Two Approximation Algorithms for 3-Cycle Covers

Markus Bläser and Bodo Manthey*

Institut für Theoretische Informatik
Universität zu Lübeck
Wallstraße 40, 23560 Lübeck, Germany
blaeser/manthey@tcs.mu-luebeck.de

Abstract. A cycle cover of a directed graph is a collection of node disjoint cycles such that every node is part of exactly one cycle. A k-cycle cover is a cycle cover in which every cycle has length at least k. While deciding whether a directed graph has a 2-cycle cover is solvable in polynomial time, deciding whether it has a 3-cycle cover is already NP-complete. Given a directed graph with nonnegative edge weights, a maximum weight 2-cycle cover can be computed in polynomial time, too. We call the corresponding optimization problem of finding a maximum weight 3-cycle cover Max-3-DCC.
In this paper we present two polynomial time approximation algorithms for Max-3-DCC. The heavier of the 3-cycle covers computed by these algorithms has at least a fraction of $\frac{3}{5} - \epsilon$, for any $\epsilon > 0$, of the weight of a maximum weight 3-cycle cover.
As a lower bound, we prove that Max-3-DCC is APX-complete, even if the weights fulfil the triangle inequality.

1 Introduction

A cycle cover of a directed or undirected graph G is a spanning subgraph consisting of node-disjoint cycles. (In the case of undirected graphs, cycle covers are also called 2-factors.) Cycle covers have been intensively studied for many decades, see e.g. Lovász and Plummer [12] and Graham et al. [8] and the abundance of references given there.

A k-cycle cover is a cycle cover in which each cycle has at least length k. Such cycle covers are also called $(k-1)$-restricted. In this paper, we are concerned with approximating maximum weight 3-cycle covers in complete directed graphs with nonnegative edge weights. To be specific, we call this problem Max-3-DCC. As our main contribution, we devise approximation algorithms for Max-3-DCC. On the other hand, we show that Max-3-DCC is APX-complete.

1.1 Previous Results

The problem of deciding whether an unweighted directed graph has a 2-cycle cover can be solved in polynomial time by computing a maximum bipartite

* Birth name: Bodo Siebert. Supported by DFG research grant Re 672/3.

matching. The corresponding optimization problem Max-2-DCC is polynomial time computable, too. On the other hand, deciding whether an unweighted directed graph has a 3-cycle cover is already NP-complete. This follows from the work of Valiant [16] (see also Garey and Johnson [7, GT 13]). Thus, considering optimization problems, Max-3-DCC is the next interesting problem. If the given graph has only edge weights zero and one, then there is a $\frac{2}{3}$-approximation algorithm for finding a maximum weight 3-cycle cover. This algorithm can be obtained from the algorithm presented by Bläser and Siebert [3] for the minimum weight 3-cycle cover problem with weights one and two by replacing weight zero with weight two.

What is known for undirected graphs? The problem of finding a 3-cycle cover in undirected graphs can be solved in polynomial time by Tutte's reduction [15] to the classical perfect matching problem in undirected graphs. The classical perfect matching problem can be solved in polynomial time (see Edmonds [5]). The corresponding maximization problem can be solved in polynomial time, even if we allow arbitrary nonnegative weights. Hartvigsen [9] has designed a powerful polynomial time algorithm for deciding whether an undirected graph has a 4-cycle cover. He has also presented a polynomial time algorithm that finds 5-cycle covers in bipartite graphs [10]. Both algorithms also work for the maximization version, if we only have the two possible edge weights zero and one. On the other hand, Cornuéjols and Pulleyblank [4] have reported that Papadimitriou showed the NP-completeness of finding a k-cycle cover in undirected graphs for $k \geq 6$.

One possibility to obtain approximation algorithms for Max-3-DCC is to modify algorithms for the maximum asymmetric TSP. Often it is sufficient just to modify the analysis. Taking the algorithm of Lewenstein and Sviridenko [11], which is the currently best approximation algorithm for this problem, one gets a polynomial time $\frac{7}{12}$-approximation algorithm for Max-3-DCC. For undirected graphs, the algorithm of Serdyukov [14] gives a polynomial time $\frac{7}{10}$-approximation algorithm for finding a 4-cycle cover of maximum weight.

1.2 Our Results

We present two approximation algorithms for Max-3-DCC. The heavier one of the cycle covers produced by these algorithms has at least a fraction of $\frac{3}{5} - \epsilon$ of the weight of an optimal 3-cycle cover. Thus, combining the two algorithms yields a $\left(\frac{3}{5} - \epsilon\right)$-approximation algorithm for Max-3-DCC (for any $\epsilon > 0$) whose running time is proportional to that of computing a maximum weight bipartite matching (and is particularly independent of ϵ). This improves the previously best algorithm for this problem, which achieves a factor of $\frac{7}{12}$. As a lower bound, we prove that Max-3-DCC is APX-complete.

