On Approximating Restricted Cycle Covers*

Bodo Manthey**

Universität zu Lübeck, Institut für Theoretische Informatik
Ratzeburger Allee 160, 23538 Lübeck, Germany
manthey@tcs.uni-luebeck.de

Abstract. A cycle cover of a graph is a set of cycles such that every vertex is part of exactly one cycle. An L-cycle cover is a cycle cover in which the length of every cycle is in the set L. A special case of L-cycle covers are k-cycle covers for $k \in \mathbb{N}$, where the length of each cycle must be at least k. The weight of a cycle cover of an edge-weighted graph is the sum of the weights of its edges.
We come close to settling the complexity and approximability of computing L-cycle covers. On the one hand, we show that for almost all L, computing L-cycle covers of maximum weight in directed and undirected graphs is APX-hard and NP-hard. Most of our hardness results hold even if the edge weights are restricted to zero and one. On the other hand, we show that the problem of computing L-cycle covers of maximum weight can be approximated with factor 2.5 for undirected graphs and with factor 3 in the case of directed graphs. Finally, we show that 4-cycle covers of maximum weight in graphs with edge weights zero and one can be computed in polynomial time.
As a by-product, we show that the problem of computing minimum vertex covers in λ-regular graphs is APX-complete for every $\lambda \geq 3$.

1 Introduction

The *travelling salesman problem* (TSP) is perhaps the best-known combinatorial optimisation problem. An instance of the TSP is a complete graph with edge weights, and the aim is to find a minimum or maximum weight cycle that visits every vertex exactly once. Such a cycle is called a *Hamiltonian cycle*. Since the TSP is NP-hard [10, ND22+23], we cannot hope to always find an optimal cycle efficiently. For practical purposes, however, it is often sufficient to obtain a cycle that is close to optimal. In such cases, we require approximation algorithms, i.e. polynomial-time algorithms that compute such near-optimal cycles.

The problem of computing *cycle covers* is a relaxation of the TSP: A cycle cover of a graph is a spanning subgraph such that every vertex is part of exactly one simple cycle. Thus, a solution to the TSP is a cycle cover consisting of a single cycle. In analogy to the TSP, the weight of a cycle cover in an edge-weighted graph is the sum of the weights of its edges.

* A full version of this work is available at http://arxiv.org/abs/cs/0504038.
** Supported by DFG research grant RE 672/3.

3rd Workshop on Approximation and Online Algo. (WAOA 2005)

In contrast to the TSP, cycle covers of maximum weight can be computed efficiently. This fact is exploited in approximation algorithms for the TSP; the computation of cycle covers forms the basis for the currently best known approximation algorithms for many variations of the TSP. These algorithms usually start by computing an initial cycle cover and then join cycles to obtain a Hamiltonian cycle.

Short cycles in a cycle cover limit the approximation ratios achieved by such algorithms. In general, the longer the cycles in the initial cover are, the better the approximation ratio. Thus, we are interested in computing cycle covers without short cycles. Moreover, there are approximation algorithms that behave particularly well if the cycle covers that are computed do not contain cycles of odd length [6]. Finally, some so-called vehicle routing problems (cf. e.g. Hassin and Rubinstein [12]) require covering vertices with cycles of bounded length.

Therefore, we consider *restricted cycle covers*, where cycles of certain lengths are ruled out a priori: Let $L \subseteq \mathbb{N}$, then an *L-cycle cover* is a cycle cover in which the length of each cycle is in L. To fathom the possibility of designing approximation algorithms based on computing cycle covers, we aim to characterise the sets L for which L-cycle covers of maximum weight can be computed efficiently.

1.1 Preliminaries

A **cycle cover** of a graph $G = (V, E)$ is a subgraph of G that consists solely of cycles such that all vertices in V are part of exactly one cycle. The length of a cycle is the number of edges it consists of. We are concerned with simple graphs, i.e. the graphs do not contain multiple edges or loops. Thus, the shortest cycles of undirected and directed graphs have length three and two, respectively.

An ***L*-cycle cover** is a cycle cover in which the length of every cycle is in the set $L \subseteq \mathbb{N}$. For undirected graphs, we have $L \subseteq \mathcal{U} = \{3, 4, 5, \ldots\}$, while $L \subseteq \mathcal{D} = \{2, 3, 4, \ldots\}$ in case of directed graphs. A ***k*-cycle cover** is a $\{k, k+1, \ldots\}$-cycle cover. Let $\overline{L} = \mathcal{U} \setminus L$ in the case of undirected graphs and $\overline{L} = \mathcal{D} \setminus L$ in the case of directed graphs (this will be clear from the context).

Given an edge weight function $w : E \to \mathbb{N}$, the **weight $w(C)$** of a subset $C \subseteq E$ of the edges of G is $w(C) = \sum_{e \in C} w(e)$. This particularly defines the weight of a cycle cover since we view cycle covers as sets of edges. Let $U \subseteq V$ be any subset of the vertices of G. The **internal edges of U** are all edges of G that have both vertices in U. We denote by **$w_U(C)$** the sum of the weights of all internal edges of U in C. The **external edges at U** are all edges of G with exactly one vertex in U.

For $L \subseteq \mathcal{U}$, the set ***L*-UCC** contains all undirected graphs that have an L-cycle cover as spanning subgraph.

Max-*L*-UCC is the following optimisation problem: Given a complete undirected graph with edge weights zero and one, find an L-cycle cover of maximum weight. We can also consider the graph as being not complete and without edge weights. Then we try to find an L-cycle cover with a minimum number of "non-edges" ("non-edges" correspond to weight zero edges, edges to weight one edges). Thus, Max-L-UCC can be viewed as a generalisation of L-UCC.

