Computing Cycle Covers without Short Cycles

Markus Bläser and Bodo Siebert*

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

Abstract. A cycle cover of a graph is a spanning subgraph where each node is part of exactly one simple cycle. A k-cycle cover is a cycle cover where each cycle has length at least k. We call the decision problems whether a directed or undirected graph has a k-cycle cover k-DCC and k-UCC. Given a graph with edge weights one and two, Min-k-DCC and Min-k-UCC are the minimization problems of finding a k-cycle cover with minimum weight.
We present factor $4/3$ approximation algorithms for Min-k-DCC with running time $O(n^{5/2})$ (independent of k). Specifically, we obtain a factor $4/3$ approximation algorithm for the asymmetric travelling salesperson problem with distances one and two and a factor $2/3$ approximation algorithm for the directed path packing problem with the same running time. On the other hand, we show that k-DCC is $\mathcal{NP}$-complete for $k \geq 3$ and that Min-k-DCC has no PTAS for $k \geq 4$, unless $\mathcal{P} = \mathcal{NP}$.
Furthermore, we design a polynomial time factor $7/6$ approximation algorithm for Min-k-UCC. As a lower bound, we prove that Min-k-UCC has no PTAS for $k \geq 12$, unless $\mathcal{P} = \mathcal{NP}$.

1 Introduction

A cycle cover of an either directed or undirected graph G is a spanning subgraph C where each node of G is part of exactly one simple cycle of C. Computing cycle covers is an important task in graph theory, see for instance Lovász and Plummer [14], Graham et al. [7], and the vast literature cited there.

A k-restricted cycle cover (or k-cycle cover for short) is a cycle cover in which each cycle has length at least k. To be specific, we call the decision problems whether a graph has a k-cycle cover k-DCC, if the graph is directed, and k-UCC, if the graph is undirected. Since k-DCC and k-UCC are $\mathcal{NP}$-complete for $k \geq 3$ and $k \geq 6$, respectively, we also consider the following relaxation: given a complete loopless graph with edge weights one and two, find a k-cycle cover of minimum weight. Note that a graph $G = (V, E)$ has a k-cycle cover if the corresponding weighted graph has a k-cycle cover of weight $|V|$, where edges get weight one and "nonedges" get weight two in the corresponding complete graph. We call these problems Min-k-DCC and Min-k-UCC. They stand in one-to-one correspondence with simple 2-factors as defined by Hartvigsen [9]. A simple 2-factor is a spanning subgraph that contains only node-disjoint paths and cycles of length at least k. (The paths arise from deleting the weight two edges from the cycles.)

* supported by DFG research grant Re 672/3

As our main contribution, we devise approximation algorithms for finding minimum weight k-cycle covers in graphs with weights one and two. Moreover, we provide lower bounds in terms of $\mathcal{NP}$-completeness and nonapproximability, thus determining the computational complexity of these problems for almost all k.

1.1 Previous results

The problems 2-DCC and Min-2-DCC of finding a (minimum) 2-cycle cover in directed graphs can be solved in polynomial time by reduction to the bipartite matching problem. To our knowledge, nothing is known for values $k \geq 3$.

The problem 3-UCC of finding a 3-cycle cover in undirected graphs can be solved in polynomial time using Tutte's reduction [18] to the classical perfect matching problem in undirected graphs which can be solved in polynomial time (see Edmonds [4]). Also Min-3-UCC can be solved in polynomial time. Hartvigsen [8] has designed a powerful polynomial time algorithm for 4-UCC. This algorithm works for Min-4-UCC, too. He has also presented a polynomial time algorithm that computes a minimum weight 5-cycle cover in graphs where the weight one edges form a bipartite graph [9]. On the other hand, Cornuéjols and Pulleyblank [3] have reported that Papadimitriou showed the $\mathcal{NP}$-completeness of k-UCC for $k \geq 6$.

Let n be the number of nodes of a graph $G = (V, E)$. For $k > n/2$ the problem Min-k-DCC is the asymmetric and Min-k-UCC is the symmetric travelling salesperson problem with distances one and two. These problems are $\mathcal{APX}$-complete [17]. For explicit lower bounds, see Engebretsen and Karpinski [5]. The best upper bound for the symmetric case is due to Papadimitriou and Yannakakis [17]. They give a factor $7/6$ approximation algorithm running in polynomial time. For the asymmetric case, Vishwanathan [19] presents a polynomial time factor $17/12$ approximation algorithm. Exploiting an algorithm by Kosaraju, Park, and Stein [12] for the asymmetric maximum travelling salesperson problem, one obtains an approximation algorithm with performance ratio $88/63 \approx 1.397$ by replacing weights two with weights zero.

Closely related to the travelling salesperson problems with distances one and two is the node-disjoint path packing problem. This problem has various applications, such as mapping parallel programs to parallel architectures and optimization of code, see e.g. Vishwanathan [19] and the pointers provided there. We are given a directed or undirected graph. Our goal is to find a spanning subgraph S consisting of node-disjoint paths such that the number of edges in S is maximized. Utilizing the algorithms of Papadimitriou and Yannakakis [17] and of Kosaraju, Park, and Stein [12], one obtains a polynomial time factor $5/6$ approximation algorithm for the undirected problem and a polynomial time approximation algorithm with performance ratio $38/63 \approx 0.603$ for the directed problem.