2 Approximation Algorithms for Max-3-DCC

We present two approximation algorithms for Max-3-DCC. The heavier one of the 3-cycle covers computed by these algorithms will have at least a fraction of $\frac{3}{5} - \epsilon$ of the weight of a maximum weight 3-cycle cover.

Input: a complete directed graph G with weight function w
Output: a 3-cycle cover $\mathcal{T}$

1. Compute a maximum weight 2-cycle cover $\mathcal{C}$ of G.
2. Discard the lightest edge of each 2-cycle in $\mathcal{C}$ to obtain a collection $\mathcal{C}'$ of node-disjoint edges and of cycles of length at least three. If there is only one 2-cycle in $\mathcal{C}$, take the lightest edge contained in any of the cycles of length at least three and discard it, too.
3. Construct a 3-cycle cover $\mathcal{T}$ by patching the paths in $\mathcal{C}'$ arbitrarily together.

Fig. 1. Algorithm 1

To avoid lengthy repetitions, we define some names that we use throughout this section. The input graph is called G, its node set is denoted by V, and the cardinality of V is n. Both algorithms start with computing a 2-cycle cover on the input graph G. We call this cycle cover $\mathcal{C}$. Technically, we treat a cycle cover as the set of its edges. The cycles in $\mathcal{C}$ are $C_1, \ldots, C_\ell$. The total weight of $\mathcal{C}$ is denoted by W. Since a 3-cycle cover is also a 2-cycle cover, W is an upper bound for the weight of an optimum 3-cycle cover. Let $I_2 \subseteq \{1, \ldots, \ell\}$ be the set of all i such that C_i is a 2-cycle (a 2-cycle is a cycle of length two). For each $i \in I_2$, we choose $b_i, c_i \in [0, 1]$ such that $b_i \cdot W$ and $c_i \cdot W$ are the weight of the heavier and lighter edge, respectively, of the 2-cycle C_i. Moreover, $b := \sum_{i \in I_2} b_i$ and $c := \sum_{i \in I_2} c_i$.

2.1 Algorithm 1

Algorithm 1 is a simple factor $\frac{1}{2}$-approximation algorithm. It starts with computing a 2-cycle cover. Then it discards the lightest edge of each 2-cycle and patches the obtained edges together to form one big cycle. If there is only one 2-cycle, then also one longer cycle will be broken. The edge of the 2-cycle and the path obtained from this longer cycle are then patched together to form a cycle of length at least five. The worst case for Algorithm 1 is $b = c = \frac{1}{2}$. However, if $\mathcal{C}$ contains cycles of length three or more that have a significant portion of the total weight of $\mathcal{C}$ or b is much larger than c, then Algorithm 1 yields a better approximation ratio. More precisely, the amount of weight contained in $\mathcal{C}'$ is at least $\left(1 - c - \frac{1}{n}\right) \cdot W$. The loss of $c \cdot W$ comes from discarding the lightest edge in each 2-cycle and the loss of $\frac{1}{n} \cdot W$ is incurred when we have to break one cycle of length at least three. Since all edge weights are nonnegative, we do not loose any weight when patching the edges and paths together. This proves the following lemma.

Lemma 1. *Algorithm 1 computes a $(1 - c - \frac{1}{n})$-approximation to a maximum weight 3-cycle cover.* □

Input: a complete directed graph G with weight function w
Output: a 3-cycle cover $\mathcal{T}$

1. Compute a maximum weight 2-cycle cover $\mathcal{C}$ of G.
2. Define a new weight function w' on G as follows: for each $i \in I_2$ assign both edges in C_i the new weight zero. All other edges keep their old weight.
3. Compute a maximum weight matching $\mathcal{M}$ on G with respect to w'.
4. Let $\mathcal{M}' = \mathcal{M} \setminus \mathcal{C}$.
5. Color the edges of $\mathcal{C} \cup \mathcal{M}'$ according to Lemma 2 with two colors.
6. Add each edge $e \in \mathcal{M} \setminus \mathcal{M}'$ that is not contained in a 2-cycle of $\mathcal{C}$ to the color class that does not already contain e.
7. Patch the paths in the color class with the larger weight arbitrarily together to obtain a 3-cycle cover $\mathcal{T}$. (If neccessary, break one longer cycle as in Algorithm 1.)

