Minimum-weight Cycle Covers and Their Approximability*

Bodo Manthey**

Yale University, Department of Computer Science
P. O. Box 208285, New Haven, CT 06520-8285, USA
manthey@cs.yale.edu

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 $L \subseteq \mathbb{N}$.
We investigate how well L-cycle covers of minimum weight can be approximated. For undirected graphs, we devise a polynomial-time approximation algorithm that achieves a constant approximation ratio for all sets L. On the other hand, we prove that the problem cannot be approximated within a factor of $2 - \varepsilon$ for certain sets L.
For directed graphs, we present a polynomial-time approximation algorithm that achieves an approximation ratio of $O(n)$, where n is the number of vertices. This is asymptotically optimal: We show that the problem cannot be approximated within a factor of $o(n)$.
To contrast the results for cycle covers of minimum weight, we show that the problem of computing L-cycle covers of maximum weight can, at least in principle, be approximated arbitrarily well.

1 Introduction

A cycle cover of a graph is a spanning subgraph that consists solely of cycles such that every vertex is part of exactly one cycle. Cycle covers are an important tool for the design of approximation algorithms for different variants of the traveling salesman problem [2, 4, 8–10, 16], for the shortest common superstring problem from computational biology [7, 24], and for vehicle routing problems [13].

In contrast to Hamiltonian cycles, which are special cases of cycle covers, cycle covers of minimum weight can be computed efficiently. This is exploited in the above mentioned algorithms, which in general start by computing a cycle cover and then join cycles to obtain a Hamiltonian cycle. Short cycles limit the approximation ratios achieved by such algorithms. Roughly speaking, the longer the cycles in the initial cover, the better the approximation ratio. Thus, we are interested in computing cycle covers without short cycles. Moreover, there

* A full version of this work is available at http://arxiv.org/abs/cs/0609103.

** Supported by the Postdoc-Program of the German Academic Exchange Service (DAAD). On leave from Saarland University, Department of Computer Science, P. O. Box 151150, 66041 Saarbrücken, Germany.

33rd Workshop on Graph-Theoret. Concepts in Comput. Sci. (WG 2007)

are algorithms that perform particularly well if the cycle covers computed do not contain cycles of odd length [4]. Finally, some vehicle routing problems [13] require covering vertices with cycles of bounded length. Therefore, we consider *restricted cycle covers*, where cycles of certain lengths are ruled out a priori: For a set $L \subseteq \mathbb{N}$, an L*-cycle cover* is a cycle cover in which the length of each cycle is in L. Unfortunately, computing L-cycle covers is hard for almost all sets L [15, 20]. Thus, in order to fathom the possibility of designing approximation algorithms based on computing cycle covers, our aim is to find out how well L-cycle covers can be approximated.

Beyond being a basic tool for approximation algorithms, cycle covers are interesting in their own right. Matching theory and graph factorization are important topics in graph theory. The classical matching problem is the problem of finding one-factors, i. e., spanning subgraphs in which every vertex is incident to exactly one edge. Cycle covers of undirected graphs are also called two-factors since every vertex is incident to exactly two edges in a cycle cover. Both structural properties of graph factors and the complexity of finding graph factors have been the topic of a considerable amount of research (cf. Lovász and Plummer [17] and Schrijver [23]).

1.1 Preliminaries

Let $G = (V, E)$ be a graph. If G is undirected, then a **cycle cover** of G is a subset $C \subseteq E$ of the edges of G such that all vertices in V are incident to exactly two edges in C. If G is a directed graph, then a cycle cover of G is a subset $C \subseteq E$ such that all vertices are incident to exactly one incoming and one outgoing edge in C. Thus, the graph (V, C) consists solely of vertex-disjoint cycles. 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 are of length three and two, respectively. We call a cycle of length λ a **$\boldsymbol{\lambda}$-cycle** for short.

An **$\boldsymbol{L}$-cycle cover** of an undirected graph is a cycle cover in which the length of every cycle is in the set $L \subseteq \mathcal{U} = \{3, 4, 5, \ldots\}$. An L-cycle cover of a directed graph is analogously defined except that $L \subseteq \mathcal{D} = \{2, 3, 4, \ldots\}$. A special case of L-cycle covers are **$\boldsymbol{k}$-cycle covers**, which are $\{k, k + 1, \ldots\}$-cycle covers. Let $\overline{L} = \mathcal{U} \setminus L$ in the case of undirected graphs, and let $\overline{L} = \mathcal{D} \setminus L$ in the case of directed graphs (whether we consider undirected or directed cycle covers will be clear from the context).