Max-W-L-UCC is the problem of finding maximum-weight L-cycle covers in graphs with arbitrary non-negative edge weights.

For $k \geq 3$, **k-UCC**, **Max-k-UCC**, and **Max-W-k-UCC** are defined like L-UCC, Max-L-UCC and Max-W-L-UCC except that k-cycle covers instead of L-cycle covers are sought.

L-DCC, **Max-L-DCC**, **Max-W-L-DCC**, **k-DCC**, **Max-k-DCC**, and **Max-W-k-DCC** are defined for directed graphs like L-UCC, Max-L-UCC, Max-W-L-UCC, k-UCC, Max-k-UCC, and Max-W-k-UCC for undirected graphs except that $L \subseteq \mathcal{D}$ and $k \geq 2$.

An instance of **Min-Vertex-Cover** is an undirected graph $H = (X, F)$. A vertex cover of H is a subset $\tilde{X} \subseteq X$ such that at least one vertex of every edge in F is in $\tilde{X}$. The aim is to find a vertex cover of minimum cardinality. **Min-Vertex-Cover(λ)** is Min-Vertex-Cover restricted to λ-regular graphs, i.e. to simple graphs in which every vertex is incident to exactly λ edges. Already Min-Vertex-Cover(3) is APX-complete [2].

We refer to Ausiello et al. [3] for a survey on NP optimisation problems.

1.2 Existing Results

Undirected Graphs. $\mathcal{U}$-UCC, Max-$\mathcal{U}$-UCC, and Max-W-$\mathcal{U}$-UCC can be solved in polynomial time via reduction to the classical perfect matching problem, which can be solved in polynomial time [1, Chap. 12]. Hartvigsen presented a polynomial-time algorithm for computing a *maximum-cardinality triangle-free two-matching* [11] (see also Sect. 5). His algorithm can be used to decide 4-UCC in polynomial time. Furthermore, it can be used to approximate Max-4-UCC within an additive error of one according to Bläser [4].

Max-W-k-UCC admits a simple factor $3/2$ approximation for all k: Compute a maximum weight cycle cover, break the lightest edge of each cycle, and join the cycles to obtain a Hamiltonian cycle, which is sufficiently long if the graph contains at least k vertices. Unfortunately, this algorithm cannot be generalised to work for Max-W-L-UCC with arbitrary L. For the problem of computing k-cycle covers of minimum weight in graphs with edge weights one and two, there exists a factor $7/6$ approximation algorithm for all k [8].

Cornuéjols and Pulleyblank presented a proof due to Papadimitriou that 6-UCC is NP-complete [9]. Vornberger showed that Max-W-5-UCC is NP-hard [14]. For $k \geq 7$, Max-k-UCC and Max-W-k-UCC are APX-complete [5]. Hell et al. [13] proved that L-UCC is NP-hard for $\overline{L} \not\subseteq \{3, 4\}$.

For most L, L-UCC, Max-L-UCC, and Max-W-L-UCC are not even recursive since there are uncountably many L. Thus for most L, L-UCC is not in NP and Max-L-UCC and Max-W-L-UCC are not in NPO. This does not matter for hardness results but may cause problems when one wants to design approximation algorithms that base on computing L-cycle covers. However, our approximation algorithms work for arbitrary L, independently of the complexity of L.

Directed Graphs. $\mathcal{D}$-DCC, Max-$\mathcal{D}$-DCC, and Max-W-$\mathcal{D}$-DCC can be solved in polynomial time by reduction to the maximum weight perfect matching problem

	L-UCC	Max-L-UCC	Max-W-L-UCC
$\overline{L} = \emptyset$	in P	in PO	in PO
$\overline{L} = \{3\}$	in P	in PO	
$\overline{L} = \{4\} \vee \overline{L} = \{3,4\}$			APX-complete
else	NP-hard	APX-hard	APX-hard

(a) Undirected cycle covers.

	L-DCC	Max-L-DCC	Max-W-L-DCC
$L = \{2\} \vee L = \mathcal{D}$	in P	in PO	in PO
else	NP-hard	APX-hard	APX-hard

(b) Directed cycle covers.

Table 1. The complexity of computing L-cycle covers.

in bipartite graphs [1, Chap. 12]. But already 3-DCC is NP-complete [10, GT13]. Max-k-DCC and Max-W-k-DCC are APX-complete for all $k \geq 3$ [5].

Similar to the factor 3/2 approximation algorithm for undirected cycle covers, Max-W-k-DCC has a simple factor 2 approximation algorithm for all k: Compute a maximum weight cycle cover, break the lightest edge of every cycle, and join the cycles to obtain a Hamiltonian cycle. Again, this algorithm cannot be generalised to work for Max-W-L-DCC with arbitrary L. There are a factor 4/3 approximation algorithm for Max-W-3-DCC [7] and a factor 3/2 approximation algorithm for Max-k-DCC for $k \geq 3$ [5].

As in the case of cycle covers in undirected graphs, for most L, L-DCC, Max-L-DCC, and Max-W-L-DCC are not recursive.

1.3 New Results

We come close to settling the complexity and approximability of restricted cycle covers. Only the complexity of the five problems 5-UCC, $\overline{\{4\}}$-UCC Max-5-UCC, Max-$\overline{\{4\}}$-UCC, and Max-W-4-UCC remains open. Table 1 shows an overview on the complexity of computing restricted cycle covers.