Fig. 2. Algorithm 2

2.2 Algorithm 2

In Algorithm 2, we pay special attention to the 2-cycles. Like Algorithm 1, Algorithm 2 starts with computing a maximum weight cycle cover $\mathcal{C}$ and then transforms it into a 3-cycle cover. To this aim, we define a new weight function w' by setting the weight of the edges of every 2-cycle to zero. Then we compute a maximum weight matching $\mathcal{M}$ with respect to the new weight function. That means, we replace the two edges between each pair of nodes by an undirected edge with weight equal to the maximum of the weight of the two replaced edges. Then we compute a matching of maximum weight on that graph. Finally, we translate everything back into the directed world by replacing each undirected edge by the directed one for which the maximum was attained (breaking ties arbitrarily). Then we color the edges of $\mathcal{M}$ and $\mathcal{C}$ with two colors in such way that each color class forms a collection of node disjoint paths and cycles of length at least three. This is the reason why we give the edges of the 2-cycles weight zero under w'. Otherwise, we could get triple edges and would consequently need three colors instead of two. Then we take the color class with larger weight and patch the paths arbitrarily together to obtain a 3-cycle cover. If this color class contains only one path and this path has length one, then we have to break one of the cycles.

The next lemma shows that the coloring described always exists. To apply this lemma, we temporarily remove all edges from $\mathcal{M}$ that are also edges in $\mathcal{C}$. Call the resulting set $\mathcal{M}' = \mathcal{M} \setminus \mathcal{C}$. The graph $(V, \mathcal{M}' \cup \mathcal{C})$ fulfills the premise of the lemma. Thus, we can color the edges in $\mathcal{M}' \cup \mathcal{C}$ with two colors. Thereafter, we deal with the edges in $\mathcal{M} \setminus \mathcal{M}'$ that are not part of a 2-cycle. These edges are already in one color class (because of their occurrence in $\mathcal{C}$) and can be safely placed into the other color class, too, without creating any 2-cycles. Edges in

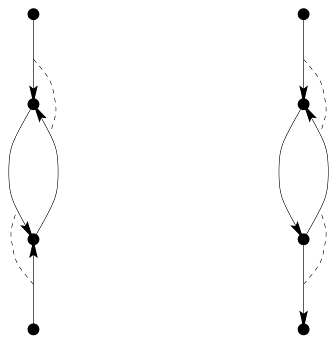

Fig. 3. The two ways how a 2-cycle can interact with the other edges. (Solid edges are edges from the graph G, dashed edges represent the resulting edges in H.)

$\mathcal{M} \setminus \mathcal{M}'$ that are part of a 2-cycle in $\mathcal{C}$ are ignored, since we only consider the modified weight $w'(\mathcal{M})$.

Lemma 2. *Let $G = (V, E)$ be a directed loopless graph such that*

1. *every node in V has indegree at most two,*
2. *every node in V has outdegree at most two, and*
3. *every node in V has total degree at most three.*

Then the edges of G can be colored with two colors such that each color class consists solely of node-disjoint paths and cycles of length at least three.

Proof. To be specific, we call the colors red and blue. We construct an auxiliary undirected graph $H = (E, Z)$ whose nodes are the edges of G. There is an edge $\{e, f\}$ in Z iff there are nodes u, v, and x in V such that $e = (u, x)$ and $f = (v, x)$ or $e = (x, u)$ and $f = (x, v)$, in other words, if e is red, then f has to be blue and vice versa. By assumption, every node in H has degree at most two. Hence, H consists solely of simple cycles and paths. By construction, all cycles in H have even length. This is due to the fact that an edge in Z corresponds to the event that either the heads or the tails of two edges from E meet in one node in G. This already shows that we can color the edges of G with the colors red and blue such that each of the two color classes consists of node-disjoint paths and cycles. We just color the edges of each path and cycle in an alternating way.

But each class could still contain 2-cycles which we now have to eliminate. Note that for each cycle and each path in H, there are two possible colorings and we can choose one of them arbitrarily. This is the key to eliminate the 2-cycles.

Figure 3 shows how a 2-cycle in G can interact with the other edges in G. Due to the degree restrictions in G, either of the two nodes of a 2-cycle can only have at most one other edge. The case depicted on the left-hand side also includes the case where both edges not in the 2-cycle are reversed. The right-hand side also treats the case where one or two edges are missing. The solid edges are edges in the graph G. The dashed edges are the edges in H connecting the edges of G, i.e, the nodes of H.