1.2 Our results

We present factor $4/3$ approximation algorithms for Min-k-DCC with running time $O(n^{5/2})$ (independent of k). Specifically, we obtain a factor $4/3$ approximation algorithm for the asymmetric travelling salesperson problem with distances one and two and a factor $2/3$ approximation algorithm for the directed node-disjoint path packing

Input: a complete loopless directed graph G with edge weights one and two, an integer $k \geq 3$.
Output: a k-cycle cover of G.

1. Compute a minimum weight 2-cycle cover C of G.
2. Form the bipartite graph B and compute the function F.
3. Compute a decomposition of F as in Lemma 2
4. Patch the cycles of C together acccording to the refined patching procedure.

Fig. 1. The algorithm for directed cycle covers.

problem with the same running time, thus improving the results of Vishwanathan and Kosaraju, Park, and Stein. On the other hand, we show that k-DCC is $\mathcal{NP}$-complete for $k \geq 3$ and that Min-k-DCC does not have a PTAS for $k \geq 4$, unless $\mathcal{P} = \mathcal{NP}$. For the undirected case, we design factor $7/6$ approximation algorithms for Min-k-UCC with polynomial running time (independent of k). It includes the algorithm of Papadimitriou and Yannakakis as a special case. As a lower bound, we prove that there is no PTAS for Min-k-UCC for $k \geq 12$, unless $\mathcal{P} = \mathcal{NP}$.

2 Approximation algorithms for directed cycle covers

In this section, we present approximation algorithms for Min-k-DCC with performance ratio $4/3$ for any $k \geq 3$ running in time $O(n^{5/2})$ (independent of k). Particularly, we obtain a factor $4/3$ approximation algorithm for the asymmetric travelling salesperson problem with distances one and two by choosing $k > n/2$.

Our input consists of a complete loopless directed graph G with node set V of cardinality n, a weight function w that assigns each edge of G weight one or two, and an integer $k \geq 3$. Our aim is to find a k-cycle cover of minimum weight. For the analysis, we assume that a minimum weight k-cycle cover of G has weight $n + \ell$ for some $0 \leq \ell \leq n$. In other words, a minimum weight k-cycle cover consists of $n - \ell$ edges of weight one and ℓ edges of weight two.

Figure 1 gives an overview of our algorithm. A detailed explanation of each of the steps is given in the subsequent paragraphs.

Computing a 2-cycle cover. We first compute an optimal 2-cycle cover C of G. This can be done in polynomial time. Assume that this cover C consists of cycles $c_1, \dots, c_r$. We denote the set $\{c_1, \dots, c_r\}$ by $\mathcal{C}$. The lengths of some of these cycles may already be k or larger, but some cycles, say $c_1, \dots, c_s$ for $s \leq r$, have length strictly less than k. The basic idea of our algorithm is to use *cycle patching* (also called subtour patching when considering travelling salesperson problems, see Lawler et al. [13]). A straight forward way is to discard one edge (if possible of weight two) of each cycle of length strictly less than k and patch the resulting paths arbitrarily together to obtain one long cycle. An easy analysis shows that this yields a factor $3/2$ approximation. We obtain the $4/3$ approximation by refining the patching procedure.

Auxiliary edges. For the refined patching procedure, we form a bipartite graph B as follows: we have the node set $\mathcal{C}^< = \{c_1, \ldots, c_s\}$ on the one side and V on the other. There is an edge (c, v) in B iff v does not belong to the cycle c and there is a node u in c such that (u, v) has weight one in G.

Lemma 1. *B has a matching of cardinality at least $s - \ell$.*

Proof. Consider an optimal k-cycle cover C_{opt} of G. Since the length of the cycles in $\mathcal{C}^<$ are strictly less than k, for each cycle c in $\mathcal{C}^<$ there is an edge (u, v) of C_{opt} such that u belongs to c but v does not. Fix such an edge for each cycle in $\mathcal{C}^<$. At least $s - \ell$ of these edges have weight one, thus appear in B and form a matching. □

Decomposition of functions. We compute a maximum matching M in B. From M we obtain a directed graph $F = (\mathcal{C}, A)$ with $(c, c') \in A$ whenever (c, u) is an edge of M and u is a node of c'. Each node of F has outdegree at most one, thus F defines a partial function $\mathcal{C} \to \mathcal{C}$ whose domain is a subset of $\mathcal{C}^<$. By abuse of notation, we call this function again F. By the construction of B, we have $F(c) \neq c$ for all $c \in \mathcal{C}$, i.e. F does not contain any loops.

Lemma 2. *Any loopless partial function F has a spanning subgraph S consisting solely of node-disjoint trees of depth one, paths of length two, and isolated nodes such that any node in the domain of F is not an isolated node of S. Such a spanning subgraph S can be found in polynomial time.*

Proof. Every weakly connected component of F is either a cycle possibly with some trees converging into it, a tree (whose root r is not contained in the domain of F), or an isolated node (which is also not contained in the domain of F). It suffices to prove the lemma for each weakly connected component of F.

The case where a component is a cycle with some trees converging into it follows from Papadimitriou and Yannakakis [17, Lem. 2]. In this case, no isolated nodes arise.

