Approximation Algorithms for Restricted Cycle Covers Based on Cycle Decompositions*

Bodo Manthey**

Universität des Saarlandes, Informatik
Postfach 151150, 66041 Saarbrücken, Germany
manthey@cs.uni-sb.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 $L \subseteq \mathbb{N}$. For most sets L, the problem of computing L-cycle covers of maximum weight is NP-hard and APX-hard.
We devise polynomial-time approximation algorithms for L-cycle covers. More precisely, we present a factor 2 approximation algorithm for computing L-cycle covers of maximum weight in undirected graphs and a factor 20/7 approximation algorithm for the same problem in directed graphs. Both algorithms work for arbitrary sets L. To do this, we develop a general decomposition technique for cycle covers.
Finally, we show tight lower bounds for the approximation ratios achievable by algorithms based on such decomposition techniques.

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 play an important role in the design of approximation algorithms for several variants of the travelling salesman problem [3, 5, 6, 9–12, 17], for the shortest common superstring problem [8, 21], and for vehicle routing problems [14].

We consider cycle covers in (directed or undirected) edge-weighted complete graphs. Given such a graph, the aim is to find a cycle cover of maximum weight. In contrast to Hamiltonian cycles, which are special cases of cycle covers, cycle covers of maximum weight can be computed efficiently. This is exploited in the aforementioned approximation algorithms, which 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, the better the approximation ratio. Thus, we are interested in computing cycle covers that do not contain short cycles. Moreover, there are approximation algorithms that

* A full version of this work is available at http://arxiv.org/abs/cs/0604020.
** Work done in part at the Institut für Theoretische Informatik of the Universität zu Lübeck and supported by DFG research grant RE 672/3.

32nd Workshop on Graph-Theoret. Concepts in Comput. Sci. (WG 2006)

perform particularly well if the cycle covers computed do not contain cycles of odd length [5]. Finally, some vehicle routing problems [14] 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 $L \subseteq \mathbb{N}$, an *L-cycle cover* is a cycle cover in which the length of each cycle is in L.

Unfortunately, computing L-cycle covers of maximum weight is hard in general [16, 19]. 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 factorisation are important topics in graph theory. Cycle covers of undirected graphs are also known as *two-factors* since every vertex is incident to exactly two edges in a cycle cover. A considerable amount of research has been done on structural properties of graph factors and on the complexity of finding graph factors (cf. Lovász and Plummer [18] and Schrijver [20]). In particular, the complexity of finding restricted two-factors, i.e. L-cycle covers in undirected graphs, has been investigated, and Hell et al. [16] and Manthey [19] showed that finding L-cycle covers in graphs is NP-hard for almost all L.

1.1 Preliminaries

Let $G = (V, E)$ be a graph with vertex set V and edge set E. 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 will refer to a cycle of length ℓ as an ℓ-cycle for short. Furthermore, cycles of odd or even length will simply be called odd or even cycles, respectively.

An *L-cycle cover* of an undirected graph is a cycle cover in which the length of every cycle is in $L \subseteq \mathcal{U} = \{3, 4, 5, \ldots\}$. An L-cycle cover of a directed graph is analogously defined except that $L \subseteq \mathcal{D} = \{2, 3, 4, \ldots\}$. A *$k$-cycle cover* is a $\{k, k+1, \ldots\}$-cycle cover. In the following, 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 a weight function $w : E \to \mathbb{N}$, the *weight* $w(C)$ of a subset $C \subseteq E$ of the edges of G is $w(C) = \sum_{e \in C} w(e)$. In particular, this defines the weight of a cycle cover since we view cycle covers as sets of edges.

Max-L-UCC is the following optimisation problem: Given an undirected complete graph with non-negative edge weights, find an L-cycle cover of maximum weight. Max-k-UCC is defined for $k \in \mathcal{U}$ like Max-L-UCC except that k-cycle covers are sought instead of L-cycle covers.

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

A *single* is a single edge (or a path of length one) in a graph, while a *double* is a path of length two.