The case on the right-hand side in Fig. 3 is easy, here we have in H one long path and one single node which is one of the edges of the 2-cycle. Since we can choose its color arbitrarily, we color this single node in H red if the other edge of the 2-cycle is colored blue and vice versa.

In the case on the left-hand side, we have two paths in H whose end-nodes are the edges of the 2-cycle in G. To ensure that these two end-nodes get different colors, we "connect" these end-nodes in H, i.e., we add the additional edge $\{e, f\}$ to the edges of H, where e and f denote the two edges of the 2-cycle. This can of course create new cycles in H. But whenever we add such an edge $\{e, f\}$, then either two tails or two heads are connected. It follows that all the newly generated cycles have even length. Thus, we can color the edges of G with the colors red and blue such that each of the two color classes consists of node-disjoint paths and cycles of length at least three. $\square$

In order to estimate the approximation performance of Algorithm 2, we have to bound the weight of the matching $\mathcal{M}$. This is done in the following lemma.

Lemma 3. *Let $\mathcal{T}_{\mathrm{opt}}$ be a maximum weight 3-cycle cover on G and let $w(\mathcal{T}_{\mathrm{opt}}) = L$. Let $I_2' \subseteq I_2$ be the set of all 2-cycles C of $\mathcal{C}$ such that C and $\mathcal{T}_{\mathrm{opt}}$ have a common edge. Furthermore, set $b' = \sum_{i \in I_2'} b_i$ and $c' = \sum_{i \in I_2'} c_i$. Then the weight of the matching $\mathcal{M}$ computed by Algorithm 2 with respect to w' is at least*

$$w'(\mathcal{M}) \geq \tfrac{1}{2} \cdot L - \tfrac{1}{6} \cdot W + \tfrac{1}{6}\bigl(c' - 2b'\bigr) \cdot W .$$

Proof. We divide the cycles of $\mathcal{T}_{\mathrm{opt}}$ into two sets $\mathcal{S}$ and $\overline{\mathcal{S}} = \mathcal{T}_{\mathrm{opt}} \setminus \mathcal{S}$. The set $\mathcal{S}$ contains all cycles that have an edge with a 2-cycle from $\mathcal{C}$ in common. With respect to w', $\mathcal{T}_{\mathrm{opt}}$ has weight at least $L - b' \cdot W$, since a cycle of length at least three can run through only one edge of a 2-cycle. On the other hand, the total weight of the cycles in $\overline{\mathcal{S}}$ is at most $(1 - c' - b') \cdot W$. Otherwise we could add the cycles C_i with $i \in I_2'$ to $\overline{\mathcal{S}}$ and would obtain a cycle cover of weight more than W, contradicting the optimality of $\mathcal{C}$. Let $D := w(\overline{\mathcal{S}})$. (Note that also $D = w'(\overline{\mathcal{S}})$ holds.) With respect to w', $\mathcal{S}$ contains weight $w'(\mathcal{S}) \geq L - b' \cdot W - D$.

We now construct a matching $\mathcal{N}$ with $w'(\mathcal{N}) \geq \frac{1}{2} \cdot L - \frac{1}{6} \cdot W + \frac{1}{6}(c' - 2b') \cdot W$. This implies the assertion of the lemma. We can color the edges of $\overline{\mathcal{S}}$ with three colors such that each color class forms a (partial) matching. The worst-case for $\overline{\mathcal{S}}$ is a collection of 3-cycles. Let $\mathcal{N}_2$ be the color class with maximum weight, breaking ties arbitrarily. We have $w'(\mathcal{N}_2) \geq \frac{1}{3} D$. Since all cycles in $\mathcal{S}$ have one edge of weight zero under w', we can color the edges with nonzero weight of $\mathcal{S}$ with two colors such that each color class forms a (partial) matching. Let $\mathcal{N}_1$ be the color class of larger weight. We have $w'(\mathcal{N}_1) \geq \frac{1}{2}(L - b' \cdot W - D)$. Then $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2$ has weight at least

$$\begin{aligned} w'(\mathcal{N}) &\geq \tfrac{1}{2} \cdot L - \tfrac{1}{2} b' \cdot W - \tfrac{1}{6} D \\ &\geq \tfrac{1}{2} \cdot L - \tfrac{1}{6} \cdot W + \tfrac{1}{6}\bigl(c' - 2b'\bigr) \cdot W , \end{aligned}$$

where the last inequality follows by plugging in $D \leq (1 - c' - b') \cdot W$. $\square$

The next lemma bounds the approximation performance of Algorithm 2.