Given edge weights $w : E \to \mathbb{N}$, the **weight $\boldsymbol{w(C)}$** of a subset $C \subseteq E$ of the edges of G is $w(C) = \sum_{e \in C} w(e)$. In particular, this defines the weight of a cycle cover since we view cycle covers as sets of edges.

Min-$\boldsymbol{L}$-UCC is the following optimization problem: Given an undirected complete graph with non-negative edge weights that satisfy the triangle inequality ($w(\{u, v\}) \leq w(\{u, x\}) + w(\{x, v\})$ for all $u, x, v \in V$) find an L-cycle cover of minimum weight. **Min-$\boldsymbol{k}$-UCC** is defined for $k \in \mathcal{U}$ like Min-L-UCC except that k-cycle covers rather than L-cycle covers are sought. The triangle inequality is not only a natural restriction, it is also necessary: If finding L-cycle covers in graphs is NP-hard, then Min-L-UCC without the triangle inequality does not allow for any approximation at all.

Min-L-DCC and **Min-k-DCC** are defined for directed graphs like Min-L-UCC and Min-k-UCC for undirected graphs except that $L \subseteq \mathcal{D}$ and $k \in \mathcal{D}$ and the triangle inequality is of the form $w(u, v) \leq w(u, x) + w(x, v)$.

Finally, **Max-L-UCC**, **Max-k-UCC**, **Max-L-DCC**, and **Max-k-DCC** are analogously defined except that cycle covers of maximum weight are sought and that the edge weights do not have to fulfill the triangle inequality.

1.2 Previous Results

Min-$\mathcal{U}$-UCC, i. e., the undirected cycle cover problem without any restrictions, can be solved in polynomial time via Tutte's reduction to the classical perfect matching problem [17]. By a modification of an algorithm of Hartvigsen [12], also 4-cycle covers of minimum weight in graphs with edge weights one and two can be computed efficiently. For Min-k-UCC restricted to graphs with edge weights one and two, there exists a factor 7/6 approximation algorithm for all k [6]. Hassin and Rubinstein [14] presented a randomized approximation algorithm for Max-{3}-UCC that achieves an approximation ratio of $83/43 + \epsilon$. Max-L-UCC admits a factor 2 approximation algorithm for arbitrary sets L [18, 20]. Goemans and Williamson [11] showed that Min-k-UCC and Min-$\{k\}$-UCC can be approximated within a factor of 4. Min-L-UCC is NP-hard and APX-hard if $\overline{L} \not\subseteq \{3\}$, i. e., for all but a finite number of sets L [15, 19, 20, 25].

Min-$\mathcal{D}$-DCC, which is also known as the *assignment problem*, can be solved in polynomial time by a reduction to the minimum weight perfect matching problem in bipartite graphs [1]. The only other L for which Min-L-DCC can be solved in polynomial time is $L = \{2\}$. For all $L \subseteq \mathcal{D}$ with $L \neq \{2\}$ and $L \neq \mathcal{D}$, Min-L-DCC and Max-L-DCC are APX-hard and NP-hard [19, 20].

There is a 4/3 approximation for Max-3-DCC [5] as well as for Min-k-DCC for $k \geq 3$ with the restriction that the only edge weights allowed are one and two [3]. Max-L-DCC can be approximated within a factor of 8/3 for all L [20].

If Min-L-UCC or Min-L-DCC is NP-hard, then the triangle inequality is necessary for efficient approximations of this problem; without the triangle inequality, the problems cannot be approximated at all.

1.3 New Results

While L-cycle covers of *maximum* weight allow for constant factor approximations, only little is known so far about the approximability of computing L-cycle covers of *minimum* weight. Our aim is to close this gap.

We present an approximation algorithm for Min-L-UCC that works for all sets $L \subseteq \mathcal{U}$ and achieves a constant approximation ratio (Section 2.1). Its running-time is $O(n^2 \log n)$. On the other hand, we show that the problem cannot be approximated within a factor of $2 - \varepsilon$ for general L (Section 2.2).

Our approximation algorithm for Min-L-DCC achieves a ratio of $O(n)$, where n is the number of vertices (Section 3.1). This is asymptotically optimal: There exist sets L for which no algorithm can approximate Min-L-DCC within a factor of $o(n)$ (Section 3.2). Furthermore, we argue that Min-L-DCC is harder to approximate than the other three variants even for more "natural" sets L than the sets used to show the inapproximability (Section 3.3).

Finally, to contrast our results for Min-L-UCC and Min-L-DCC, we show that Max-L-UCC and Max-L-DCC can be approximated arbitrarily well at least in principle (Section 4).