Hardness Results. We prove that Max-L-UCC is APX-hard for all L with $\overline{L} \not\subseteq \{3,4\}$ (Sect. 3). We also prove that Max-W-L-UCC is APX-hard if $\overline{L} \not\subseteq \{3\}$ (this follows from the results of Sect. 3 and the APX-completeness of Max-W-5-UCC and Max-W-$\overline{\{4\}}$-UCC shown in Sect. 2). The hardness results for Max-W-L-UCC hold even if we allow only the edge weights zero, one, and two.

We show a dichotomy for cycle covers of directed graphs: For all L with $L \neq \{2\}$ and $L \neq \mathcal{D}$, L-DCC is NP-hard (Theorem 6) and Max-L-DCC and Max-W-L-DCC are APX-hard (Theorem 5), while it is known that all three problems are solvable in polynomial time if $L = \{2\}$ or $L = \mathcal{D}$.

To show the hardness of directed cycle covers, we show that certain kinds of graphs, called *L-clamps*, exist for non-empty $L \subseteq \mathcal{D}$ if and only if $L \neq \mathcal{D}$ (Theorem 4). This graph-theoretical result might be of independent interest.

As a by-product, we prove that Min-Vertex-Cover(λ) is APX-complete for all $\lambda \geq 3$ (Sect. 6). We need this result for the APX-hardness proofs in Sect. 3.

Algorithms. We present a polynomial-time factor 2.5 approximation algorithm Max-W-L-UCC and a factor 3 approximation algorithm for Max-W-L-DCC (Sect. 4). Both algorithms work for arbitrary L.

Finally, we prove that Max-4-UCC is solvable in polynomial time (Sect. 5).

2 A Generic Reduction for L-Cycle Covers

In this section, we present a generic reduction from Min-Vertex-Cover(3) to Max-L-UCC or Max-W-L-UCC. To instantiate the reduction for a certain L, we use a small graph, which we call *gadget*, the specific structure of which depends on L. Such a gadget together with the generic reduction is an L-reduction from Min-Vertex-Cover(3) to Max-L-UCC or Max-W-L-UCC. The mere aim is to prove the APX-hardness of Max-W-$\overline{\{4\}}$-UCC and Max-W-5-UCC.

2.1 The Generic Reduction

Let $H = (X, F)$ be a cubic graph with vertex set X and edge set F as an instance of Min-Vertex-Cover(3). Let $n = |X|$ and $m = |F| = 3n/2$. We construct an undirected complete graph G with edge weight function w as a generic instance of Max-L-UCC or Max-W-L-UCC.

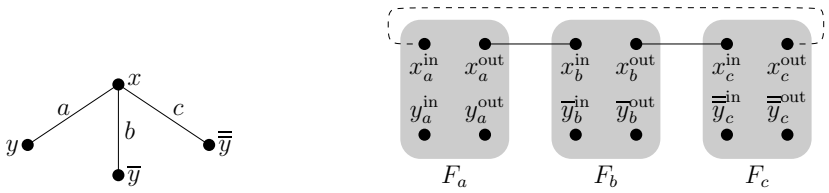
For each edge $a = \{x, y\} \in F$, we construct a subgraph F_a of G called the **gadget of a**. We define F_a as set of vertices, thus $w_{F_a}(C)$ for a subset C of the edges of G is well defined. This gadget contains four distinguished vertices x_a^{in}, x_a^{out}, y_a^{in}, and y_a^{out}. These four vertices are used to connect F_a to the rest of the graph. What such a gadget looks like depends on L. If all edges in such a gadget have weight zero or one, we obtain an instance of Max-L-UCC since all edges between different gadgets will have weight zero or one. Otherwise, we have an instance of Max-W-L-UCC. Figure 2 shows an example of such a gadget.

Let $a, b, c \in F$ be the three edges incident to vertex $x \in X$ (the order is arbitrary). Then we assign weight one to the edges connecting x_a^{out} to x_b^{in} and x_b^{out} to x_c^{in} and weight zero to the edge connecting x_c^{out} to x_a^{in}. We call the three edges $\{x_a^{\text{out}}, x_b^{\text{in}}\}$, $\{x_b^{\text{out}}, x_c^{\text{in}}\}$, and $\{x_c^{\text{out}}, x_a^{\text{in}}\}$ the **junctions of x**. We say that $\{x_a^{\text{out}}, x_b^{\text{in}}\}$ and $\{x_c^{\text{out}}, x_a^{\text{in}}\}$ are the junctions of x **at F_a**. Figure 1 shows an example.

We call an edge **illegal** if it connects two different gadgets but is not a junction. Thus, an illegal edge is an external edge at two different gadgets. All illegal edges have weight zero, i.e. there are no edges of weight one that connect two different gadgets except for the junctions. The weights of the internal edges of the gadgets depend on the gadget, which in turn depends on L.

The following terms are defined for arbitrary subsets C of the edges of G, and so in particular for L-cycle covers. We say that **C legally connects F_a** if

- C contains no illegal edges incident to F_a,



(a) Vertex x and its edges.

(b) The gadgets F_a, F_b, and F_c and their connections via the three junctions of x. The dashed edge has weight zero. Other weight zero edges and the junctions of y, $\overline{y}$, and $\overline{\overline{y}}$ are not shown.

Fig. 1. The construction for a vertex $x \in X$ incident to $a, b, c \in F$.

- C contains exactly two or four junctions at F_a, and
- if C contains exactly two junctions at F_a, then these belong to the same vertex $x \in a$.