In the case of a tree, we take a leaf that has maximum distance from the root r. Let s be the successor of that leaf. If s equals r, then the component considered is a tree of depth one and we are done. Otherwise, we build a tree of height one with root s and all predecessors of s as leaves, remove this tree, and proceed inductively. We end up with a collection of node disjoint trees of height one and possibly one isolated node, the root r of the component. Since r is not contained in the domain of F, this case is completed.

If a node is isolated, then this node is not contained in the domain of F, because F is loopless. Again, we are done.

This decomposition can be computed in polynomial time. □

A refined patching procedure. We compute a decomposition of the directed graph F according to Lemma 2. Isolated nodes of this decomposition correspond to elements of $\mathcal{C}$ not in the domain of F, i.e. either cycles of length at least k or unmatched cycles from $\mathcal{C}^<$. The former ones, call them $\mathcal{C}^{\geq}_{\mathrm{iso}}$, fulfil the requirements of a k-cycle cover, thus we can ignore them in the subsequent considerations. We denote the latter ones by $\mathcal{C}^{<}_{\mathrm{iso}}$. The cycles in $\mathcal{C}^{<}_{\mathrm{iso}}$ have length strictly less than k. We merge those cycles to one long cycle d, breaking an edge of weight two whenever possible.

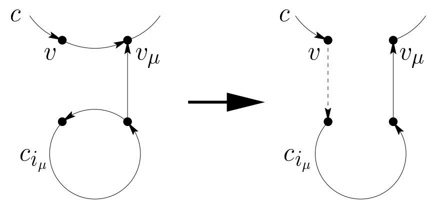

Fig. 2. Trees of height one

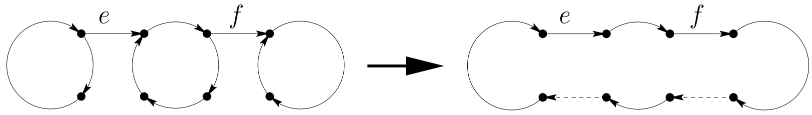

Fig. 3. Paths of length two

Next, we consider the trees of height one. Let c be the root of such a tree and $c_{i_1}, \dots, c_{i_m} \in \mathcal{C}^< \setminus \mathcal{C}^<_{\text{iso}}$ be its leaves. For each cycle c_{i_μ}, there is an edge from c_{i_μ} to a node v_μ of c. By construction, these nodes $v_1, \dots, v_m$ are pairwise distinct. We merge c_{i_1} and c as depicted in Fig. 2. We call this new cycle again c and incorporate the remaining cycles $c_{i_2}, \dots, c_{i_m}$ in the same fashion. (The node v in Fig. 2 may be some $v_{\mu'}$ but this does not matter.) After that we merge c with d. In c, we break one of the edges drawn dashed in Fig. 2. In d, we discard an edge that does not belong to a cycle in $\mathcal{C}^<_{\text{iso}}$, i.e. has been added during the merging process. We call this new cycle again d.

Finally, we consider the paths of length two. The three cycles corresponding to such a path are merged as shown in Fig. 3. (The end node of the edge e and the start node of edge f may coincide. Moreover, the two removed edges of the cycle in the middle may coincide. In the latter case, we only incur weight one instead of weight two.) The resulting cycle will be merged with d as described above in the case of a tree.

At the end of this procedure, we are left with the cycle d and the cycles in $\mathcal{C}^{\geq}_{\text{iso}}$. If the cycle d still has length stricly less than k, we break one cycle $b \in \mathcal{C}^{\geq}_{\text{iso}}$ and merge it with d. The resulting cycle has length at least k. If possible, we choose b such that b contains an edge of weight two and break this edge.

Analysis. The algorithm runs in polynomial time. On a unit-cost RAM (with all used numbers bounded by a polynomial in n), the 2–cycle cover and the bipartite matching can be computed in time $O(n^{5/2})$, see e.g. Papadimitriou and Steiglitz [15]. The decomposition of F and the cycle patching can be done in time $O(n^2)$ in a straight forward manner. Thus, the overall running time is $O(n^{5/2})$.

We proceed with estimating the approximation performance.

Lemma 3. *For $n \geq 12$, the k-cycle cover produced by the algorithm has weight no worse than $4/3$ times the weight of an optimal k-cycle cover.*

Proof. Let C_{opt} be an optimal k-cycle cover of G. Since C_{opt} is also a 2-cycle cover of G, we have $w(C) \leq w(C_{\text{opt}}) = n + \ell$. The cost of the k-cycle cover produced by the algorithm is $w(C)$ plus the extra costs due to the mergings.

First, when merging the cycles in $\mathcal{C}_{\mathrm{iso}}^{<}$ to form the cycle d, we incur an extra cost of one for each $c \in \mathcal{C}_{\mathrm{iso}}^{<}$.

Next, we consider the merging as shown in Fig. 2. We charge the costs to v_μ and the nodes of c_μ. These are at least three nodes. Since the edge of F has weight one, the cost of this merging is at most $1/3$ per node involved. The merging of c with d is free of costs, since we only break edges we have already paid for when forming c and d.

In the case depicted in Fig. 3, we charge the costs of the merging to the nodes of the three cycles. These are at least six nodes. Altogether, the cost of this merging is again at most $1/3$ per node involved. As above, the merging with d is free of costs.