2 Approximability of Min-L-UCC

2.1 An Approximation Algorithm for Min-L-UCC

The aim of this section is to devise an approximation algorithm for Min-L-UCC that works for all sets $L \subseteq \mathcal{U}$. The catch is that for most L it is impossible to decide whether some cycle length is in L since there are uncountably many sets L: If, for instance, L is not a recursive set, then deciding whether a cycle cover is an L-cycle cover is impossible. One option would be to restrict ourselves to sets L such that the unary language $\{1^\lambda \mid \lambda \in L\}$ is in P. For such L, Min-L-UCC and Min-L-DCC are NP optimization problems. Another possibility for circumventing the problem would be to include the permitted cycle lengths in the input. While such restrictions are mandatory if we want to compute optimum solutions, they are not needed for our approximation algorithms.

A complete n-vertex graph contains an L-cycle cover as a spanning subgraph if and only if 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 $\langle L \rangle = \{n \mid n \text{ is } L\text{-admissible}\}$. Although L can be arbitrarily complicated, $\langle L \rangle$ always allows efficient membership testing: For all $L \subseteq \mathbb{N}$, there exists a finite set $L' \subseteq L$ with $\langle L' \rangle = \langle L \rangle$ [20].

Let g_L be the greatest common divisor of all numbers in L. Then $\langle L \rangle$ is a subset of the set of natural numbers divisible by g_L. There exists a minimum $p_L \in \mathbb{N}$ such that $\eta g_L \in \langle L \rangle$ for all $\eta > p_L$. The number p_L is the Frobenius number [22] of $\{\lambda \mid g_L \lambda \in L\}$, which is L scaled down by g_L.

In the following, it suffices to know such a finite set $L' \subseteq L$. The L-cycle covers computed by our algorithm will in fact be L'-cycle covers. In order to estimate the approximation ratio, this cycle cover will be compared to an optimal $\langle L' \rangle$-cycle cover. Since $L' \subseteq L \subseteq \langle L' \rangle$, every L'- or L-cycle cover is also a $\langle L' \rangle$-cycle cover. Thus, the weight of an optimal $\langle L' \rangle$-cycle cover provides a lower bound for the weight of both an optimal L'- and an optimal L-cycle cover. For simplicity, we do not mention L' in the following. Instead, we assume that already L is a finite set, and we compare the weight of the L-cycle cover computed to the weight of an optimal $\langle L \rangle$-cycle cover to bound the approximation ratio.

Goemans and Williamson have presented a technique for approximating constrained forest problems [11], which we will exploit. Let $G = (V, E)$ be an undirected graph, and let $w : E \to \mathbb{N}$ be non-negative edge weights. Let 2^V denote the power set of V. A function $f : 2^V \to \{0, 1\}$ is called a **proper function** if it satisfies

– $f(S) = f(V \setminus S)$ for all $S \subseteq V$ (symmetry),
– if A and B are disjoint, then $f(A) = f(B) = 0$ implies $f(A \cup B) = 0$ (disjointness), and
– $f(V) = 0$.

The aim is to find a set F of edges such that there is at least one edge connecting S to $V \setminus S$ for all $S \subseteq V$ with $f(S) = 1$. (The name "constrained forest problems" comes from the fact that it suffices to consider forests as solutions; cycles only increase the weight of a solution.) Goemans and Williamson have presented an approximation algorithm [11, Fig. 1] for constrained forest problems that are characterized by proper functions. We will refer to their algorithm as GoeWill.

Theorem 1 (Goemans and Williamson [11]). *Let ℓ be the number of vertices v with $f(\{v\}) = 1$. Then* GoeWill *is a $(2 - \frac{2}{\ell})$-approximation for the constrained forest problem defined by a proper function f.*

In particular, the function f_L given by

$$f_L(S) = \begin{cases} 1 \text{ if } |S| \not\equiv 0 \pmod{g_L} \text{ and} \\ 0 \text{ if } |S| \equiv 0 \pmod{g_L} \end{cases}$$

is proper if $|V| = n$ is divisible by g_L. (If n is not divisible by g_L, then G does not contain an L-cycle cover at all.) Given this function, a solution is a forest $H = (V, F)$ such that the size of every connected component of H is a multiple of g_L. In particular, if $g_L = 1$, then $f_L(S) = 0$ for all S, and an optimum solution are n isolated vertices.

If the size of all components of the solution obtained are in $\langle L \rangle$, we are done: By duplicating all edges, we obtain Eulerian components. Then we construct an $\langle L \rangle$-cycle cover by traversing the Eulerian components and taking shortcuts whenever we come to a vertex that we have already visited. Finally, we divide each λ-cycle into paths of lengths $\lambda_1 - 1, \ldots, \lambda_k - 1$ for some k such that $\lambda_1 + \ldots + \lambda_k = \lambda$ and $\lambda_i \in L$ for all i. By connecting the respective endpoints of each path, we obtain cycles of lengths $\lambda_1, \ldots, \lambda_k$. We perform this for all components to get an L-cycle cover. A careful analysis shows that the ratio achieved is 4. The details for the special case of $L = \{k\}$ are spelled out by Goemans and Williamson [11].