1.2 Previous Results

Undirected Cycle Covers. Max-$\mathcal{U}$-UCC, i.e. the undirected cycle cover problem without any restrictions, can be solved in polynomial time via Tutte's reduction (cf. Lovász and Plummer [18, Sect. 10.1]) to the perfect matching problem, which can be solved in polynomial time [1, Chap. 12]. By a modification of an algorithm of Hartvigsen [13], it is possible to show that 4-cycle covers of maximum weight in graphs with edge weights zero and one can be computed efficiently [19].

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 [7]. Hassin and Rubinstein [15] devised a randomised approximation algorithm for Max-$\{3\}$-UCC that achieves an approximation ratio of $169/89 + \epsilon$. Max-L-UCC can be approximated within a factor of 2.5 for arbitrary sets L [19].

Testing whether an undirected graph contains an L-cycle cover as a spanning subgraph is NP-hard if $\overline{L} \not\subseteq \{3,4\}$, i.e. for almost all L [16]. Vornberger showed that Max-5-UCC is NP-hard [22]. Max-L-UCC is APX-hard if $\overline{L} \not\subseteq \{3\}$ [19], i.e. for almost all L, Max-L-UCC is unlikely to possess a polynomial-time approximation scheme. (We refer to Ausiello et al. [2] for a survey on optimisation problems and their approximability.) Even a restriction of Max-L-UCC where only edge weights zero and one are allowed is APX-hard for all L with $\overline{L} \not\subseteq \{3,4\}$ [19].

Directed Cycle Covers. Max-$\mathcal{D}$-DCC, which is also known as the *assignment problem*, can be solved in polynomial time by a reduction to the maximum weight perfect matching problem in bipartite graphs [1, Chap. 12]. The only other L for which Max-L-DCC can be solved in polynomial time is $L = \{2\}$. For all $L \subseteq \mathcal{D}$ with $L \neq \{2\}$ and $L \neq \mathcal{D}$, Max-L-DCC is APX-hard and NP-hard, even if only edge weights zero and one are allowed [19].

There are a factor 4/3 approximation algorithm for Max-3-DCC [6] and a factor 3/2 approximation algorithm for computing k-cycle covers of maximum weight for $k \geq 3$ with the restriction that the only edge weights allowed are zero and one [4]. Max-L-DCC admits a factor 3 approximation for arbitrary L [19].

1.3 New Results

In this paper, we present approximation algorithms for Max-L-UCC and Max-L-DCC that work for arbitrary sets L. Our algorithms achieve an approximation ratio of 2 for Max-L-UCC (Section 3.1) and an approximation ratio of 20/7 for Max-L-DCC (Section 3.2). The best approximation algorithms previously known for these problems achieve ratios of 2.5 and 3, respectively [19].

As a main ingredient of the algorithms, we prove a decomposition lemma that shows how an arbitrary cycle cover can be decomposed while preserving as much of its weight as possible (Section 2).

Finally, we show the limits of decomposition-based approximation algorithms (Section 4): For approximating undirected L-cycle covers, a ratio of 2 is the best one can achieve using decomposition techniques. Thus, our algorithm for Max-L-UCC is an optimal decomposition-based algorithm. For directed L-cycle covers, only a ratio of 3 can be achieved. Our approximation algorithm for Max-L-DCC achieves the ratio of $20/7 < 3$ by a combination of the decomposition technique and a matching-based algorithm.

2 Decomposing Cycle Covers

In this section, we present a general decomposition technique for cycle covers. The technique can be applied to all cycle covers that do not contain 2-cycles, thus in particular to cycle covers of undirected graphs. But it can also be applied to directed cycle covers without 2-cycles.

We decompose cycle covers into a collection of vertex-disjoint singles, doubles, and isolated vertices. Our aim is to decompose a cycle cover C on n vertices into roughly $n/6$ singles and $n/6$ doubles. Thus, we retain half of the edges of C. We aim at decomposing the cycle covers such that at least half of the weight of the cycle cover is preserved.

The reason why we decompose cycle covers into singles and doubles is the following: If we decomposed them into longer paths, then we would run into trouble when trying to decompose a 3-cycle. If we restricted ourselves to decomposing the cycle covers into singles only, then 3-cycles would limit the weight preserved: We would get only one third of the edges of the 3-cycles, thus at most one third of their weight in general. Finally, if we restricted ourselves to doubles, then 5-cycles would limit the weight we could obtain since we would get only a fraction of $2/5$ of their edges.

