{q}\Big(|F| - k - \sum_{i=2}^{q}(i-1)\gamma_i(F)\Big). \tag{5}$$

From (4) and (5), we finally infer

$$\begin{aligned}||F|| &\leqslant |F| - k + \sum_{i=2}^{q}\gamma_i(F) + \frac{1}{q}\Big(|F| - k - \sum_{i=2}^{q}(i-1)\,\gamma_i(F)\Big)\\ &\leqslant \frac{q+1}{q}(|F|-k) + \sum_{i=2}^{q}\frac{q-i+1}{q}\,\gamma_i(F). \qquad \square\end{aligned}$$

6.3 The Sparsity Inequalities

We define the *load* of a vertex v in a graph G as

$$\ell_G(v) := d_G(v) - \sum_{i=2}^{q} \lambda_i\,\gamma_i(G,v),$$

where, for $i \in \{2,\dots,q\}$, $\gamma_i(G,v)$ denotes the number of cycles of length i incident to v in G and

$$\lambda_i := \frac{q-i+1}{\lfloor i/2 \rfloor\, q}.$$

Lemma 3. *Let X be a hitting set of a graph G where no two cycles of length at most q share an edge. Then,*

$$\sum_{v\in X}\Big(\ell_G(v) - \frac{q+1}{q}\Big) \geqslant ||G|| - \frac{q+1}{q}|G| - \sum_{i=2}^{q}\frac{q-i+1}{q}\,\gamma_i(G) + \frac{q+1}{q}. \tag{6}$$

We call Inequality (6) a *sparsity inequality.*

Proof. For $i \in \{2, \dots, q\}$ and $j \in \{0, \dots, i\}$, we denote by ξ_i^j the number of cycles of G that have length i and exactly j vertices in X.

Letting $||X||$ and $|\delta(X)|$ respectively denote the number of edges of G with both ends in X and the number of edges of G having an end in X and the other in $V(G) - X$, we have

$$\begin{aligned}\sum_{v\in X} \ell_G(v) &= 2||X|| + |\delta(X)| - \sum_{i=2}^{q}\sum_{j=1}^{i} j\,\lambda_i\,\xi_i^j \\ &= ||X|| + ||G|| - ||G - X|| - \sum_{i=2}^{q}\sum_{j=1}^{i} j\,\lambda_i\,\xi_i^j \\ &\geqslant ||X|| + ||G|| - \frac{q+1}{q}(|G - X| - 1) - \sum_{i=2}^{q} \frac{q-i+1}{q}\,\xi_i^0 - \sum_{i=2}^{q}\sum_{j=1}^{i} j\,\lambda_i\,\xi_i^j,\end{aligned}$$

where the last inequality follows from Lemma 2 applied to the forest of cacti $G - X$ (notice that $\gamma_i(G - X) = \xi_i^0$).

Because no two cycles of length at most q share an edge and, in a cycle of length i, each subset of size j induces a subgraph that contains at least $2j - i$ edges, we have

$$||X|| \geqslant \sum_{i=2}^{q}\sum_{j=1+\lfloor \frac{i}{2} \rfloor}^{i} (2j - i)\,\xi_i^j.$$

Thus, we obtain

$$\begin{aligned}\sum_{v\in X} \ell_G(v) \geqslant \sum_{i=2}^{q}\sum_{j=1+\lfloor \frac{i}{2} \rfloor}^{i} (2j - i)\,\xi_i^j + ||G|| - \frac{q+1}{q}(|G - X| - 1) \\ - \sum_{i=2}^{q} \frac{q-i+1}{q}\,\xi_i^0 - \sum_{i=2}^{q}\sum_{j=1}^{i} j\,\lambda_i\,\xi_i^j.\end{aligned}$$

We leave it to the reader to check that, in the right hand side of the last inequality, the total coefficient of ξ_i^j is at least $-\frac{q-i+1}{q}$, for all $i \in \{2, \dots, q\}$ and $j \in \{0, \dots, i\}$. Hence,

$$\sum_{v\in X} \ell_G(v) \geqslant ||G|| - \frac{q+1}{q}|G| + \frac{q+1}{q}|X| + \frac{q+1}{q} - \sum_{i=2}^{q} \frac{q-i+1}{q}\,\gamma_i(G).$$

Inequality (6) follows. □

6.4 The Extended Sparsity Inequalities

Consider a shaved graph H and denote by $\widetilde{H}$ a reduced graph of H. Suppose that, in $\widetilde{H}$, no two cycles of length at most q share an edge. The sparsity inequality (6) for $\widetilde{H}$ yields a valid inequality also for H, where the coefficient of each variable corresponding to a vertex that was removed when H is zero. However, as it is, the inequality is useless. We have to raise the coefficients to those variables in such a way that the resulting inequality remains valid.

First, we associate to each edge of $\widetilde{H}$ a corresponding *primitive subgraph* in H, defined as follows. Consider an edge $e \in E(\widetilde{H})$. If e was already present in H, then its primitive subgraph is the edge itself and its two ends. Otherwise, the primitive subgraph of e is the bond whose reduction produced e. In particular, if e has a twin edge e', then the primitive subgraphs of e and e' coincide. The primitive subgraph J of a subgraph $\widetilde{J} \subseteq \widetilde{H}$ is defined simply as the union of the primitive subgraphs of every edge in $E(\widetilde{J})$.