However, this procedure does not work for general sets L since the sizes of some components may not be in $\langle L \rangle$. This can happen if $p_L > 0$ (for $L = \{k\}$, for which the algorithm works, we have $p_L = 0$).

In the following, our aim is to add edges to the forest $H = (V, E)$ output by GoeWill such that the size of each component is in $\langle L \rangle$. This will lead to an approximation algorithm for Min-L-UCC with a ratio of $4 \cdot (p_L + 4)$, which is constant for each L. Let F^* denote the set of edges of a minimum-weight forest such that the size of each component is in $\langle L \rangle$. The set F^* is a solution to G, w, and f_L, but not necessarily an optimum solution.

By Theorem 1, we have $w(F) \leq 2 \cdot w(F^*)$ since $w(F^*)$ is at least the weight of an optimum solution to G, w, and f_L. Let $C = (V', F')$ be any connected component of F with $|V'| \notin \langle L \rangle$. The optimum solution F^* must contain an edge that connects V' to $V \setminus V'$. The weight of this edge is at least the weight of the minimum-weight edge connecting V' to $V \setminus V'$.

We will add edges until the sizes of all components is in $\langle L \rangle$. Our algorithm acts in phases as follows: Let $H = (V, F)$ be the graph at the beginning of the current phase, and let $C_1, \ldots, C_a$ be its connected components, where V_i is the vertex set of C_i. We will construct a new graph $\tilde{H} = (V, \tilde{F})$ with $\tilde{F} \supseteq F$. Let $C_1, \ldots, C_b$ be the connected components with $|V_i| \notin \langle L \rangle$. We call these components **illegal**. For $i \in \{1, \ldots, b\}$, let e_i be the cheapest edge connecting V_i to $V \setminus V_i$. (Note that $e_i = e_j$ for $i \neq j$ is allowed.) We add all these edges to F to obtain $\tilde{F} = F \cup \{e_1, \ldots, e_b\}$. Since e_i is the cheapest edge connecting V_i to $V \setminus V_i$, the graph $\tilde{H} = (V, \tilde{F})$ is a forest. (If some e_i are not uniquely determined, cycles may occur. We can avoid these cycles by discarding some of the e_i to break the cycles. For the sake of simplicity, we ignore this case in the following analysis.) If $\tilde{H}$ still contains illegal components, we set H to be $\tilde{H}$ and iterate the procedure.

Now we have $w(\tilde{F}) \leq w(F) + 2 \cdot w(F^*)$, i. e., in the overall weight increases by at most $2 \cdot w(F^*)$ in every phase. Furthermore, after at most $\lfloor p_L/2 \rfloor + 1$ phases, $\tilde{H}$ does not contain any illegal components.

Eventually, we obtain a forest that consists solely of components whose sizes are in $\langle L \rangle$. We call this forest $\tilde{H} = (V, \tilde{F})$. Then we proceed as already described above: We duplicate each edge, thus obtaining Eulerian components. After that, we take shortcuts to obtain an $\langle L \rangle$-cycle cover. Finally, we break edges and connect the endpoints of each path to obtain an L-cycle cover. The weight of this L-cycle cover is at most $4 \cdot w(\tilde{F})$.

Overall, we obtain ApxUndir (Algorithm 1) and the following theorem.

Theorem 2. *For every $L \subseteq \mathcal{U}$,* ApxUndir *is a factor $(4 \cdot (p_L + 4))$ approximation algorithm for* Min-L-UCC. *Its running-time is $O(n^2 \log n)$.*

Proof. Let C^* be a minimum-weight $\langle L \rangle$-cycle cover. The weight of $\tilde{F}$ is bounded from above by $w(\tilde{F}) \leq \left(\left\lfloor \frac{p_L}{2} \right\rfloor + 1\right) \cdot 2 \cdot w(F^*) + 2 \cdot w(F^*) \leq \left(p_L + 4\right) \cdot w(C^*)$. Combining this with $w(C^{\mathrm{apx}}) \leq 4 \cdot w(\tilde{F})$ yields the approximation ratio.

Executing GoeWill takes time $O(n^2 \log n)$. All other operations can be implemented to run in time $O(n^2)$. □

We conclude the analysis of this algorithm by mentioning that the approximation ratio of the algorithm depends indeed linearly on p_L.