In our approximation algorithms, we exploit the following observation: If every cycle cover on n vertices can be decomposed into α singles and β doubles, then, for every L, every L-cycle cover on n vertices can be decomposed in the same way. This implies that we can build cycle covers from such a decomposition: Given α singles and β doubles, and $n - 2\alpha - 3\beta$ isolated vertices, we can join them to form an L-cycle cover. (The only restriction is that there must exist L-cycle covers on n vertices. We refer to Section 3 for more details.)

If n is not divisible by six, we replace $n/6$ by $\lfloor n/6 \rfloor$ or $\lceil n/6 \rceil$: Assume that $n = 6k + \ell$ for $k, \ell \in \mathbb{N}$ and $\ell \leq 5$. Then we take $k + \alpha_\ell$ singles and $k + \beta_\ell$ doubles, where α_ℓ and β_ℓ are given in Table 1.

Since we want a decomposition weighing at least half of the weight of the cycle cover, we need to take at least half of the edges of the cycle cover. Otherwise, we would get a decomposition of less weight. It can be checked easily that by taking $k + \alpha_\ell$ singles and $k + \beta_\ell$ doubles, we obtain $\lceil n/2 \rceil$ edges of C.

ℓ	0	1	2	3	4	5
α_ℓ	0	1	1	0	0	1
β_ℓ	0	0	0	1	1	1

Table 1. A cycle cover on $n = 6k + \ell$ vertices will be decomposed into $k + \alpha_\ell$ singles and $k + \beta_\ell$ doubles.

Lemma 1 (Decomposition Lemma). *Let $C = (V, E)$ be a cycle cover on $n = 6k + \ell$ vertices such that the length of each cycle is at least three. Let $w : E \to \mathbb{N}$ be an edge weight function.*

Then there exists a decomposition $D \subseteq E$ of C with the following properties:

- *(V, D) consists of $k + \alpha_\ell$ singles, $k + \beta_\ell$ doubles, and $n - 5k - 3\beta_\ell - 2\alpha_\ell$ isolated vertices, such that all these subgraphs are pairwise vertex-disjoint, and*
- *$w(D) \geq \frac{1}{2} \cdot w(E)$.*

The following two lemmas will simplify the proof of the decomposition lemma.

Lemma 2. *Let $\lambda, \alpha, \beta \in \mathbb{N}$ with $\alpha + 2\beta \geq \lambda/2$ and $2\alpha + 3\beta \leq \lambda$. Then every cycle c of length λ can be decomposed into α singles and β doubles such that the weight of the decomposition is at least $w(c)/2$.*

Lemma 3. *Let $\lambda \in \mathbb{N}$. Suppose that every cycle c of length λ can be decomposed into α singles and β doubles of weight at least $w(c)/2$. Then every cycle c' of length $\lambda + 6$ can be decomposed into $\alpha + 1$ singles and $\beta + 1$ doubles of weight at least $w(c')/2$.*

Lemma 3 also holds if we consider more than one cycle: Assume that every collection of k cycles of lengths $\lambda_1, \ldots, \lambda_k$ can be decomposed into α singles and β doubles such that the weight of the decomposition is at least half the weight of the cycles. Then k cycles of lengths $\lambda_1 + 6, \lambda_2, \ldots, \lambda_k$ can be decomposed into $\alpha + 1$ singles and $\beta + 1$ doubles such that also at least half of the weight of the cycles is preserved.

Due to Lemma 3, we can restrict ourselves to cycles of length at most eight in the following. The reason for this is the following: If we know how to decompose cycles of length λ, then we also know how to decompose cycles of length $\lambda + 6, \lambda + 12, \ldots$ from Lemma 3.

We now come to the proof of the decomposition lemma. The decomposition described can clearly be done in polynomial time.

Proof (Decomposition Lemma). We prove the lemma by induction on the number of cycles. As induction basis, we consider two cases:

One cycle. Due to Lemma 3, we can restrict ourselves to considering cycles of length at most eight.