It is clear that each node is only charged once this way. For the moment, assume that the cycle d has length at least k, thus an additional merging is not needed. Let n_2 be the total number of nodes contained in the cycles from $\mathcal{C}_{\mathrm{iso}}^{<}$. The weight $w(C_{\mathrm{apx}})$ of the k-cycle cover C_{apx} produced by the algorithm is at most

$$w(C_{\mathrm{apx}}) \leq n + \ell + \tfrac{1}{3}(n - n_2) + |\mathcal{C}_{\mathrm{iso}}^{<}|. \tag{1}$$

We have $n_2 \geq 2 \cdot |\mathcal{C}_{\mathrm{iso}}^{<}|$ and, by Lemma 1, $|\mathcal{C}_{\mathrm{iso}}^{<}| \leq \ell$. Hence $w(C_{\mathrm{apx}}) \leq \frac{4}{3}(n + \ell)$.

If d has length strictly less than k, then one additional merging is needed. This yields an approximation ratio of $4/3 + \epsilon$ for any $\epsilon > 0$. We can get rid of the ϵ by refining the analysis as follows. Either the merging process of $b \in \mathcal{C}_{\mathrm{iso}}^{\geq}$ and d is free of costs, since b contains an edge of weight two, or all cycles in $\mathcal{C}_{\mathrm{iso}}^{\geq}$ consist solely of weight one edges. Since $\mathcal{C}_{\mathrm{iso}}^{\geq}$ is nonempty, these are at least $n/2$ edges. The cycle d contains at least half of the original edges of the merged cycles. Hence d and the cycles in $\mathcal{C}_{\mathrm{iso}}^{\geq}$ contain at least a fraction of $3/4$ of the edges of the 2-cycle cover C. Thus, after the last merging step, we have a cycle cover of weight at most $\frac{5}{4}(n + \ell) + 1 \leq \frac{4}{3}(n + \ell)$ for $n \geq 12$. □

Theorem 1. *There is a factor* $4/3$ *approximation algorithm for* Min-k-DCC *running in time* $O(n^{5/2})$ *for any* $k \geq 3$. □

Corollary 1. *There is a factor* $4/3$ *approximation algorithm for the asymmetric travelling salesperson problem with distances one and two running in time* $O(n^{5/2})$. □

Corollary 2. *There is a factor* $2/3$ *approximation algorithm for the node-disjoint path packing problem in directed graphs running in time* $O(n^{5/2})$.

Proof. We transform a given directed graph G into a complete loopless directed graph H with edge weights one and two by assigning edges of G weight one and "nonedges" weight two. The details are spelled out by Vishwanathan [19, Sect. 2]. □

3 Approximation algorithms for undirected cycle covers

We outline factor $7/6$ approximation algorithms for Min-k-UCC for any $k \geq 5$. In particular, we recover the factor $7/6$ approximation algorithm for the symmetric travelling salesperson problem with distances one and two of Papdimitriou and Yannakakis [17, Thm. 2] by choosing $k > n/2$. The algorithm is quite similar to the directed case, so we confine ourselves to pointing out the differences.

Computing an optimal 4-cycle cover. Instead of starting with a minimum weight 2-cycle cover, we exploit Hartvigsen's polynomial time algorithm [8] for computing a minimum weight 4-cycle cover C. This gives us the inequality $n_2 \geq 4 \cdot |\mathcal{C}_{\mathrm{iso}}^{<}|$ (instead of $n_2 \geq 2 \cdot |\mathcal{C}_{\mathrm{iso}}^{<}|$ in the directed case).

Auxiliary edges. A little more care is necessary when collecting auxiliary edges via the matching in B, since for each weight two edge, we may now only spend an extra amount of $1/6$ instead of $1/3$. We normalize the computed 4-cycle cover C as follows: first we may assume that there is only one cycle t with weight two edges, since we may merge two such cycles without any costs. Second, we may assume that for each weight two edge $\{u, v\}$ of t there is no weight one edge $\{v, x\}$ in G for some node x of a different cycle, because otherwise we may merge this cycle with t at no costs. We now may bound the number of nodes for which we have to charge extra costs of $1/6$ in (1) by $n - n_2 - \ell$ instead of $n - n_2$. This is due to the fact that if t is the root of a tree of height one according to the decomposition of Lemma 2, then at least ℓ nodes of t are unmatched because of the second above mentioned property of t. Altogether, the total weight is $w(C_{\mathrm{apx}}) \leq \frac{7}{6}n + \frac{5}{6}\ell + \frac{1}{3}|\mathcal{C}_{\mathrm{iso}}^{<}| \leq \frac{7}{6}(n + \ell)$.

Decomposition of functions. The decomposition according to Lemma 2 works without any changes in the undirected case.

A refined patching procedure. Here we use the patching procedure as devised by Papadimitriou and Yannakakis [17, Fig. 2], which is only suited for the undirected case. Together with the fact that each involved circle has at least four nodes (instead of two in the directed case) we obtain lower the merging costs of $1/6$ per node.

Applying the above mentioned modifications to our algorithm for the directed case, we get the following theorem.