Lemma 4. *Let b' and c' be defined as in Lemma 3. Algorithm 2 computes a $(\frac{2}{3} + \frac{1}{12}(c' - 2b') - \frac{1}{n})$-approximation to a maximum weight 3-cycle cover of G.*

Proof. After step 6 of Algorithm 2, both color classes together contain all edges from $\mathcal{C}$ and those edges from $\mathcal{M}$ that are not part of any 2-cycle of $\mathcal{C}$. Thus, the total weight in both color classes is at least $w(\mathcal{C}) + w'(\mathcal{M})$. We have $w(\mathcal{C}) = W$. The weight contained in $\mathcal{M}$ with respect to w' is $w'(\mathcal{M}) \geq \frac{1}{2} \cdot L - \frac{1}{6} \cdot W + \frac{1}{6}(c' - 2b') \cdot W$ by Lemma 3. In step 7, we perhaps loose weight at most $\frac{1}{n} \cdot W$. Thus the total weight of the heavier color class is at least

$$\begin{aligned} &\tfrac{1}{2}\big(W + \tfrac{1}{2} \cdot L - \tfrac{1}{6} \cdot W + \tfrac{1}{6}\big(c' - 2b'\big) \cdot W\big) - \tfrac{1}{n} \cdot W \\ &\qquad \geq \tfrac{2}{3} \cdot L + \tfrac{1}{12}\big(c' - 2b'\big) \cdot L - \tfrac{1}{n} \cdot L\,, \end{aligned}$$

because $W \geq L$ and the sum of the coefficients of W is positive. □

2.3 Combining the Algorithms

The final algorithm runs Algorithm 1 and 2 on G and returns the heavier 3-cycle cover.

For the analysis, we skip the terms $\frac{1}{n} \cdot W$. We will pay for this at the end of the analysis by subtracting an arbitrarily small constant $\epsilon > 0$. The approximation factor of the combined algorithm can be bounded as follows:

$$\begin{array}{ll} \text{minimize} & \max\{1 - c, \frac{2}{3} + \frac{1}{12}(c' - 2b')\} \\ \text{subject to} & 0 \leq c' \leq b'\,, \\ & c' + b' \leq 1\,, \\ & 0 \leq c \leq \frac{1}{2}(1 - b' + c')\,. \end{array}$$

Some simple calculations show that the above minimum is $\frac{3}{5}$. It is attained for $b' = \frac{3}{5}$ and $c = c' = \frac{2}{5}$.

Theorem 1. *For any $\epsilon > 0$, there is a factor $\big(\frac{3}{5} - \epsilon\big)$-approximation algorithm for* Max-3-DCC *running in polynomial time.* □

The running time of the above algorithm is dominated by the time needed to compute a maximum weight cycle cover. This time is proportional to the time needed to compute a maximum weight bipartite matching. Hence it is $O(n^3)$ or $O(n^{5/2} \log(nB))$ where B is the largest weight in the given graph (see Ahuja et al. [1]).

3 Max-3-DCC is APX-complete

In this section we prove that Max-3-DCC is APX-complete. For this purpose we reduce E3-Max-Cut to Max-3-DCC. An instance of Max-Cut is an undirected graph $H = (X, Z)$. The goal is to find a subset $\tilde{X} \subseteq X$ of nodes such that the

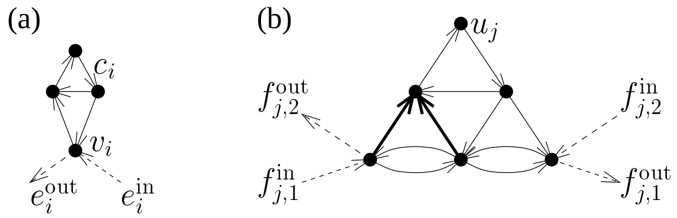

Fig. 4. (a) The node gadget X_i representing the node x_i. All edges drawn have weight 2. (b) The edge gadget Z_j representing edge $z_j \in Z$. The thick edges have weight $\frac{11}{6}$, all other edges drawn have weight 2. Edges left out in the figures have weight 1.

number of edges connecting $\tilde{X}$ and $X \setminus \tilde{X}$ is maximized. In the following, we denote the set of edges between $\tilde{X}$ and $X \setminus \tilde{X}$ by $\text{cut}(\tilde{X})$. The problem Max-Cut is known to be APX-complete, even if restricted to cubic graphs (see Alimonti and Kann [2]). We call Max-Cut restricted to cubic graphs E3-Max-Cut.

We present an L-reduction (see Papadimitriou and Yannakakis [13] for a definition) from E3-Max-Cut to Max-3-DCC to prove the APX-hardness of Max-3-DCC.