We call C **legal** if C legally connects all gadgets. If C is legal, then for all $x \in X$, either all junctions of x are in C or no junction of x is in C. Furthermore, from a legal set C we obtain a vertex cover $\tilde{X} = \{x \mid \text{the junctions of } x \text{ are in } C\}$.

Let us now define the requirements the gadgets must fulfil. In the following, let C be an arbitrary L-cycle cover of G and $a = \{x, y\} \in F$ be an arbitrary edge of H.

R0: There exists a fixed number $s \in \mathbb{N}$, which we call the **gadget parameter**, that depends only on the gadget. The role of the gadget parameter will become clear in the subsequent requirements.
R1: $w_{F_a}(C) \leq s - 1$.
R2: If C contains 2α external edges at F_a, then $w_{F_a}(C) \leq s - \alpha$.
R3: If C contains exactly one junction of x at F_a and exactly one junction of y at F_a, then $w_{F_a}(C) \leq s - 2$. (In this case, C does not legally connect F_a.)
R4: Let C' be an arbitrary subset of the edges of G that legally connects F_a. Assume that there are 2α junctions ($\alpha \in \{1, 2\}$) at F_a in C'.
Then there exists a C'' with the following properties:
  - C'' differs from C' only in F_a's internal edges and
  - $w_{F_a}(C'') = s - \alpha$.

Thus, given C', C'' can be obtained by locally modifying C' within F_a. We call the process of obtaining C'' from C' **rearranging C' in F_a**.
R5: Let C' be a legal subset of the edges of G. Then there exists a subset $\tilde{C}$ of edges obtained by rearranging all gadgets as described in R4 such that $\tilde{C}$ is an L-cycle cover.

The requirements assert that connecting the gadgets legally is never worse than connecting them illegally. This yields the main result of this section.

Lemma 1. *Assume that a gadget as described exists for $L \subseteq \mathcal{U}$. Then the reduction presented is an L-reduction from* Min-Vertex-Cover(3) *to* Max-W-L-UCC. *If the gadget contains only edges of weight zero or one, then the reduction is an L-reduction from* Min-Vertex-Cover(3) *to* Max-L-UCC.

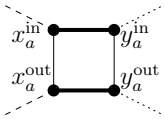
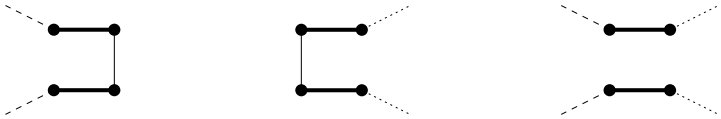


Fig. 2. The edge gadget F_a for an edge $a = \{x, y\}$ that is used to prove the APX-completeness of Max-W-5-UCC. Bold edges are internal edges of weight two, solid edges are internal edges of weight one, internal edges of weight zero are not shown. The dashed and dotted edges are the junctions of x and y, respectively, at F_a.



(a) $x \in \tilde{X}$. (b) $y \in \tilde{X}$. (c) $x, y \in \tilde{X}$.

Fig. 3. Traversals of the gadget for Max-W-5-UCC that achieve maximum weight.

2.2 Max-W-5-UCC and Max-W-$\overline{\{4\}}$-UCC

The gadget for Max-W-5-UCC is shown in Fig. 2. Let G be the graph constructed via the reduction presented in Sect. 2.1 with the gadget of this section. Let C be an arbitrary L-cycle cover of G and $a = \{x, y\} \in F$. By proving that it fulfils all requirements, we obtain the following result.

Theorem 1. Max-W-5-UCC *is* APX*-hard, even if the edge weights are restricted to be zero, one, or two.*

Although the status of Max-5-UCC is still open, allowing only one additional edge weight of two already yields an APX-complete problem.

The generic reduction together with the gadget used for Max-W-5-UCC works also for Max-W-$\overline{\{4\}}$-UCC. The gadget only requires that cycles of length four are forbidden since otherwise R1 is not satisfied. Thus, all requirements are fulfilled for Max-W-$\overline{\{4\}}$-UCC in exactly the same way as for Max-W-5-UCC.

Theorem 2. Max-W-$\overline{\{4\}}$-UCC *is* APX*-hard, even if the edge weights are restricted to be zero, one, or two.*

3 A Uniform Reduction for L-Cycle Covers

3.1 Clamps

We now define so-called *clamps*, which were introduced by Hell et al. [13]. Clamps are crucial for the hardness proof presented in this section.

Let $K = (V, E)$ be an undirected graph, let $u, v \in V$ be two vertices of K, and let $L \subseteq \mathcal{U}$. We denote by K_{-u} and K_{-v} the graphs obtained from K by deleting u and v, respectively, and their incident edges. Moreover, K_{-u-v} denotes the

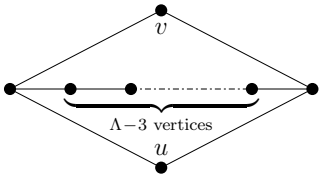


Fig. 4. An L-clamp for finite L with $\max(L) = \Lambda$.

graph obtained from K by deleting both u and v. Finally, for $k \in \mathbb{N}$, K^k is the following graph: Let $y_1, \ldots, y_k$ be vertices with $y_i \notin V$, add edges $\{u, y_1\}$, $\{y_i, y_{i+1}\}$ for $1 \leq i \leq k-1$, and $\{y_k, v\}$. For $k = 0$, we directly connect u to v.