2.2 Unconditional Inapproximability of Min-*L*-UCC

In this section, we provide a lower bound for the approximability of Min-L-UCC as a counterpart to the approximation algorithm of the previous section. We show that the problem cannot be approximated within a factor of $2 - \varepsilon$. This inapproximability result is unconditional, i. e., it does not rely on complexity theoretic assumptions like $\mathsf{P} \neq \mathsf{NP}$.

Algorithm 1 ApxUndir.

Input: undirected complete graph $G = (V, E)$, $|V| = n$; edge weights $w : E \to \mathbb{N}$ satisfying the triangle inequality

Output: an L-cycle cover C^{apx} of G if n is L-admissible, $\perp$ otherwise

if $n \notin \langle L \rangle$ **then**
  return $\perp$
run GoeWill using the function f_L described in the text to obtain $H = (V, F)$
while the size of some connected components of H is not in $\langle L \rangle$ **do**
  let $C_1, \ldots, C_a$ be the connected components of H, where V_i is the vertex set of C_i; let $C_1, \ldots, C_b$ be its illegal components
  let e_i be the lightest edge connecting V_i to $V \setminus V_i$
  add $e_1, \ldots, e_b$ to F
  while H contains cycles **do**
    remove one e_i to break a cycle
duplicate each edge to obtain a multi-graph consisting of Eulerian components
for all components of the multi-graph **do**
  walk along an Eulerian cycle
  take shortcuts to obtain a Hamiltonian cycle
  discard edges to obtain a collection of paths, the number of vertices of each of which is in L
  connect the two endpoints of every path in order to obtain cycles
the union of all cycles constructed forms C^{apx}; return C^{apx}

The key to the inapproximability of Min-L-UCC are **immune sets** [21]: An infinite set $L \subseteq \mathbb{N}$ is called an immune set if L does not contain an infinite recursively enumerable subset. Our result limits the possibility of designing general approximation algorithms for L-cycle covers. To obtain algorithms with a ratio better than 2, we have to design algorithms tailored to specific sets L.

Finite variations of immune sets are again immune sets. Thus for every $k \in \mathbb{N}$, there exist immune sets L containing no number smaller than k.

Theorem 3. *Let $\varepsilon > 0$ be arbitrarily small. Let $k > 2/\varepsilon$, and let $L \subseteq \{k, k+1, \ldots\}$ be an immune set. Then* Min-L-UCC *cannot be approximated within a factor of* $2 - \varepsilon$.

Theorem 3 is tight since L-cycle covers can be approximated within a factor of 2 by L'-cycle covers for every set $L' \subseteq L$ with $\langle L' \rangle = \langle L \rangle$. For finite sets L', all L'-cycle cover problems are NP optimization problems. This means that in principle optimum solutions can be found, although this may take exponential time. The following Theorem 4 holds in particular for finite sets L'. In order to actually get an approximation algorithm for Min-L-UCC out of it, we have to solve Min-L'-UCC finite L', which is NP-hard and APX-hard. But the proof of Theorem 4 shows also that any approximation algorithm for Min-L'-UCC for finite sets L' that achieves an approximation ratio of r can be turned into an approximation algorithm for the general problem with a ratio of $2r$. Let $\min_L(G, w)$ denote the weight of a minimum-weight L-cycle cover of G with edge weights w.

Algorithm 2 ApxDir.

Input: directed complete graph $G = (V, E)$, $|V| = n$; edge weights $w : E \to \mathbb{N}$ satisfying the triangle inequality

Output: an L-cycle cover C^{apx} of G if n is L-admissible, $\perp$ otherwise

```
if n ∉ ⟨L⟩ then
    return ⊥
construct an undirected complete graph G_U = (V, E_U) with edge weights
    w_U({u, v}) = w(u, v) + w(v, u)
run ApxUndir on G_U and w_U to obtain C_U^apx
for all cycles c_U of C_U^apx do
    c_U corresponds to a cycle of G that can be oriented in two ways; put the
        orientation c that yields less weight into C^apx
return C^apx
```

Theorem 4. *Let $L \subseteq \mathcal{U}$ be a non-empty set, and let $L' \subseteq L$ with $\langle L' \rangle = \langle L \rangle$. Then $\min_{L'}(G, w) \leq 2 \cdot \min_L(G, w)$ for all undirected graphs G with edge weights w satisfying the triangle inequality.*

3 Approximability of Min-*L*-DCC

3.1 An Approximation Algorithm for Min-*L*-DCC

In this section, we present an approximation algorithm for Min-L-DCC. The algorithm exploits ApxUndir to achieve an approximation ratio of $O(n)$. The hidden factor depends on p_L again. This result matches asymptotically the lower bound of Section 3.2 and shows that Min-L-DCC can be approximated at least to some extent. (For instance, without the triangle inequality, no polynomial-time algorithm achieves a ratio of $O(\exp(n))$ for an NP-hard L-cycle cover problem unless $\mathsf{P} = \mathsf{NP}$.)