3-cycles and 4-cycles have to be decomposed into one double. 5-cycles and 6-cycles have to be decomposed into one double and one single. Finally, 7-cycles and 8-cycles have to be decomposed into two singles and one double. According to Lemma 2, these decompositions can be made such that at least half of the weight of the cycle is preserved.

Two odd cycles. The two cycles can be of length three, five, or seven. Thus, there are six cases to be considered. We describe exemplarily how to decompose two 3-cycles. The other five cases are treated similarly.
Six vertices are involved, thus we need one single and one double. A single can be chosen such that at least one third of the weight of the cycle is preserved. Analogously, a double can be chosen such that at least two thirds of the weight of the cycle are preserved. We take the double of the heavier cycle and the single of the lighter cycle. Both the single and the double are chosen such that their weight is maximised. Then their total weight is at least one half of the sum of the weight of the two cycles.

As induction hypothesis, we assume that the lemma holds if the number of cycles is less than r. Assume that we have a cycle cover C consisting of r cycles. Let $n = 6k + \ell$ for the number of its vertices for $k, \ell \in \mathbb{N}$ and $\ell \leq 5$. This means that C has to be decomposed into $k + \alpha_\ell$ singles and $k + \beta_\ell$ doubles. In the following, let C' be the new cycle cover obtained by removing one or two cycles of C.

We proceed as follows: First, we show how to remove an even cycle from C. Second, we show how to remove a pair of odd cycles from C. Special care is needed when removing a pair of one 3-cycle and one 5-cycle.

Let us start by considering the removal an even cycle. It suffices to consider cycles of length four, six, or eight. The easiest case is removing a 6-cycle: We decompose it into one single and one double preserving at least half of its weight. The new cycle cover C' consists of $n - 6$ vertices. Consequently, C' can be decomposed into $k + \alpha_\ell - 1$ singles and $k + \beta_\ell - 1$ doubles by the induction hypothesis. In addition, we have one single and one double from the 6-cycle. Thus, C can be decomposed into $k + \alpha_\ell$ singles and $k + \beta_\ell$ doubles such that at least half of its weight is preserved.

If we want to remove a 4-cycle or an 8-cycle, several cases have to be distinguished. A 4-cycle has to be decomposed either into one double or into two singles. This depends on the value of ℓ: If, for instance, $\ell = 4$, then C' consists of $6k$ vertices. Thus, C' has to be decomposed into k singles and k doubles. Since $\alpha_4 = 0$ and $\beta_4 = 1$, the 4-cycle has to be decomposed into one double. On the other hand, if $\ell = 2$, then C' consists of $6(k - 1) + 4$ vertices. Thus, C' has to be decomposed into k doubles and $k - 1$ singles, while C has to be decomposed into k doubles and $k + 1$ singles. In this case, we have to decompose the 4-cycle into two singles.

If $\ell \in \{0, 3, 4, 5\}$, the 4-cycle has to be decomposed into one double. Otherwise, i.e. if $\ell \in \{1, 2\}$, it has to be decomposed into two singles. By the induction hypothesis and Lemma 2, C' and the 4-cycle can be decomposed appropriately such that at least half of the weight of both is preserved.

Analogously, an 8-cycle has to be decomposed into two doubles if $\ell \in \{3, 4\}$ or into two singles and one double if $\ell \in \{0, 1, 2, 5\}$.

Now we consider removing a pair of odd cycles. We have to distinguish six cases as we already did in the proof of the induction basis. As an example, we consider the cases of two 3-cycles and of a 3-cycle and a 7-cycle. Furthermore, we consider the case of a 3-cycle and a 5-cycle since this case needs special attention.

Twice length three. We decompose the heavier of the 3-cycles into a double and the lighter one into a single. The new cycle cover C' has $n - 6$ vertices. It thus has to be decomposed into $k + \alpha_\ell - 1$ singles and $k + \beta_\ell - 1$ doubles. Plus one single and one double from the two 3-cycles, we obtain a feasible decomposition of C.