Let $H = (X, Z)$ be an instance for E3-Max-Cut, i.e., an undirected cubic graph. Let $X = \{x_1, x_2, \ldots, x_n\}$ be the set of nodes and $Z = \{z_1, z_2, \ldots, z_m\}$ be the set of edges. Since H is a cubic graph, we have $n = \frac{2}{3} \cdot m$.

Let us now construct a directed edge weighted graph $G = (V, E)$ as an instance for Max-3-DCC. For each $x_i \in X$ create a node $v_i \in V$. This node v_i is connected with a cycle c_i of length 3 (see Fig. 4a). Furthermore, there is an edge e_i^{out} starting at v_i and an edge e_i^{in} ending at v_i. Such a subgraph is called *node gadget* X_i *for* x_i. For each edge $z_j \in Z$ create an *edge gadget* Z_j as depicted in Fig. 4b.

The gadgets and the nodes are connected as follows. We order the nodes incident with an edge arbitrarily. Analogously, we order the edges incident with a node arbitrarily. Assume that z_j, $z_{j'}$, and $z_{j''}$ are the first, second, and third edge, respectively, incident with a node x_i. If x_i is the first node of edge z_j then e_i^{out} and $f_{j,1}^{\text{in}}$ are identical, if x_i is the second node of z_j then e_i^{out} and $f_{j,2}^{\text{in}}$ are identical. The corresponding outgoing edge of Z_j is identical with one incoming edge of $Z_{j'}$ depending on whether x_i is the first or the second node of $z_{j'}$. The gadgets $Z_{j'}$ and $Z_{j''}$ are connected in a similar manner. Finally, one of the edges $f_{j'',1}^{\text{out}}$ and $f_{j'',2}^{\text{out}}$ is identical with e_i^{in}. Figure 5 shows an example of a graph H and the corresponding graph G.

We call a cycle cover $\mathcal{C}$ of G *consistent with a subset* $\tilde{X} \subseteq X$, if the following properties hold:

1. All edge gadgets are traversed as depicted in Fig. 6.
2. If $x_i \in \tilde{X}$ then both e_i^{in} and e_i^{out} are in $\mathcal{C}$ and c_i forms a cycle of length 3. If $x_i \notin \tilde{X}$ then v_i and c_i form a cycle of length 4.

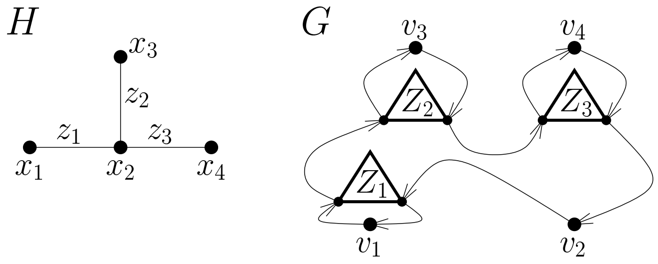

Fig. 5. A graph H and the corresponding graph G. For the sake of readability H is not cubic. The edge gadgets are symbolized by triangles, the cycles of the node gadgets are left out. The node x_2 is the second node of z_1 and the first node of both z_2 and z_3.

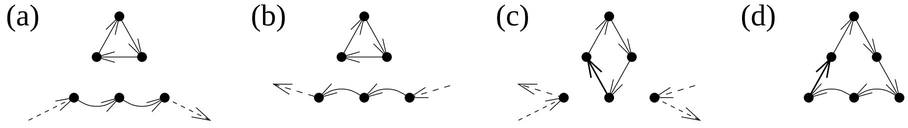

Fig. 6. Consistent traversals of the gadget Z_j representing the edge $z_j = \{x_i, x_{i'}\}$. (a) only x_i is in $\tilde{X}$, (b) only $x_{i'}$ is in $\tilde{X}$, (c) both x_i and $x_{i'}$ are in $\tilde{X}$, (d) neither node of z_j is in $\tilde{X}$.

We call a cycle cover $\mathcal{C}$ *consistent* if there exists a subset $\tilde{X}$ such that $\mathcal{C}$ is consistent with $\tilde{X}$.