In order to approximate Min-L-DCC, we reduce the problem to a variant of Min-L-UCC, where also 2-cycles are allowed: We obtain a 2-cycle of an undirected graph by taking an edge $\{u, v\}$ twice. Let $G = (V, E)$ be a directed complete graph with n vertices and edge weights $w : E \to \mathbb{N}$ that fulfill the triangle inequality. The corresponding undirected complete graph $G_U = (V, E_U)$ has weights $w_U : E_U \to \mathbb{N}$ with $w_U(\{u, v\}) = w(u, v) + w(v, u)$.

Let C be any cycle cover of G. The corresponding cycle cover C_U of G_U is given by $C_U = \{\{u, v\} \mid (u, v) \in C\}$. Note that we consider C_U as a multiset: If both (u, v) and (v, u) are in C, i. e., u and v form a 2-cycle, then $\{u, v\}$ occurs twice in C_U. We can bound the weight of C_U in terms of the weight of C: For every cycle cover C of G, we have $w_U(C_U) \leq n \cdot w(C)$.

Our algorithm computes an L'-cycle cover for some finite $L' \subseteq L$ with $\langle L' \rangle = \langle L \rangle$. As in Section 2.1, the weight of the cycle cover computed is compared to an optimum $\langle L \rangle$-cycle cover rather than an optimum L-cycle cover. Thus, we can again assume that already L is a finite set.

The algorithm ApxUndir, which was designed for undirected graphs, remains to be an $O(1)$ approximation if we allow $2 \in L$. The numbers p_L and g_L are defined in the same way as in Section 2.1.

Let C_U^{apx} be the L-cycle cover output by ApxUndir on G_U. We transfer C_U^{apx} into an L-cycle cover C^{apx} of G. For every cycle c_U of C_U^{apx}, we can orient the corresponding directed cycle c in two directions. We take the orientation that yields less weight, thus $w(C^{\text{apx}}) \leq w_U(C_U^{\text{apx}})/2$. Overall, we obtain ApxDir (Algorithm 2), which achieves an approximation ratio of $O(n)$ for every L.

Theorem 5. *For every $L \subseteq \mathcal{D}$,* ApxDir *is a factor $(2n \cdot (p_L + 4))$ approximation algorithm for* Min-L-DCC*. Its running-time is $O(n^2 \log n)$.*

Proof. Theorem 2 yields $w_U(C_U^{\text{apx}}) \leq 4 \cdot (p_L + 4) \cdot w_U(C_U^*)$, where C_U^* is an optimal $\langle L \rangle$-cycle cover of G_U. Now consider an optimum $\langle L \rangle$-cycle cover C^* of G. Overall, we obtain $w(C^{\text{apx}}) \leq \frac{1}{2} \cdot w_U(C_U^{\text{apx}}) \leq 2 \cdot (p_L + 4) \cdot w_U(C_U^*) \leq 2n \cdot (p_L + 4) \cdot w(C^*)$.

The running-time is dominated by the time needed to execute GoeWill in ApxUndir, which is $O(n^2 \log n)$. □

3.2 Unconditional Inapproximability of Min-L-DCC

For undirected graphs, both Max-L-UCC and Min-L-UCC can be approximated efficiently to within constant factors. Surprisingly, in case of directed graphs, this holds only for the maximization variant of the directed L-cycle cover problem. Min-L-DCC cannot be approximated within a factor of $o(n)$ for certain sets L, where n is the number of vertices of the input graph. In particular, ApxDir achieves asymptotically optimal approximation ratios for Min-L-DCC.

Similar to the case of Min-L-UCC, this result shows that to find approximation algorithms, specific properties of the sets L have to be exploited. Moreover, as we will discuss in Section 3.3, Min-L-DCC seems to be much harder a problem than the other three variants, even for more practical sets L.

Theorem 6. *Let $L \subseteq \mathcal{U}$ be an immune set. Then no approximation algorithm for* Min-L-DCC *achieves an approximation ratio of $o(n)$, where n is the number of vertices of the input graph.*

Min-L'-DCC for a finite set L' is an NP optimization problem. Thus, it can be solved, although this may take exponential time. Therefore, the following result shows that Min-L-DCC can be approximated for all L within a ratio of n/s for arbitrarily large constants s, although this may also take exponential time. In this sense, Theorem 6 is tight.