Length three and seven. If $\ell \in \{0, 3, 4, 5\}$, then we decompose the two cycles into one single and two doubles. We take either a double of the 3-cycle and a single and a double of the 7-cycle or a single of the 3-cycle and two doubles of the 7-cycle. This depends on which alternative yields more weight. In this way, we preserve at least half of the weight of the two cycles.

If $\ell \in \{1, 2\}$, then we take three singles and one double. Either we decompose the 3-cycle into a double and the 7-cycle into three singles or the 3-cycle into a single and the 7-cycle into two singles and one double. Again, we choose the alternative that yields more weight, and we preserve at least half of the weight of the cycle cover.

Length three and five. The case of a 3-cycle and 5-cycle is a bit more complicated than the other cases. We run into troubles if, for instance, $\ell = 3$. In this case, we have to decompose the two cycles into two doubles. If the 5-cycle is much heavier than the 3-cycle, then it is impossible to preserve half of the weight of the two cycles.

However, we can avoid the problem as follows: As long as there is an even cycle, we decompose this one. After that, as long as their are at least three odd cycles, we can choose two of them such that we do not have a pair of one $(3 + 6\xi)$-cycle and one $(5 + 6\xi')$-cycle for some $\xi, \xi' \in \mathbb{N}$.

Thus, the only situation in which we cannot avoid to decompose a $(3 + 6\xi)$-cycle and a $(5+6\xi')$-cycle is when there are only two cycles left. In this case, we have $\ell = 2$, and we have treated this case already in the induction basis.

If there is only one odd cycle, then either $r = 1$ or all other cycles are of even length. We have already dealt with the former case in the induction basis. In the latter case, we proceed by removing even cycles as described above. □

3 Approximation Algorithms

In this section, we apply the decomposition lemma to devise approximation algorithms for restricted cycle covers both in undirected and directed graphs. 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 corresponds to the halting problem, 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, Max-L-UCC and Max-L-DCC are NP optimisation problems. Another possibility for circumventing the problem is to include the permitted cycle lengths in the input. While such restrictions are mandatory when we want to compute optimum solutions, they are not needed for our approximation algorithms.

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 $\langle L \rangle = \{n \mid n \text{ is } L\text{-admissible}\}$. Although L can be arbitrarily complicated, $\langle L \rangle$ always allows efficient membership testing. In fact, it has been proved that for all $L \subseteq \mathbb{N}$, there exists a finite set $L' \subseteq L$ with $\langle L' \rangle = \langle L \rangle$ [19].

For every fixed L, we can not only test in time polynomial in n whether n is L-admissible, but we can, provided that $n \in \langle L \rangle$, also find numbers $\lambda_1, \ldots, \lambda_k \in L'$ that add up to n, where $L' \subseteq L$ denotes a finite set with $\langle L \rangle = \langle L' \rangle$. This can be done via dynamic programming in time $O(n \cdot |L'|)$, which is $O(n)$ for fixed L.

Instead of computing L'-cycle covers in the following two sections, we assume without loss of generality that already L is a finite set. This does not affect the approximation ratios achieved by our algorithms since the L-cycle covers computed are compared to optimal cycle covers without restrictions.

In general, our algorithms work as follows: They start by computing an initial cycle cover C^{init}. Then C^{init} is decomposed according to Lemma 1. Finally, the singles, doubles, and isolated vertices are joined to form an L-cycle cover C^{apx}. Since no weight is lost during the final merging, the weight of the decomposition is a lower bound for the weight of C^{apx}. With this approach, we can achieve approximation ratios of 2 and 3 for undirected and directed L-cycle covers, respectively (see Section 4, where a decomposition technique for directed graphs is sketched). We improve on the factor of 3 for directed graphs by using a more sophisticated algorithm.

3.1 Undirected Cycle Covers

Theorem 1. *Algorithm 1 is a factor* 2 *approximation algorithm for* Max-L-UCC *for all* $L \subseteq \mathcal{U}$.

Proof. Algorithm 1 returns $\perp$ if and only if $n \notin \langle L \rangle$. Otherwise, an L-cycle cover C^{apx} is returned. Let $C^\star$ denote an L-cycle cover of maximum weight of G. We have $w(C^\star) \leq w(C^{\text{init}}) \leq 2 \cdot w(D) \leq 2 \cdot w(C^{\text{apx}})$. □

The running-time of the algorithm is dominated by the time needed to compute the initial cycle cover, which is $O(n^3)$ according to Ahuja et al. [1, Chap. 12].