The weight of a node in V with respect to a cycle cover $\mathcal{C}$ is the sum of half the weight of its incoming edge plus half the weight of its outgoing edge. The weight $w(Z_j)$ of the edge gadget Z_j is the sum of the weights of its nodes. Likewise, the weight $w(X_i)$ of the node gadget X_i is the sum of the weights of its nodes. Assume that $z_j = \{x_i, x_{i'}\}$. Then we call

$$b(Z_j) := w(Z_j) + \tfrac{1}{3} \cdot (w(X_i) + w(X_{i'}))$$

the *burden of* Z_j. We have

$$w(\mathcal{C}) = \sum_{j=1}^{m} w(Z_j) + \sum_{i=1}^{n} w(X_i) = \sum_{j=1}^{m} b(Z_j) .$$

In a cycle cover consistent with $\tilde{X}$ we have $w(X_i) = 8$ for all i. For every edge gadget Z_j, we have either $w(Z_j) = 12$ and $b(Z_j) = \frac{104}{6}$ if $z_j \in \mathrm{cut}(\tilde{X})$ or $w(Z_j) = \frac{71}{6}$ and $b(Z_j) = \frac{103}{6}$ if $z_j \notin \mathrm{cut}(\tilde{X})$. Thus, we have the following lemma.

Lemma 5. *Let $\mathcal{C}$ be the cycle cover consistent with $\tilde{X}$. Then $w(\mathcal{C}) = \frac{103}{6} \cdot m + \frac{1}{6} \cdot |\mathrm{cut}(\tilde{X})|$.* □

Consider an arbitrary cycle cover $\mathcal{C}$ of G with weight $w(\mathcal{C}) = \frac{103}{6} \cdot m + \frac{1}{6} \cdot \gamma$. We describe how to construct a subset X' with the property that if $b(Z_j) > \frac{103}{6}$ then $z_j \in \mathrm{cut}(X')$. Thus, $|\mathrm{cut}(X')| \geq \gamma$.

In the following, assume that $z_j = \{x_i, x_{i'}\}$. If a node gadget X_i is inconsistent then we have $w(X_i) \leq \frac{15}{2}$. If an edge gadget Z_j is inconsistent then we have $w(Z_j) \leq \frac{23}{2}$. Hence, if the edge gadget Z_j or one of the node gadgets X_i or $X_{i'}$ is inconsistent, we have $b(Z_j) \leq \frac{103}{6}$.

Let us now consider all edge gadgets Z_j with burden $b(Z_j) > \frac{103}{6}$. Due to the considerations above, all these edge gadgets are consistent. Furthermore, if $b(Z_j) > \frac{103}{6}$ then both X_i and $X_{i'}$ are consistent. Thus, all edge gadgets with $b(Z_j) > \frac{103}{6}$ are consistent and fulfil $b(Z_j) = \frac{104}{6}$. If we can find a subset $\tilde{X}$ fulfilling $\mathrm{cut}(\tilde{X}) \supseteq \{z_j \mid b(Z_j) = \frac{104}{6}\}$, we are done.

An edge gadget Z_j with $b(Z_j) = \frac{104}{6}$ is called a *witness for $x_i \in \tilde{X}$*, if it is traversed as shown in Fig. 6a or 6c. Otherwise, it is called a *witness for $x_i \notin \tilde{X}$*. Likewise, we call Z_j a witness for $x_{i'} \in \tilde{X}$ if it is traversed as shown in Fig. 6b or 6c, and for $x_{i'} \notin \tilde{X}$, otherwise.

Assume that for some node x_i we have two edge gadgets Z_j and $Z_{j'}$ such that Z_j is a witness for $x_i \in \tilde{X}$ and $Z_{j'}$ is a witness for $x_i \notin \tilde{X}$. Since both gadgets are consistent, z_j and $z_{j'}$ are the first and the third edge, respectively, of x_i or vice versa. But this implies, that the node gadget X_i cannot be consistent. Hence, $b(Z_j) \leq \frac{103}{6}$ and $b(Z_{j'}) \leq \frac{103}{6}$, a contradiction.

The witnesses induce a partition $X_\in, X_\notin, X_?$ of X, such that all nodes in $X_\in$ have only witnesses for $x \in \tilde{X}$, all nodes in $X_\notin$ have only witnesses for $x \notin \tilde{X}$, and all nodes in $X_?$ do not have any witness. Due to the construction, there are at least γ edges between $X_\in$ and $X_\notin$. Choose $\tilde{X}$ arbitrarily with $X_\in \subseteq \tilde{X} \subseteq X_\in \cup X_?$. Then $|\mathrm{cut}(\tilde{X})| \geq \gamma$. Hence, we have proved the following lemma.

Lemma 6. *Given an arbitrary cycle cover $\mathcal{C}$ with weight $w(\mathcal{C}) = \frac{103}{6} \cdot m + \frac{1}{6} \cdot \gamma$ we can construct a consistent cycle cover $\mathcal{C}'$ with weight $w(\mathcal{C}') \geq w(\mathcal{C})$ in polynomial time.* □

Now we can prove the main theorem of this section.