Theorem 2. *For any $k \geq 5$, there is a polynomial time factor $7/6$ approximation algorithm for* Min-k-UCC. □

4 Lower bounds

4.1 $\mathcal{NP}$-completeness of 3-DCC

To show the $\mathcal{NP}$-completeness of 3-DCC we will reduce 3-Dimensional Matching (3DM) to this problem. Consider a hypergraph $H = (W, X)$ with $W = W_0 \cup W_1 \cup W_2$ and $X \subseteq W_0 \times W_1 \times W_2$. The sets W_0, W_1, W_2 are disjoint and of the same size, $W_k = \{w_1^k, \ldots, w_n^k\}$. 3DM is the question whether there exists a subset $X' \subseteq X$ such that each element of W appears in exactly one element of X' (perfect 3-dimensional matching). 3DM is known to be $\mathcal{NP}$-complete (Garey, Johnson [6]).

We construct a graph $G = (V, E)$ such that G has a 3-cycle cover iff H has a perfect 3-dimensional matching. Let $V^{\times} = (W_0 \times W_1) \cup (W_1 \times W_2) \cup (W_2 \times W_0)$. For $k = 0, 1, 2$ and $j = 1, \ldots, n$ let $U_j^k = \{u_j^k[i, q] \mid i = 1, \ldots, n-1 \wedge q = 1, 2, 3\}$ be a set of *helper nodes* for w_j^k. The set of nodes V is given by $V = V^{\times} \cup \big(\bigcup_{k=0}^{2} \bigcup_{j=1}^{n} U_j^k\big)$.

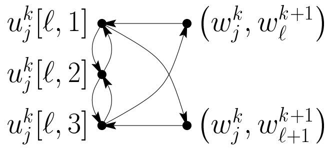

Fig. 4. The subgraph connecting $\left(w_j^k, w_\ell^{k+1}\right)$ and $\left(w_j^k, w_{\ell+1}^{k+1}\right)$ via three helper nodes.

For each edge $\left(w_a^0, w_b^1, w_c^2\right) \in X$ we construct three edges $\left((w_a^0, w_b^1), (w_b^1, w_c^2)\right)$, $\left((w_b^1, w_c^2), (w_c^2, w_a^0)\right)$, $\left((w_c^2, w_a^0), (w_a^0, w_b^1)\right) \in E$ connecting the corresponding elements of $V^\times$. Furthermore, two nodes $\left(w_j^k, w_\ell^{(k+1) \bmod 3}\right)$ and $\left(w_j^k, w_{\ell+1}^{(k+1) \bmod 3}\right)$ are connected via helper nodes as shown in Fig. 4. In the following we write $k+1$ instead of $(k+1) \bmod 3$ for short.

We divide the set $V^\times$ into subsets $\Gamma_j^k = \left\{\left(w_j^k, w_\ell^{k+1}\right) \mid \ell = 1, \ldots, n\right\}$. The subset Γ_j^k contains the nodes that represent w_j^k.

Assume that G has a 3-cycle cover C. We call a helper node $u_j^k[\ell, 2]$ and a node of $V^\times$ *companions* if they are part of the same cycle in C. Due to the construction either $\left(w_j^k, w_\ell^{k+1}\right)$ or $\left(w_j^k, w_{\ell+1}^{k+1}\right)$ is the only companion of $u_j^k[\ell, 2]$. Hence, the following lemma holds.

Lemma 4. *Assume G has a 3-cycle cover G. For any $w_j^k \in V$ exactly $n-1$ of the n nodes in Γ_j^k have a companion.* □

We say that the only node $\left(w_j^k, w_\ell^{k+1}\right) \in \Gamma_j^k$ that has no companion *participates* for w_j^k. Now we are prepared to prove the $\mathcal{NP}$-completeness of 3-DCC.

Theorem 3. 3-DCC *is $\mathcal{NP}$-complete.*

Proof. Given a hypergraph H we construct a graph G as described above.

Assume H has a 3-dimensional matching $X' \subseteq X$. Then G has the following 3-cycle cover. For any $\left(w_a^0, w_b^1, w_c^2\right) \in X'$ let $\left(w_a^0, w_b^1\right)$, $\left(w_b^1, w_c^2\right)$, and $\left(w_c^2, w_a^0\right)$ participate for w_a^0, w_b^1, and w_c^2. These three nodes form a cycle of length 3. Let $\left(w_j^k, w_\ell^{k+1}\right)$ be the node that participates for w_j^k. Then for $\ell' < \ell$ the nodes $u_j^k[\ell', 2]$ and $\left(w_j^k, w_{\ell'}^{k+1}\right)$ and for $\ell' > \ell$ the nodes $u_j^k[\ell'-1, 2]$ and $\left(w_j^k, w_{\ell'}^{k+1}\right)$ are companions. Thus, for $\ell' < \ell$ the nodes $\left(w_j^k, w_{\ell'}^{k+1}\right)$ and $u_j^k[\ell', q]$ ($q = 1, 2, 3$) and for $\ell' > \ell$ the nodes $\left(w_j^k, w_{\ell'}^{k+1}\right)$ and $u_j^k[\ell'-1, q]$ ($q = 1, 2, 3$) form cycles each of length 4. Thus all nodes of G are covered by a cycle of length at least 3.