3.2 Directed Cycle Covers

For directed graphs, we cannot apply the decomposition lemma directly. The reason is that we have to cope with 2-cycles. Therefore, we balance two approaches. The first approach is a simple matching-based algorithm: We compute

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

if $n \notin \langle L \rangle$ **then**
  return $\perp$
compute a cycle cover C^{init} in G of maximum weight
decompose C^{init} into a set $D \subseteq C^{\text{init}}$ of edges according to Lemma 1
join the singles and doubles in D to obtain an L-cycle cover C^{apx}
return C^{apx}

Algorithm 1: A 2-approximation algorithm for Max-L-UCC.

a maximum-weight matching of a certain cardinality and join the edges of the matching to obtain an L-cycle cover. This works particularly well if an optimum cycle cover has much of its weight in cycles of even length. Since 2-cycles are even cycles, this works well if an optimum L-cycle cover has much of its weight in 2-cycles. The cardinality of the maximum is chosen such that an L-cycle cover can be built from such a matching. A cycle of length λ yields a matching of cardinality $\lfloor \lambda/2 \rfloor$. Thus, a matching of cardinality d in a graph of n vertices can be extended to form an L-cycle cover if and only if $d \leq D(n, L) = \max\bigl\{\sum_{i=1}^{k} \lfloor \lambda_i/2 \rfloor \mid k \in \mathbb{N}, \sum_{i=1}^{k} \lambda_i = n, \text{ and } \lambda_i \in L \text{ for } 1 \leq i \leq k\bigr\}$.

If an optimum L-cycle cover has much of its weight in cycles of length at least three, then we compute an approximate 3-cycle cover using an approximation algorithm by Bläser et al. [6]. This 3-cycle cover is then decomposed according to the decomposition lemma. A problem with this approach is that an optimum L-cycle cover may contain 2-cycles if $2 \in L$. But a collection of τ cycles of length two can be rejoined to form two τ-cycles for some $\tau \in L$. In this way, we lose at most two thirds of their weight. We still might have to cope with $\xi < \tau$ cycles of length two. Since L is fixed, τ is a constant. Thus, we can simply try all subsets of vertices of even cardinality $2\xi \leq 2\tau - 2$, join them to form ξ cycles of length two, and remove them to proceed with the approximation algorithm on the remaining graph.

Theorem 2. *Algorithm 2 is a factor* 20/7 *approximation algorithm for* Max-L-DCC *for all* $L \subseteq \mathcal{D}$.

Proof. First, we consider the case that $2 \notin L$. The algorithm starts by computing a 4/3-approximation C_3^{init} to an optimal 3-cycle cover by using an algorithm of Bläser et al. [6]. This cycle cover, which does not contain 2-cycles, is then decomposed into singles and doubles according to the decomposition lemma. The resulting set D of edges has a weight of at least 3/8 of an optimal L-cycle cover. By joining the singles and doubles of D to an L-cycle cover, we obtain a factor 8/3 approximation. Note that $8/3 < 20/7$.

If $L = \{2\}$, then we can solve the problem optimally in polynomial time.

What remains to be considered is the case that $2 \in L$ and $L \neq \{2\}$. The main idea in this case is that we balance two approaches for computing L-cycle covers: One is that we form an L-cycle cover from a (not necessarily perfect) matching, which works particularly well if much of the weight of an optimum L-cycle cover