The graph K is called an ***L*-clamp** if the following properties hold:

- Both K_{-u} and K_{-v} contain an L-cycle cover.
- Neither K nor K_{-u-v} nor K^k for any $k \in \mathbb{N}$ contains an L-cycle cover.

We call u and v the **connectors** of the L-clamp K.

Lemma 2 (Hell et al. [13]). *Let $L \subseteq \mathcal{U}$ be non-empty. Then there exists an L-clamp if and only if $\overline{L} \not\subseteq \{3, 4\}$.*

Figure 4 shows an example of an L-clamp for finite L.

If there exists an L-clamp for some L, then we can assume that the connectors u and v both have degree two since we can remove all edges that are not used in the L-cycle covers of K_{-v} and K_{-u}.

For our purpose, consider any non-empty set $L \subseteq \{3, 4, 5, \ldots\}$ with $\overline{L} \not\subseteq \{3, 4\}$. We fix one L-clamp K with connectors $u, v \in V$ arbitrarily and refer to it in the following as the L-clamp, although there exists more than one L-clamp. Let σ be the number of vertices of K.

We are concerned with edge-weighted graphs. Therefore, we transfer the notion of clamps to graphs with edge weights zero and one in the obvious way: Let G be an undirected complete graph with vertex set V and edge weights zero and one and let K be an L-clamp. Let $U \subseteq V$. We say that U is an L-clamp with connectors $u, v \in U$ if the subgraph of G induced by U restricted to the edges of weight one is isomorphic to K with u and v mapped to connectors of K.

3.2 The Reduction

Let $L \subseteq \mathcal{U}$ be non-empty with $\overline{L} \not\subseteq \{3, 4\}$. Thus, L-clamps exist and we fix one as in the previous section. Let σ be the number of vertices in the L-clamp. Let $\lambda = \min(L)$. (This choice is arbitrary. We could choose any number in L.) We will reduce Min-Vertex-Cover(λ) to Max-L-UCC. Min-Vertex-Cover(λ) is APX-complete since $\lambda \geq 3$ (see Sect. 6).

Let $H = (X, F)$ be an instance of Min-Vertex-Cover(λ) with $n = |X|$ vertices and $m = |F| = \lambda n/2$ edges. Our instance G for Max-L-UCC consists of λ subgraphs $G_1, \ldots, G_\lambda$, each containing $2\sigma m$ vertices. We start by describing G_1.

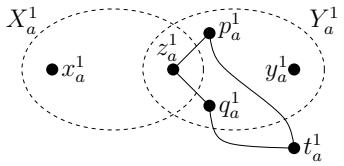


Fig. 5. The edge gadget for $a = \{x, y\}$ consisting of two L-clamps. The vertex z_a^1 is the only vertex that belongs to both clamps X_a^1 and Y_a^1.

Then we state the differences between G_1 and $G_2, \ldots, G_\lambda$ and say to which edges between these graphs we assign weight one.

Let $a = \{x, y\} \in F$ be any edge of H. We construct an edge gadget F_a for a that consists of two L-clamps X_a^1 and Y_a^1 and one additional vertex t_a^1 as shown in Fig. 5. The connectors of X_a^1 are x_a^1 and z_a^1 while the connectors of Y_a^1 are y_a^1 and z_a^1, i.e. X_a^1 and Y_a^1 share the connector z_a^1. Let p_a^1 and q_a^1 be the two unique vertices in Y_a^1 that share a weight one edge with z_a^1. (The choice of Y_a^1 is arbitrary, we could choose the corresponding vertices in X_a^1 as well.) We assign weight one to both $\{p_a^1, t_a^1\}$ and $\{q_a^1, t_a^1\}$. Thus, the vertex t_a^1 can also serve as a connector for Y_a^1.

Now let $x \in X$ be any vertex of H and let $a_1, \ldots, a_\lambda \in F$ be the λ edges that are incident to x. We connect the vertices $x_{a_1}^1, \ldots, x_{a_\lambda}^1$ to form a path by assigning weight one to the edges $\{x_{a_\eta}^1, x_{a_{\eta+1}}^1\}$ for $\eta \in \{1, \ldots, \lambda - 1\}$. Together with edge $\{x_{a_\lambda}^1, x_{a_1}^1\}$, these edges form a cycle of length $\lambda \in L$, but note that $w(\{x_{a_\lambda}^1, x_{a_1}^1\}) = 0$. These λ edges are called the **junctions of x**. The **junctions at F_a** for some $a = \{x, y\} \in F$ are the junctions of x and y that are incident to F_a. Overall, the graph G_1 consists of $2\sigma m$ vertices since every edge gadget consists of 2σ vertices.

The graphs $G_2, \ldots, G_\lambda$ are almost exact copies of G_1. The graph G_ξ, $\xi \in \{2, \ldots, \lambda\}$ has clamps X_a^ξ and Y_a^ξ and vertices $x_a^\xi, y_a^\xi, z_a^\xi, t_a^\xi, p_a^\xi, q_a^\xi$ for each edge $a = \{x, y\} \in F$, just as above. The edge weights are also identical with the single exception that the edge $\{x_{a_\lambda}^\xi, x_{a_1}^\xi\}$ also has weight one. Note that we only use the term "gadget" for the subgraphs of G_1 defined above although almost the same subgraphs occur in $G_2, \ldots, G_\lambda$ as well. Similarly, the term "junction" refers only to an edge in G_1 as defined above.