On the other hand assume that G has a 3-cycle cover C. Due to Lemma 4 we only have to take care for the participating nodes in $V^\times$. The participating nodes form cycles whose lengths are multiples of 3. We cut all the edges from $W_1 \times W_2$ to $W_2 \times W_0$. The remaining paths of lengths two yield a 3-dimensional matching for H.

Noting that 3-DCC is in $\mathcal{NP}$ completes the proof. □

By replacing the nodes of $V^\times$ by paths and extending the helper node constructions we obtain the following generalization.

Theorem 4. *The problem k-DCC is $\mathcal{NP}$-complete for any $k \geq 3$.* □

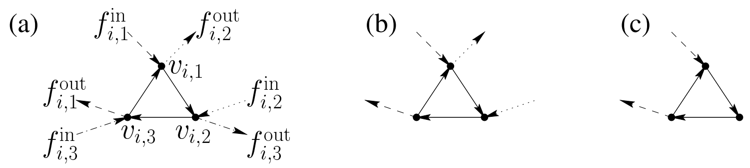

Fig. 5. The clause gadget G_i if c_i consists of (a) three, (b) two, or (c) one literal. The dashed, dotted, and dash-dotted edges correspond to the first, second, and third literal of c_i, respectively.

4.2 Nonapproximability of Min-4-DCC

In this section we show that Min-4-DCC does not have a polynomial time approximation scheme (PTAS, see e.g. Ausiello et al. [2]), unless $\mathcal{NP} = \mathcal{P}$. For this purpose we reduce Max-3SAT(3) to this problem. An instance of Max-3SAT is a set F of disjunctive clauses where each clause consists of at most three literals. Max-3SAT is the problem of finding the maximum number of simultaneously satisfiable clauses. Max-3SAT(3) is the restricted version where each variable occurs at most three times in F. We may assume that each variable occurs at least once positive and at least once negative. Otherwise, we can eliminate this variable by setting it to the appropriate value. In particular, each variable occurs twice or three times. Papadimitriou and Yannakakis [16] have shown that Max-3SAT(3) is MAX $\mathcal{SNP}$-complete. They have presented a reduction from Max-3SAT to Max-3SAT(3) using so called regular expanders (see e.g. Ajtai [1]). A set F of clauses will be called η-satisfiable iff $\eta \cdot |F|$ is the maximum number of satisfiable clauses in F. Håstad [10] has proven that it is $\mathcal{NP}$-hard to distinguish 1- and $(7/8+\epsilon)$-satisfiable instances of Max-3SAT for any $\epsilon > 0$. The reduction of Papadimitriou and Yannakakis and the result of Håstad yield the following lemma.

Lemma 5. *There exists a constant $\lambda < 1$ such that it is $\mathcal{NP}$-hard to distinguish* 1- *and λ-satisfiable instances of* Max-3SAT(3). □

We reduce Max-3SAT(3) to Min-4-DCC. For this purpose, let $F = \{c_1, \ldots, c_t\}$ be a set of disjunctive clauses over variables $U = \{x_1, \ldots, x_r\}$. We construct a graph $G = (V, E)$. For each variable x_j we have one node $u_j \in V$. These nodes will be called *variable nodes*. For each clause c_i we have three nodes $v_{i,1}, v_{i,2}, v_{i,3} \in V$. Let $V_i = \{v_{i,1}, v_{i,2}, v_{i,3}\}$ be the set of these nodes.

In the following we describe how the nodes of G are connected via edges with weight one. All other edges have weight two. The nodes in V_i are connected via a cycle as shown in Fig. 5. The subgraph induced by V_i will be called the *clause gadget G_i*. The clause gadgets and the variable nodes are connected as follows. Each variable node u_j has two incoming and two outgoing edges $e_{j,+}^{\mathrm{in}}$, $e_{j,+}^{\mathrm{out}}$ representing the literal x_j, and $e_{j,-}^{\mathrm{in}}$, $e_{j,-}^{\mathrm{out}}$ representing $\overline{x}_j$ as depicted in Fig. 6a. If c_i is the first clause where the literal x_j appears in, then the edge $e_{j,+}^{\mathrm{out}}$ is identical with either $f_{i,1}^{\mathrm{in}}$, $f_{i,2}^{\mathrm{in}}$, or $f_{i,3}^{\mathrm{in}}$ depending on where x_j occurs in c_i. If a literal occurs in more than one clause then one of the outgoing edges of the first gadget and one of the incoming edges of the second gadget are identical according to where the literal appears in these clauses. If c_i is the last clause where the literal x_j appears in, then the edge $e_{j,+}^{\mathrm{in}}$ is identical with either $f_{i,1}^{\mathrm{out}}$,

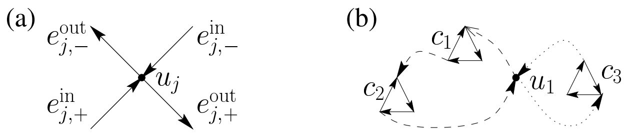

Fig. 6. (a) The edges starting and ending at u_j. (b) Example: x_1 is the first literal in both c_1 and c_2, and $\overline{x}_1$ is the second literal in c_3.

$f^{\text{out}}_{i,2}$, or $f^{\text{out}}_{i,3}$ depending on where x_j occurs in c_i. The clauses which contain $\overline{x}_j$ are connected in a similar fashion. An example is shown in Figure 6b.