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

if $n \notin \langle L \rangle$ **then**
    return $\perp$
if $2 \notin L$ **then**
    compute a 4/3-approximation C_3^{init} to an optimal 3-cycle cover
    decompose C_3^{init} into a set $D \subseteq C_3^{\mathrm{init}}$ of edges according to Lemma 1
    join the singles and doubles in D to obtain an L-cycle C^{apx}
else if $L = \{2\}$ **then**
    compute a $\{2\}$-cycle cover C^{apx} of maximum weight
else
    compute a matching M of cardinality at most $D(n, L)$ that has maximum weight among all such matchings
    construct an L-cycle cover $C_{\mathrm{match}}^{\mathrm{apx}} \supseteq M$
    $\tau \leftarrow \min\{L \setminus \{2\}\}$
    for all $\xi \leftarrow 0, 2, 4, \ldots, 2\tau - 2$ **do**
        for all $V' \subseteq V$ of cardinality ξ **do**
            compute a maximum weight $\{2\}$-cycle cover $C_{2,V'}$ on V'
            remove V' from G to obtain G'
            compute a 4/3-approximation C_3^{init} to an optimal 3-cycle cover of G'
            decompose C_3^{init} into a set $D \subseteq C_3^{\mathrm{init}}$ of edges according to Lemma 1
            join the singles and doubles in D to obtain an L-cycle cover $C_{V'}^{\mathrm{apx}}$
            add $C_{2,V'}$ to $C_{V'}^{\mathrm{apx}}$
    let C_3^{apx} be the cycle cover of maximum weight among all $C_{V'}^{\mathrm{apx}}$
    let C^{apx} be the heavier cycle cover of $C_{\mathrm{match}}^{\mathrm{apx}}$ and C_3^{apx}
return C^{apx}

Algorithm 2: A 20/7-approximation algorithm for Max-L-DCC.

is contained in even cycles. The other approach is using the 4/3-approximation algorithm by Bläser et al. [6] and the decomposition lemma. This works well if an optimum L-cycle cover does not contain too much weight in 2-cycles. We omit the details of the proof due to space constraints. □

4 Limits for Decomposition-Based Algorithms

The aim of this section is to fathom the possibilities of designing approximation algorithms for L-cycle covers that base on cycle decompositions as described in Section 2.

An approximation ratio of 2 is the best possible for undirected L-cycle covers. Hence, the algorithm presented in Section 3.1 is an optimal decomposition-based algorithm. For further improvements of the approximation ratio, we thus need more sophisticated techniques that in particular take the set L into account.

For directed L-cycle covers, already the previously known factor 3 approximation algorithm [19] can be viewed as a decomposition algorithm: Every directed cycle cover on n vertices can be decomposed into $\lceil n/3 \rceil$ singles such that at least one third of the weight of the cycle cover is preserved.

We have presented an algorithm for directed cycle covers that exploits properties of the set L: The bottleneck for the decomposition-based approach are cycles of length two since they can only be decomposed into paths of length one. By taking special care of 2-cycles, and by applying an approximation algorithm for 3-cycle covers, we were able to achieve the improved approximation ratio of 20/7.

Overall, every approximation algorithm for Max-L-UCC that works for arbitrary sets L and is purely decomposition-based achieves at best an approximation ratio of 2. For Max-L-DCC, such algorithms achieve achieve at best a ratio of 3.

5 Conclusions

One way to get better approximation algorithms is balancing several approximation algorithms as we did to achieve the ratio of 20/7 for directed L-cycle covers. One option to do this for undirected graphs might be to start with a 4-cycle cover instead of 3-cycle cover. This is possible for approximating Max-L-UCC restricted to edge weights zero and one since the corresponding 4-cycle cover problem can be solved in polynomial time [13, 19]. Another option is to use approximation algorithms for 4-cycle covers. In either case, we need a decomposition lemma that preserves more than half of the weight of the cycle cover.

Finally, from a more abstract point of view, we are interested in structural properties of restricted cycle covers: Let u_L and d_L denote the best approximation ratios for undirected and directed L-cycle covers, respectively, that can be achieved by polynomial-time algorithms. What is the minimum number $u^\star$ such that all L-cycle cover problems can be approximated with a ratio of $u^\star$, i.e. what is $\sup_{L \subseteq \mathcal{U}, L \neq \emptyset} u_L$? Analogously, what is the minimum number $d^\star$ such that all L-cycle cover problems can be approximated with a ratio of $d^\star$, i.e. what is $\sup_{L \subseteq \mathcal{D}, L \neq \emptyset} d_L$? For the moment, we know $u^\star \leq 2$ and $d^\star \leq 20/7$. On the other hand, does there exist an $r > 1$ as a general lower bound for the approximability of L-cycle cover problems? What we mean by a general lower bound r is the following: If an L-cycle cover problem is NP-hard, then it cannot be approximated with a ratio of less than r unless $\mathsf{P} = \mathsf{NP}$. The reductions known so far do not yield such a general lower bound.