Finally, we describe how to connect $G_1, \ldots, G_\lambda$ with each other. For every edge $a \in F$, there are λ vertices $t_a^1, \ldots, t_a^\lambda$. These are connected to form a cycle consisting solely of weight one edges, i.e. we assign weight one to all edges $\{t_a^\xi, t_a^{\xi+1}\}$ for $\xi \in \{1, \ldots, \lambda - 1\}$ and to $\{t_a^\lambda, t_a^1\}$. Figure 6 shows an example of the whole construction from the viewpoint of a single vertex.

As in the previous section, we call edges that are not junctions but connect two different gadgets **illegal**. Edges with both vertices in the same gadget are again called internal edges. In addition to junctions, illegal edges, and internal edges, we have a fourth kind of edges: The ***t*-edges** of F_a for $a \in F$ are the two edges $\{t_a^1, t_a^2\}$ and $\{t_a^1, t_a^\lambda\}$. The t-edges are not illegal. All other edges connecting G_1 to G_ξ for $\xi \neq 1$ are illegal.

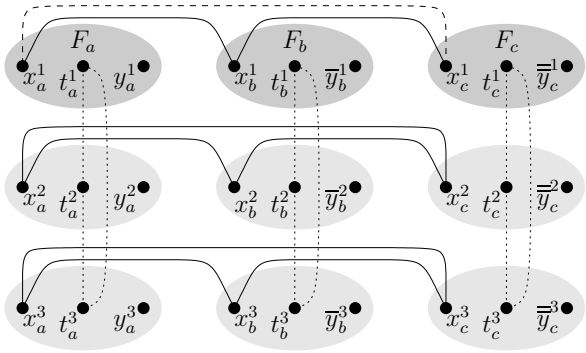


Fig. 6. The construction for a vertex $x \in X$ incident to edges $a, b, c \in F$ for $\lambda = 3$ (Fig. 1(a) on page 6 shows the corresponding graph). The dark grey areas are the edge gadgets F_a, F_b, and F_c. Their copies in G_2 and G_3 are light grey. The cycles connecting the t-vertices are dotted. The cycles connecting the x-vertices are solid, except for the edge $\{x_c^1, x_a^1\}$, which has weight zero and is dashed. The vertices $z_a^1, \ldots, z_c^3$ are not shown for legibility.

Let C be any subset of the edges of the graph G thus constructed, and let $a = \{x, y\} \in F$ be an arbitrary edge of H. We say that $\boldsymbol{C}$ **legally connects** $\boldsymbol{F_a}$ if the following properties are fulfilled:

- C contains no illegal edges incident to F_a and exactly two or four junctions at F_a.
- If C contains exactly two junctions at F_a, then these belong to the same vertex and there are two t-edges at F_a in C.
- If C contains four junctions at F_a, then these are the only external edges in C incident to F_a. In particular, C does not contain t-edges at F_a.

We call C **legal** if C legally connects all gadgets.

We can prove that the construction described above is an L-reduction from Min-Vertex-Cover(λ) to Max-L-UCC for all L with $\overline{L} \not\subseteq \{3, 4\}$.

Theorem 3. *For all $L \subseteq \mathcal{U}$ with $\overline{L} \not\subseteq \{3, 4\}$,* Max-$L$-UCC *is* APX*-hard.*

3.3 Clamps in Directed Graphs

The aim of this section is to introduce directed L-clamps. Let $K = (V, E)$ be a directed graph and $u, v \in V$. Again, K_{-u}, K_{-v}, and K_{-u-v} denote the graphs obtained by deleting u, v, and both u and v, respectively. For $k \in \mathbb{N}$, K_u^k denotes the following graph: Let $y_1, \ldots, y_k \notin V$ be new vertices and add edges $(u, y_1), (y_1, y_2), \ldots, (y_k, v)$. For $k = 0$, we add the edge (u, v). The graph K_v^k is similarly defined, except that we now start at v, i.e. we add the edges $(v, y_1), (y_1, y_2), \ldots, (y_k, u)$. K_v^0 is K with the additional edge (v, u).

Now we can define clamps for directed graphs: Let $L \subseteq \mathcal{D}$. A directed graph $K = (V, E)$ with $u, v \in V$ is a **directed $\boldsymbol{L}$-clamp** with connectors u and v if the following properties hold:

– Both K_{-u} and K_{-v} contain an L-cycle cover.
– Neither K nor K_{-u-v} nor K_u^k nor K_v^k for any $k \in \mathbb{N}$ contains an L-cycle cover.

Theorem 4. *Let $L \subseteq \mathcal{D}$ be non-empty. Then there exists a directed L-clamp if and only if $L \neq \mathcal{D}$.*

3.4 L-Cycle Covers in Directed Graphs

From the hardness results in the previous sections and the work by Hell et al. [13], we obtain the NP-hardness and APX-hardness of L-DCC and Max-L-DCC, respectively, for all L with $2 \notin L$ and $\overline{L} \not\subseteq \{2,3,4\}$: We use the same reduction as for undirected cycle covers and replace every undirected edge $\{u,v\}$ by a pair of directed edges (u,v) and (v,u). However, this does not work if $2 \in L$ and also leaves open the cases when $\overline{L} \subsetneq \{2,3,4\}$. We will show that $L = \{2\}$ and $L = \mathcal{D}$ are the only cases in which directed L-cycle covers can be computed efficiently by proving the NP-hardness of L-DCC and the APX-hardness of Max-L-DCC for all other L. Thus, we settle the complexity for directed graphs.

The APX-hardness of the directed cycle cover problem is obtained by a proof similar to the proof for undirected cycle covers. All we need is a $\lambda \in L$ with $\lambda \geq 3$ and the existence of an L-clamp.