Theorem 7. *For every L and every $s > 1$, there exists a finite set $L' \subseteq L$ with $\langle L' \rangle = \langle L \rangle$ such that $\min_{L'}(G, w) \leq \frac{n}{s} \cdot \min_L(G, w)$ for all directed graphs G with edge weights w satisfying the triangle inequality.*

3.3 Remarks on the Approximability of Min-L-DCC

It might seem surprising that Min-L-DCC is much harder to approximate than Min-L-UCC or the maximization problems Max-L-UCC and Max-L-DCC. In the following, we give some reasons why Min-L-DCC is more difficult than the other three L-cycle cover problems. In particular, even for "easy" sets L, for which membership testing can be done in polynomial time, it seems that Min-L-DCC is much harder to approximate than the other three variants.

Why is minimization harder than maximization? To get a good approximation ratio in the case of maximization problems, it suffices to detect a few "good", i. e., heavy edges. If we have a decent fraction of the heaviest edges, their total weight is already within a constant factor of the weight of an optimal L-cycle cover. In order to form an L-cycle cover, we have to connect the heavy edges using other edges. These other edges might be of little weight, but they do not decrease the weight that we have already obtained from the heavy edges. For approximating cycle covers of minimum weight, it does not suffice to detect a couple of "good", i. e., light edges: Once we have selected a couple of good edges, we might have to connect them with heavy-weight edges. These heavy-weight edges can worsen the approximation ratio dramatically.

Why is Min-L-DCC harder than Min-L-UCC? If we have a cycle in an undirected graph whose length is in $\langle L \rangle$ but not in L (or not in L' but we do not know if it is in L), then we can decompose it into smaller cycles all lengths of which are in L. This can be done such that the weight at most doubles. However, by decomposing a long cycle of a directed graph into smaller ones, the weight can increase tremendously.

Finally, a question that arises naturally is if we can do better if all allowed cycle lengths are known a priori. This can be achieved by restricting ourselves to sets L that allow efficient membership testing. Another option is to include the allowed cycle lengths in the input, i. e., in addition to an n-vertex graph and edge weights, we are given a subset of $\{2, 3, \ldots, n\}$ of allowed cycle lengths. A constant factor approximation for either variant would, however, yield an approximation algorithm for the asymmetric traveling salesman problem (ATSP) with dramatically improved approximation ratio.

4 Properties of Maximum-weight Cycle Covers

To contrast our results for Min-L-UCC and Min-L-DCC, we show that their maximization counterparts Max-L-UCC and Max-L-DCC can, at least in principle, be approximated arbitrarily well; their inapproximability is solely due to their APX-hardness and not to the difficulties arising from undecidable sets L.

Let $\max_L(G, w)$ be the weight of a maximum-weight L-cycle cover of G with edge weights w. The edge weights w do not have to fulfill the triangle inequality. We will show that $\max_L(G, w)$ can be approximated arbitrarily well by $\max_{L'}(G, w)$ for finite sets $L' \subseteq L$ with $\langle L' \rangle = \langle L \rangle$. Thus, any approximation algorithm for Max-L'-UCC or Max-L'-DCC for finite sets L' immediately yields an approximation algorithm for general sets L with an only negligibly worse approximation ratio.

The following theorem for directed cycle covers contains the case of undirected graphs as a special case.

Theorem 8. *Let $L \subseteq \mathcal{D}$ be any non-empty set, and let $\varepsilon > 0$. Then there exists a finite subset $L' \subseteq L$ with $\langle L' \rangle = \langle L \rangle$ such that $\max_{L'}(G, w) \geq (1-\varepsilon) \cdot \max_L(G, w)$ for all graphs G with edge weights w.*

5 Concluding Remarks

First of all, we would like to know if there is a general upper bound for the approximability of Min-L-UCC: Does there exist an r (independent of L) such that Min-L-UCC can be approximated within a factor of r? We conjecture that such an algorithm exists. If such an algorithm works also for the slightly more general problem Min-L-UCC with $2 \in L$ (see Section 3.1), then we would obtain a factor $rn/2$ approximation for Min-L-DCC as well.

While the problem of computing L-cycle cover of minimum weight can be approximated efficiently in the case of undirected graphs, the directed variant seems to be much harder. We are interested in developing approximation algorithms for Min-L-DCC for particular sets L or for certain classes of sets L. For instance, how well can Min-L-DCC be approximated if L is a finite set? Are there non-constant lower bounds for the approximability of Min-L-DCC, for instance bounds depending on $\max(L)$?