References

1. Ravindra K. Ahuja, Thomas L. Magnanti, James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, 1993.
2. Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, Marco Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties.* Springer, 1999.
3. Markus Bläser. A 3/4-approximation algorithm for maximum ATSP with weights zero and one. In *Proc. of the 7th Int. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, vol. 3122 of *Lecture Notes in Comput. Sci.*, pp. 61–71. Springer, 2004.

4. Markus Bläser, Bodo Manthey. Approximating maximum weight cycle covers in directed graphs with weights zero and one. *Algorithmica*, 42(2):121–139, 2005.
5. Markus Bläser, Bodo Manthey, Jiří Sgall. An improved approximation algorithm for the asymmetric TSP with strengthened triangle inequality. *J. Discrete Algorithms*, to appear.
6. Markus Bläser, L. Shankar Ram, 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.
7. Markus Bläser, 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.
8. Avrim L. Blum, Tao Jiang, Ming Li, John Tromp, Mihalis Yannakakis. Linear approximation of shortest superstrings. *J. ACM*, 41(4):630–647, 1994.
9. Hans-Joachim Böckenhauer, Juraj Hromkovič, Ralf Klasing, Sebastian Seibert, Walter Unger. Approximation algorithms for the TSP with sharpened triangle inequality. *Inform. Process. Lett.*, 75(3):133–138, 2000.
10. L. Sunil Chandran, L. Shankar Ram. Approximations for ATSP with parameterized triangle inequality. In *Proc. of the 19th Int. Symp. on Theoretical Aspects of Computer Science (STACS)*, vol. 2285 of *Lecture Notes in Comput. Sci.*, pp. 227–237. Springer, 2002.
11. Zhi-Zhong Chen, Takayuki Nagoya. Improved approximation algorithms for metric Max TSP. In *Proc. of the 13th Ann. European Symp. on Algorithms (ESA)*, vol. 3669 of *Lecture Notes in Comput. Sci.*, pp. 179–190. Springer, 2005.
12. Zhi-Zhong Chen, Yuusuke Okamoto, Lusheng Wang. Improved deterministic approximation algorithms for Max TSP. *Inform. Process. Lett.*, 95(2):333–342, 2005.
13. David Hartvigsen. *An Extension of Matching Theory.* PhD thesis, Carnegie Mellon University, Pittsburgh, USA, 1984.
14. Refael Hassin, Shlomi Rubinstein. On the complexity of the k-customer vehicle routing problem. *Oper. Res. Lett.*, 33(1):71–76, 2005.
15. Refael Hassin, Shlomi Rubinstein. An approximation algorithm for maximum triangle packing. *Discrete Appl. Math.*, 154(6):971–979, 2006.
16. Pavol Hell, David G. Kirkpatrick, Jan Kratochvíl, Igor Kríz. On restricted two-factors. *SIAM J. Discrete Math.*, 1(4):472–484, 1988.
17. Haim Kaplan, Moshe Lewenstein, Nira Shafrir, Maxim Sviridenko. Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. *J. ACM*, 52(4):602–626, 2005.
18. László Lovász, Michael D. Plummer. *Matching Theory*, vol. 121 of *North-Holland Mathematics Studies*. Elsevier, 1986.
19. Bodo Manthey. On approximating restricted cycle covers. In *Proc. of the 3rd Workshop on Approximation and Online Algorithms (WAOA 2005)*, vol. 3879 of *Lecture Notes in Comput. Sci.*, pp. 282–295. Springer, 2006.
20. Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, vol. 24 of *Algorithms and Combinatorics*. Springer, 2003.
21. Z. Sweedyk. A $2\frac{1}{2}$-approximation algorithm for shortest superstring. *SIAM J. Comput.*, 29(3):954–986, 1999.
22. Oliver Vornberger. Easy and hard cycle covers. Technical report, Universität/Gesamthochschule Paderborn, 1980.