Theorem 5. *Let $L \subseteq \mathcal{D}$ be a non-empty set. If $L \neq \{2\}$ and $L \neq \mathcal{D}$, then* Max-L-DCC *and* Max-W-L-DCC *are* APX*-hard.*

We can also prove that for all $L \notin \{\{2\}, \mathcal{D}\}$, L-DCC is NP-hard.

Theorem 6. *Let $L \subseteq \mathcal{D}$ be a non-empty set. If $L \neq \{2\}$ and $L \neq \mathcal{D}$, then* L-DCC *is* NP*-hard.*

Let $L \notin \{\{2\}, \mathcal{D}\}$. L-DCC is in NP and therefore NP-complete if and only if the language $\{1^\lambda \mid \lambda \in L\}$ is in NP.

4 Approximation Algorithms

The goal of this section is to devise approximation algorithms for Max-W-L-UCC and Max-W-L-DCC that work for arbitrary L. The catch is that for most L it is impossible to decide whether some cycle length is in L or not. One possibility would be to restrict ourselves to sets L such that $\{1^\lambda \mid \lambda \in L\}$ is in P. For such L, Max-W-L-UCC and Max-W-L-DCC are NP optimisation problems. Another possibility for circumventing the problem is to include the permitted cycle lengths in the input. However, it turns out that such restrictions are not necessary since we can restrict ourselves to finite sets L.

A necessary and sufficient condition for a complete graph with n vertices to have an L-cycle cover is that there exist (not necessarily distinct) lengths $\lambda_1, \ldots, \lambda_k \in L$ for some $k \in \mathbb{N}$ with $\sum_{i=1}^{k} \lambda_i = n$. We call such an n ***L*-admissible** and define $\langle L \rangle = \{n \mid n \text{ is } L\text{-admissible}\}$.

Input: an undirected graph $G = (V, U(V))$ with $|V| = n$;
an edge weight function $w : U(V) \to \mathbb{N}$
Output: an L-cycle cover C^{apx} of G if n is L-admissible, $\perp$ otherwise

1. If $n \notin \langle L \rangle$, then return $\perp$.
2. Compute a cycle cover C^{init} of maximum weight.
3. Compute a subset $P \subseteq C^{\text{init}}$ of maximum weight such that (V, P) consists of $\lceil n/5 \rceil$ paths of length two and $n - 3 \cdot \lceil n/5 \rceil$ isolated vertices.
4. Join the paths to obtain an L-cycle cover C^{apx}, return C^{apx}.

Fig. 7. A factor 2.5 approximation algorithm for Max-W-L-UCC.

Lemma 3. *For all $L \subseteq \mathbb{N}$, there exists a finite set $L' \subseteq L$ with $\langle L' \rangle = \langle L \rangle$.*

Instead of computing L'-cycle covers in the following, we assume without loss of generality that L is already a finite set.

The main idea of the two approximation algorithms is as follows: We start by computing a cycle cover C^{init} of maximum weight. Then we take a subset S of the edges of C^{init} that weighs as much as possible under the restriction that there exists an L-cycle cover that includes all edges of S. We add edges to obtain an L-cycle cover $C^{\text{apx}} \supseteq S$. Let $C^{\star}$ be an L-cycle cover of maximum weight, and assume that we can guarantee $\rho \cdot w(S) \geq w(C^{\text{init}})$ for some $\rho \geq 1$. Then $w(C^{\star}) \leq w(C^{\text{init}}) \leq \rho \cdot w(S) \leq \rho \cdot w(C^{\text{apx}})$. Thus, we have computed a factor ρ approximation to an L-cycle cover of maximum weight.

4.1 Approximating Undirected Cycle Covers

The input of our algorithm for undirected graphs is an undirected complete graph $G = (V, U(V))$ with $|V| = n$ and an edge weight function $w : U(V) \to \mathbb{N}$.

The main idea of the approximation algorithm is as follows: Every cycle cover can be decomposed into $\lceil n/5 \rceil$ vertex-disjoint paths of length two and $n - 3 \cdot \lceil n/5 \rceil$ isolated vertices. Conversely, every collection P of $\lceil n/5 \rceil$ paths of length two together with $n - 3 \cdot \lceil n/5 \rceil$ isolated vertices can be extended to form an L-cycle cover, provided that n is L-admissible.

Theorem 7. *For every fixed L, the algorithm shown in Fig. 7 is a factor* 2.5 *approximation algorithm for* Max-W-L-UCC *with running time $O(n^3)$.*

4.2 Approximating Directed Cycle Covers

Now we present an approximation algorithm for Max-W-L-DCC that achieves an approximation ratio of 3. The input consists of a directed complete graph $G = (V, D(V))$ with $|V| = n$ and an edge weight function $w : D(V) \to \mathbb{N}$.

Given a cycle cover C, we can obtain a matching $M \subseteq C$ consisting of $\lceil n/3 \rceil$ edges such that $w(M) \geq w(C)/3$. Conversely, if n is L-admissible, then every matching of cardinality $\lceil n/3 \rceil$ can be extended to form an L-cycle cover.

Input: a directed graph $G = (V, D(V))$ with $|V| = n$;
an edge weight function $w : D(V) \to \mathbb{N}$
Output: an L-cycle cover C^{apx} of G if n is L-admissible, $\perp$ otherwise

1. If $n \notin \langle L \rangle$, then return $\perp$.
2. Compute a maximum weight matching M^{init} of G of cardinality $\lceil n/3 \rceil$.
3. Join the edges in M^{init} to obtain an L-cycle cover C^{apx}, return C^{apx}.