Let $C = (V, E_C)$ be a cycle cover of G. We introduce a weight $\nu_C\big(\tilde{V}\big)$ for subsets $\tilde{V} \subseteq V$ of nodes. For a single node $z \in V$ the weight $\nu_C\big(\{z\}\big)$ of z with respect to the cycle cover C is half of the weight of its incoming edge plus half of the weight of its outgoing edge. For $\tilde{V} \subseteq V$ we have $\nu_C\big(\tilde{V}\big) = \sum_{z \in \tilde{V}} \nu_C\big(\{z\}\big)$. Then $\nu_C\big(V\big) = w(C)$.

Let an assignment for the variables U be given that satisfies k of the t clauses of F. We construct a 4-cycle cover C with weight $w(C) = r + 4 \cdot t - k$ as follows. If $x_j = \texttt{true}$ then $e^{\text{in}}_{j,+}, e^{\text{out}}_{j,+} \in E_C$, otherwise $e^{\text{in}}_{j,-}, e^{\text{out}}_{j,-} \in E_C$. If the j-th literal of c_i is $\texttt{true}$ then $f^{\text{in}}_{i,j}, f^{\text{out}}_{i,j} \in E_C$. For any satisfied clause c_i add some of the edges $\big(v_{i,1}, v_{i,2}\big)$, $\big(v_{i,2}, v_{i,3}\big)$, and $\big(v_{i,3}, v_{i,1}\big)$ if necessary. The clauses that are not satisfied are connected in one big cycle. This yields weight 4 per unsatisfied clause. Every node in C has indegree 1 and outdegree 1. Hence C is a cycle cover. If a clause c_i is satisfied by only one literal, then the cycle passing G_i contains two edges within G_i and both one incoming and one outgoing edge. If a clause is satisfied by more than one literal, then the cycle passes at least two variable nodes. Thus, the obtained cycle cover is a 4-cycle cover. The case in which only one clause is not satisfied by the assignment is a special one since then we cannot form one big loop of the unsatisfied clauses. But this case is negligible for sufficiently large instances of Max-3SAT(3), since we are interested in approximability.

For the other direction consider a 4-cycle cover C of G and a clause gadget G_i with $\nu_C\big(V_i\big) = 3$. Such a gadget will be called *satisfying*. Any clause gadget that is not satisfying yields weight at least $7/2$. It holds that G_i is satisfying iff all edges that start or end in G_i have weight one. Since all cycles must have at least length 4 we have the following lemma.

Lemma 6. *Let $C = (V, E_C)$ be an arbitrary 4-cycle cover of G and G_i be a satisfying clause gadget in C. Then the following properties hold:*

1. *At least two of the edges $f^{\text{in}}_{i,1}, f^{\text{out}}_{i,1}, \ldots, f^{\text{out}}_{i,3}$ are in E_C.*
2. *For $j = 1, 2, 3$: $f^{\text{in}}_{i,j} \in E' \;\Leftrightarrow\; f^{\text{out}}_{i,j} \in E'$.* □

A satisfying clause gadget G_i yields a partial assignment for the variables of c_i. If $f^{\text{in}}_{i,j}, f^{\text{out}}_{i,j} \in E'$ then the j-th literal of c_i is set $\texttt{true}$ and the corresponding variable is assigned an appropriate value, otherwise we do not assign a value to this literal. Due to Lemma 6 the obtained partial assignment satisfies c_i.

By considering all satisfying clause gadgets we step by step obtain a partial assignment that satisfies at least those clauses whose gadgets are satisfying. The following lemma assures that the obtained partial assignment is consistent, i.e. we never have to assign both `true` and `false` to one variable.

Lemma 7. *Let C be an arbitrary 4-cycle cover of G. There are no two satisfying clause gadgets G_i and $G_{i'}$ and a variable x_j such that x_j has to be assigned different values according to these clause gadgets.* □

Proof. If x_j has to be assigned different values then x_j occurs positive in c_i and negative in $c_{i'}$ or vice versa. We only consider the first case. By symmetry, we can restrict ourselves to the case where the literal $\overline{x}_j$ occurs exactly once in F and that x_j is the first variable in both clauses. According to $G_{i'}$ the literal $\overline{x}_j$ has to be assigned `true`. Since $c_{i'}$ is the only clause that contains $\overline{x}_j$, the two edges $e_{j,-}^{\mathrm{in}} = f_{i',1}^{\mathrm{out}}$ and $e_{j,-}^{\mathrm{out}} = f_{i',1}^{\mathrm{in}}$ belong to E_C. On the other hand, $f_{i,1}^{\mathrm{out}}$ and $f_{i,1}^{\mathrm{in}}$ belong to E_C. Since x_j occurs at most twice positive in F at least one of the edges $f_{i,1}^{\mathrm{out}}$ and $f_{i,1}^{\mathrm{in}}$ connects V_i to u_j. Thus, u_j has indegree at least 2 or outdegree at least 2, a contradiction. □

Each variable node yields at least weight 1. Each satisfying clause gadget yields weight 3. All other clause gadgets yield at least weight $7/2$.

The following theorem proves that a constant $\xi > 1$ exists such that Min-4-DCC cannot be approximated in polynomial time with performance ratio ξ, unless $\mathcal{NP} = \mathcal{P}$. Thus, Min-4-DCC does not have a PTAS, unless $\mathcal{NP} = \mathcal{P}$.