Thus, the graph H can be uniquely decomposed into simple or double pieces corresponding respectively to edges or pairs of parallel edges in $\widetilde{H}$. Here, the pieces of H are defined as follows: let v and w be two adjacent vertices of $\widetilde{H}$, and let $\widetilde{J}$ denote the subgraph of $\widetilde{H}$ induced by $\{v, w\}$ (thus $\widetilde{J}$ is either an edge or a pair of parallel edges, together with the endpoints v and w). The primitive subgraph of $\widetilde{J}$ in H, say J, is a *piece* of H with ends v and w. Such a piece is *simple* if $\widetilde{J}$ has exactly one edge and *double* otherwise (see Fig. 3), that is, if $\widetilde{J}$ has two parallel edges. The vertices of H are of two types: the branch vertices are those that survive in $\widetilde{H}$, and the other vertices are internal to some piece of H.

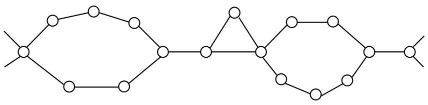

Fig. 3. A double piece

Consider a double piece Q of H (if any) and a cycle C contained in Q. Then C is a block of Q. A vertex v of C is said to be an *end* of the cycle C if v is an end of the piece Q or v belongs to a block of Q distinct from C. Observe that C has always two distinct ends. The cycle C has also two *handles*, defined as the two v–w paths in C, where v and w are the two ends of C.

The two handles of C are labelled *top* and *bottom* as described in [6, Section 6]. Denote by $t(C)$ (resp. $b(C)$) the minimum residual cost of a vertex in the top handle (resp. bottom handle) of C. Thus, $t(C) \leqslant b(C)$. Also, we choose a cycle of Q (if any) and declare it to be *special*. If possible, the special cycle is chosen among the cycles C contained in Q with $t(C) = b(C)$. (So *every* double piece of H has a special cycle.)

The *extended sparsity inequality* for H reads

$$\sum_{v \in V(H)} a_v x_v \geqslant \beta, \tag{7}$$

where

$$a_v := \begin{cases} \ell_{\widetilde{H}}(v) - \dfrac{q+1}{q} & \text{if } v \text{ is a branch vertex,} \\ 1 & \text{if } v \text{ is an internal vertex of a simple piece,} \\ 2 & \text{if } v \text{ is an internal vertex of a double piece} \\ & \text{and does not belong to any handle,} \\ 0 & \text{if } v \text{ belongs to the top handle} \\ & \text{of a cycle } C \text{ with } t(C) < b(C), \\ 2 & \text{if } v \text{ belongs to the bottom handle} \\ & \text{of a cycle } C \text{ with } t(C) < b(C), \\ 1 & \text{if } v \text{ belongs to a handle of a cycle } C \text{ with } t(C) = b(C) \\ & \text{or } C \text{ is special,} \end{cases}$$

and

$$\beta := ||\widetilde{H}|| - \frac{q+1}{q}|\widetilde{H}| - \sum_{i=2}^{q} \frac{q-i+1}{q}\gamma_i(\widetilde{H}) + \frac{q+1}{q}.$$

In the definition of a_v above, we always assume that the cycle C is contained in a double piece of H. The next lemma is proved in [6, Lemma 6.3].

Lemma 4. *Let H be a graph and let $\widetilde{H}$ be a reduced graph of H such that no two cycles of length at most q share an edge. Then Inequality* (7) *is valid, that is,*

$$\sum_{v\in X} a_v \geqslant \beta$$

whenever X is a hitting set of H.

The following lemma is the key of our analysis of Algorithm 3. Notice that the constant involved is smaller than the approximation guarantee of our algorithm. The number 9 comes from the fact that we use blended diamond inequalities for diamonds with up to 9 pieces.

Lemma 5. *Let H be a shaved graph and let $\widetilde{H}$ be a reduced graph of H. Suppose that, in $\widetilde{H}$, no two cycles of length at most q share an edge. Then,*

$$\sum_{v\in X} a_v \leqslant 8\,\beta$$

for every minimal hitting set X of H.

Acknowledgements

We thank Dirk Oliver Theis for his valuable input in the early stage of this research. We also thank Jean Cardinal and Marcin Kamiński for stimulating discussions.

References

1. Bafna, V., Berman, P., Fujito, T.: A 2-approximation algorithm for the undirected feedback vertex set problem. SIAM Journal on Discrete Mathematics 12(3), 289–297 (1999)
2. Bar-Yehuda, R., Geiger, D., Naor, J., Roth, R.M.: Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference. SIAM Journal on Computing 27(4), 942–959 (1998)
3. Becker, A., Geiger, D.: Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. Artificial Intelligence 83, 167–188 (1996)
4. Chudak, F.A., Goemans, M.X., Hochbaum, D.S., Williamson, D.P.: A primaldual interpretation of two 2-approximation algorithms for the feedback vertex set problem in undirected graphs. Operations Research Letters 22, 111–118 (1998)
5. Even, G., Naor, J., Schieber, B., Zosin, L.: Approximating minimum subset feedback sets in undirected graphs with applications. SIAM Journal on Discrete Mathematics 13(2), 255–267 (2000)
6. Fiorini, S., Joret, G., Pietropaoli, U.: Hitting diamonds and growing cacti (2010), `http://arxiv.org/abs/0911.4366v2`
7. Goemans, M.X., Williamson, D.P.: The primal-dual method for approximation algorithms and its application to network design problems. In: Approximation Algorithms for NP-Hard Problems, ch. 4, pp. 144–191. PWS Publishing Company (1997)
8. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within $2 - \varepsilon$. Journal of Computer and System Sciences 74(3), 334–349 (2008)
9. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Englewood Cliffs (1982)