Theorem 2. Max-3-DCC *is* APX*-complete.*

Proof. We show that the reduction presented above is an L-reduction.

We denote the size of the maximum cut of H with $\mathrm{opt}(H)$ and the weight of the maximum cycle cover of G with $\mathrm{opt}(G)$.

For any graph $H = (X, Z)$ there exists a subset $\tilde{X}$ of X such that $|\mathrm{cut}(\tilde{X})| \geq \frac{m}{2}$. Thus, $\mathrm{opt}(H) \geq \frac{m}{2}$ and $\mathrm{opt}(G) \leq \frac{104}{6} \cdot m \leq \frac{104}{3} \cdot \mathrm{opt}(H)$.

On the other hand, given a cycle cover $\mathcal{C}$ with weight $w(\mathcal{C}) = \frac{103}{6} \cdot m + \frac{1}{6} \cdot \gamma$ we can construct a subset $\tilde{X}$ of X with $|\mathrm{cut}(\tilde{X})| \geq \gamma$ in polynomial time, see Lemma 6. Thus, the following holds:

$$\begin{aligned} \left|\mathrm{opt}(H) - |\mathrm{cut}(\tilde{X})|\right| &= 6 \cdot \left|\mathrm{opt}(G) - \left(\tfrac{103}{6} \cdot m + \tfrac{1}{6} \cdot |\mathrm{cut}(\tilde{X})|\right)\right| \\ &\leq 6 \cdot \left|\mathrm{opt}(G) - w(\mathcal{C})\right| . \end{aligned}$$

This proves that the reduction presented is an L-reduction. Since E3-Max-Cut is APX-complete and Max-3-DCC is in APX, the theorem is proved. □

4 Open Problems

A problem that remains open is the approximability of computing minimum weight 3-cycle covers. Without any restrictions, this problem is NPO-complete. With the triangle inequality, we can perform two iterations of the algorithm of Frieze, Galbiati, and Maffioli [6] for the asymmetric TSP. This yields an approximation factor of two. It seems to be a challenging problem to improve this factor of two, since this could also yield new insights into the approximability of the asymmetric TSP.

Another open problem is the status of maximum weight undirected 4-cycle covers. The question is whether this problem can be solved in polynomial time or has at least a polynomial time approximation scheme, provided that $\mathsf{P} \neq \mathsf{NP}$.

References

1. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1993.
2. P. Alimonti and V. Kann. Some APX-completeness results for cubic graphs. *Theoret. Comput. Sci.*, 237(1-2):123–134, 2000.
3. M. Bläser and B. Siebert. Computing cycle covers without short cycles. In *Proc. 9th Ann. European Symp. on Algorithms (ESA)*, volume 2161 of *Lecture Notes in Comput. Sci.*, pages 368–379. Springer, 2001.
4. G. P. Cornuéjols and W. R. Pulleyblank. A matching problem with side conditions. *Discrete Math.*, 29:135–159, 1980.
5. J. Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.
6. A. M. Frieze, G. Galbiati, and F. Maffioli. On the worst-case performance of some algorithms for the traveling salesman problem. *Networks*, 12(1):23–39, 1982.
7. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.
8. R. L. Graham, M. Grötschel, and L. Lovász, editors. *Handbook of Combinatorics*, volume 1. Elsevier, 1995.
9. D. Hartvigsen. *An Extension of Matching Theory.* PhD thesis, Carnegie-Mellon University, 1984.
10. D. Hartvigsen. The square-free 2-factor problem in bipartite graphs. In *Proc. 7th Int. Conf. on Integer Programming and Combinatorial Optimization (IPCO)*, volume 1620 of *Lecture Notes in Comput. Sci.*, pages 234–241. Springer, 1999.
11. M. Lewenstein and M. Sviridenko. A 5/8 approximation algorithm for the asymmetric maximum TSP. Manuscript, 2002.
12. L. Lovász and M. D. Plummer. *Matching Theory.* Elsevier, 1986.
13. C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. System Sci.*, 43(3):425–440, 1991.
14. A. I. Serdyukov. An algorithm with an estimate for the traveling salesman problem of the maximum. *Upravlyaemye Sistemy*, 25:80–86, 1984. (in Russian).

15. W. T. Tutte. A short proof of the factor theorem for finite graphs. *Canad. J. Math.*, 6:347–352, 1954.
16. L. G. Valiant. The complexity of computing the permanent. *Theoret. Comput. Sci.*, 8(2):189–201, 1979.