Theorem 5. *There exists a constant $\xi > 1$ such that it is $\mathcal{NP}$-hard to distinguish instances $G = (V, E)$ of* Min-4-DCC *whose minimum cycle cover has weight $|V|$ and instances whose minimum cycle cover has at least weight $\xi \cdot |V|$.*

Proof. Due to the reduction described above, a 1-satisfiable instance of Max-3SAT(3) yields a graph which has a 4-cycle cover with weight $|V| = r + 3 \cdot t$. On the other hand, every 4-cycle cover of a graph corresponding to a λ-satisfiable instance has at least weight $r + 3 \cdot \lambda \cdot t + (7/2) \cdot (1 - \lambda) \cdot t = K_\lambda$. Since every clause consists of at most three literals and every variable appears at least twice we have $r/t \leq 3/2$. Therefore, the following inequality holds:

$$\frac{K_\lambda}{r + 3 \cdot t} \geq \frac{(3/2 + 3 \cdot \lambda + (7/2) \cdot (1 - \lambda)) \cdot t}{(3/2 + 3) \cdot t} = \frac{10 - \lambda}{9} = \xi > 1.$$

Thus, deciding whether the minimum 4-cycle cover of a graph has weight $|V|$ or at least weight $\xi \cdot |V|$ is at least as hard as distinguishing 1- and λ-satisfiable instances of Max-3SAT(3). This completes the proof due to Lemma 5. □

If we replace the variable nodes by paths of lengths $k - 4$ we obtain the result that Min-k-DCC does not have a PTAS for any $k \geq 4$.

Theorem 6. *For any $k \geq 4$ there exists a constant $\xi_k > 1$ such that* Min-k-DCC *cannot be approximated with performance ratio ξ_k, unless $\mathcal{NP} = \mathcal{P}$.* □

We can transform a directed graph into an undirected graph by replacing each node with three nodes (see e.g. Hopcroft and Ullman [11]). Applying this transformation to the graph constructed in this section we obtain the following theorem.

Theorem 7. Min-k-UCC *does not have a PTAS for any $k \geq 12$, unless $\mathcal{NP} = \mathcal{P}$.* □

5 Conclusions and open problems

We have presented factor $4/3$ and $7/6$ approximation algorithms for Min-k-DCC and Min-k-UCC, respectively, with polynomial running time (independent of k). On the other hand, we have shown that k-DCC is $\mathcal{NP}$-complete for $k \geq 3$ and Min-k-DCC does not possess a PTAS for $k \geq 4$, unless $\mathcal{NP} = \mathcal{P}$. The status of Min-3-DCC is open. We strongly conjecture that this problem also has no PTAS, unless $\mathcal{NP} = \mathcal{P}$. In the undirected case, Papadimitriou has shown $\mathcal{NP}$-hardness of k-UCC for $k \geq 6$. The complexity of 5-UCC and the approximability of Min-k-UCC for $5 \leq k \leq 11$ remains open.

References

1. M. Ajtai. Recursive construction for 3-regular expanders. *Combinatorica*, 14(4):379–416, 1994.
2. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation*. Springer, 1999.
3. G. Cornuéjols and W. Pulleyblank. A matching problem with side constraints. *Discrete Math.*, 29:135–159, 1980.
4. J. Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.
5. L. Engebretsen and M. Karpinski. Approximation hardness of TSP with bounded metrics. Technical Report 00-089, Electronic Colloquium on Comput. Complexity (ECCC), 2000.
6. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to NP-Completeness*. W. H. Freeman and Company, 1979.
7. R. L. Graham, M. Grötschel, and L. Lovász, editors. *Handbook of Combinatorics*, volume 1. Elvsevier, 1995.
8. D. Hartvigsen. *An Extension of Matching Theory*. PhD thesis, Carnegie-Mellon University, 1984.
9. D. Hartvigsen. The square-free 2-factor problem in bipartite graphs. In *7th Int. Conf. on Integer Programming and Combinatorial Optimization (IPCO)*, volume 1620 of *Lecture Notes in Comput. Sci.*, pages 234–241. Springer, 1999.
10. J. Håstad. Some optimal inapproximability results. In *Proc. 29th Ann. Symp. on Theory of Comput. (STOC)*, pages 1–10. ACM, 1997.
11. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
12. S. R. Kosaraju, J. K. Park, and C. Stein. Long tours and short superstrings. In *Proc. 35th Ann. Symp. on Foundations of Comput. Sci. (FOCS)*, pages 166–177. IEEE, 1994.
13. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors. *The Traveling Salesman Problem*. Wiley, 1985.
14. L. Lovász and M. D. Plummer. *Matching Theory*. Elsevier, 1986.
15. C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.
16. C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. System Sci.*, 43(3):425–440, 1991.
17. C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Math. Oper. Res.*, 18:1–11, 1993.
18. W. T. Tutte. A short proof of the factor theorem for finite graphs. *Canad. J. Math.*, 6:347–352, 1954.
19. S. Vishwanathan. An approximation algorithm for the asymmetric travelling salesman problem with distances one and two. *Inform. Process. Lett.*, 44:297–302, 1992.