References

1. Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, 1993.
2. Markus Bläser. A 3/4-approximation algorithm for maximum ATSP with weights zero and one. In Klaus Jansen, Sanjeev Khanna, José D. P. Rolim, and Dana Ron, editors, *Proc. of the 7th Int. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, volume 3122 of *Lecture Notes in Computer Science*, pages 61–71. Springer, 2004.
3. Markus Bläser and Bodo Manthey. Approximating maximum weight cycle covers in directed graphs with weights zero and one. *Algorithmica*, 42(2):121–139, 2005.
4. Markus Bläser, Bodo Manthey, and Jiří Sgall. An improved approximation algorithm for the asymmetric TSP with strengthened triangle inequality. *Journal of Discrete Algorithms*, 4(4):623–632, 2006.
5. Markus Bläser, L. Shankar Ram, and Maxim I. Sviridenko. Improved approximation algorithms for metric maximum ATSP and maximum 3-cycle cover problems. In Frank Dehne, Alejandro López-Ortiz, and Jörg-Rüdiger Sack, editors, *Proc. of the 9th Workshop on Algorithms and Data Structures (WADS)*, volume 3608 of *Lecture Notes in Computer Science*, pages 350–359. Springer, 2005.
6. Markus Bläser and Bodo Siebert. Computing cycle covers without short cycles. In Friedhelm Meyer auf der Heide, editor, *Proc. of the 9th Ann. European Symp. on Algorithms (ESA)*, volume 2161 of *Lecture Notes in Computer Science*, pages 368–379. Springer, 2001. Bodo Siebert is the birth name of Bodo Manthey.

7. Avrim L. Blum, Tao Jiang, Ming Li, John Tromp, and Mihalis Yannakakis. Linear approximation of shortest superstrings. *Journal of the ACM*, 41(4):630–647, 1994.
8. Hans-Joachim Böckenhauer, Juraj Hromkovič, Ralf Klasing, Sebastian Seibert, and Walter Unger. Approximation algorithms for the TSP with sharpened triangle inequality. *Information Processing Letters*, 75(3):133–138, 2000.
9. L. Sunil Chandran and L. Shankar Ram. On the relationship between ATSP and the cycle cover problem. *Theoretical Computer Science*, 370(1-3):218–228, 2007.
10. Zhi-Zhong Chen, Yuusuke Okamoto, and Lusheng Wang. Improved deterministic approximation algorithms for Max TSP. *Information Processing Letters*, 95(2):333–342, 2005.
11. Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.
12. David Hartvigsen. *An Extension of Matching Theory.* PhD thesis, Department of Mathematics, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, September 1984.
13. Refael Hassin and Shlomi Rubinstein. On the complexity of the k-customer vehicle routing problem. *Operations Research Letters*, 33(1):71–76, 2005.
14. Refael Hassin and Shlomi Rubinstein. Erratum to "An approximation algorithm for maximum triangle packing" [Discrete Applied Mathematics 154 (2006) 971–979]. *Discrete Applied Mathematics*, 154(18):2620, 2006.
15. Pavol Hell, David G. Kirkpatrick, Jan Kratochvíl, and Igor Kríz. On restricted two-factors. *SIAM Journal on Discrete Mathematics*, 1(4):472–484, 1988.
16. Haim Kaplan, Moshe Lewenstein, Nira Shafrir, and Maxim I. Sviridenko. Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. *Journal of the ACM*, 52(4):602–626, 2005.
17. László Lovász and Michael D. Plummer. *Matching Theory*, volume 121 of *North-Holland Mathematics Studies.* Elsevier, 1986.
18. Bodo Manthey. Approximation algorithms for restricted cycle covers based on cycle decompositions. In Fedor V. Fomin, editor, *Proc. of the 32nd Int. Workshop on Graph-Theoretical Concepts in Computer Science (WG)*, volume 4271 of *Lecture Notes in Computer Science*, pages 336–347, 2006.
19. Bodo Manthey. On approximating restricted cycle covers. In Thomas Erlebach and Giuseppe Persiano, editors, *Proc. of the 3rd Workshop on Approximation and Online Algorithms (WAOA 2005)*, volume 3879 of *Lecture Notes in Computer Science*, pages 282–295. Springer, 2006.
20. Bodo Manthey. On approximating restricted cycle covers. Report 07-011, Electronic Colloquium on Computational Complexity (ECCC), 2007.
21. Piergiorgio Odifreddi. *Classical Recursion Theory*, volume 125 of *Studies in Logic and The Foundations of Mathematics.* Elsevier, 1989.
22. Jorge L. Ramírez Alfonsín. *The Diophantine Frobenius Problem*, volume 30 of *Oxford Lecture Series in Mathematics and Its Applications.* Oxford University Press, 2005.
23. Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics.* Springer, 2003.
24. Z. Sweedyk. A $2\frac{1}{2}$-approximation algorithm for shortest superstring. *SIAM Journal on Computing*, 29(3):954–986, 1999.
25. Oliver Vornberger. Easy and hard cycle covers. Technical report, Universität/Gesamthochschule Paderborn, 1980.