Fig. 8. A factor 3 approximation algorithm for Max-W-L-DCC.

Instead of computing an initial cycle cover, the algorithm shown in Fig. 8 directly computes a matching of cardinality $\lceil n/3 \rceil$.

Theorem 8. *For every fixed L, the algorithm shown in Fig. 8 is a factor* 3 *approximation algorithm for* Max-W-L-UCC *with running time $O(n^3)$.*

5 Solving Max-4-UCC in Polynomial Time

The aim of this section is to show that Max-4-UCC can be solved deterministically in polynomial time. To do this, we exploit Hartvigsen's algorithm for computing a maximum-cardinality triangle-free two-matching.

A **two-matching** of an undirected graph G is a spanning subgraph in which every vertex of G has degree *at most* two. Thus, two-matchings consist of disjoint simple cycles and paths. A two-matching is a relaxation of a cycle cover (or two-factor): In a cycle cover, every vertex has degree *exactly* two. A **triangle-free two-matching** is a two-matching in which each cycle has a length of at least four. The paths can have arbitrary lengths. A triangle-free two-matching of maximum weight in graphs with edge weights zero and one can be computed deterministically in time $O(n^3)$, where n is the number of vertices [11, Chap. 3].

We want to solve Max-4-UCC, i.e. all cycles must have a length of at least four and no paths are allowed. Therefore, let M be a maximum weight triangle-free two-matching of a graph G of n vertices. If M does not contain any paths, then M is already a 4-cycle cover of maximum weight.

Let ℓ be the number of vertices of G that lie on paths in M. If $\ell \geq 4$, then we connect these paths to get a cycle of length ℓ. No weight is lost in this way, and the result is a maximum weight 4-cycle cover.

We run into trouble if $\ell \in \{1, 2, 3\}$. Let $Y = \{y_1, \ldots, y_\ell\}$ be the set of vertices that lie on paths in M. Let ℓ' be the number of edges of weight one in M that connect two vertices of Y. Then $0 \leq \ell' \leq \ell - 1$ and $w(M) = n - \ell + \ell' \leq n - 1$.

An obvious way to obtain a cycle cover from M is to break one edge of one cycle and connect the vertices of Y to this cycle. Unfortunately, breaking an edge might cause a loss of weight one. This yields the aforementioned approximation within an additive error of one. We can prove the following with a more careful analysis: Either we can avoid the loss of weight one, or indeed a maximum weight 4-cycle cover has only weight $w(M) - 1$. This yields the following result.

Theorem 9. Max-4-UCC *can be solved deterministically in time* $O(n^3)$.

6 Vertex Cover in Regular Graphs

We can prove that Min-Vertex-Cover(λ) is APX-complete for every $\lambda \geq 3$. Previously, this was only known for cubic, i.e. three-regular, graphs [2]. We need the APX-hardness of Min-Vertex-Cover(λ) for all $\lambda \geq 3$ in Sect. 3.

Theorem 10. *For every* $\lambda \in \mathbb{N}$, $\lambda \geq 3$, Min-Vertex-Cover(λ) *is* APX*-complete.*

References

1. Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, 1993.
2. Paola Alimonti and Viggo Kann. Some APX-completeness results for cubic graphs. *Theoret. Comput. Sci.*, 237(1–2):123–134, 2000.
3. Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 1999.
4. Markus Bläser. Approximationsalgorithmen für Graphüberdeckungsprobleme. Habilitationsschrift, Institut für Theoretische Informatik, Universität zu Lübeck, Lübeck, Germany, 2002.
5. Markus Bläser and Bodo Manthey. Approximating maximum weight cycle covers in directed graphs with weights zero and one. *Algorithmica*, 42(2):121–139, 2005.
6. Markus Bläser, Bodo Manthey, and Jiří Sgall. An improved approximation algorithm for the asymmetric TSP with strengthened triangle inequality. *J. Discrete Algorithms*, to appear.
7. Markus Bläser, L. Shankar Ram, and Maxim I. Sviridenko. Improved approximation algorithms for metric maximum ATSP and maximum 3-cycle cover problems. In *Proc. of the 9th Workshop on Algorithms and Data Structures (WADS)*, vol. 3608 of *Lecture Notes in Comput. Sci.*, pp. 350–359. Springer, 2005.
8. Markus Bläser and Bodo Siebert. Computing cycle covers without short cycles. In *Proc. of the 9th Ann. European Symp. on Algorithms (ESA)*, vol. 2161 of *Lecture Notes in Comput. Sci.*, pp. 368–379. Springer, 2001. Bodo Siebert is the birth name of Bodo Manthey.
9. Gérard P. Cornuéjols and William R. Pulleyblank. A matching problem with side conditions. *Discrete Math.*, 29(2):135–159, 1980.
10. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
11. David Hartvigsen. *An Extension of Matching Theory*. PhD thesis, Department of Mathematics, Carnegie Mellon University, 1984.
12. Refael Hassin and Shlomi Rubinstein. On the complexity of the k-customer vehicle routing problem. *Oper. Res. Lett.*, 33:1, 71-76 2005.
13. Pavol Hell, David G. Kirkpatrick, Jan Kratochvíl, and Igor Kríz. On restricted two-factors. *SIAM J. Discrete Math.*, 1(4):472–484, 1988.
14. Oliver Vornberger. Easy and hard cycle covers. Technical report, Universität/Gesamthochschule Paderborn, 1980.