






KAI HOPPMANN-BAUM¹, GIONI MEXI², OLEG
BURDAKOV³, CARL JOHAN CASSELGREN⁴, THORSTEN
KOCH⁵

Length-Constrained Cycle Partition with an Application to UAV Routing

¹  0000-0001-9184-8215
²  0000-0003-0964-9802
³  0000-0003-1836-4200
⁴  0000-0002-2741-468X
⁵  0000-0002-1967-0077

Zuse Institute Berlin
Takustr. 7
14195 Berlin
Germany

Telephone: +49 30-84185-0
Telefax: +49 30-84185-125

E-mail: bibliothek@zib.de
URL: <http://www.zib.de>

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Length-Constrained Cycle Partition with an Application to UAV Routing

Kai Hoppmann^{1,2}[0000-0001-9184-8215], Gioni Mexi¹[0000-0003-0964-9802],
Oleg Burdakov³[0000-0003-1836-4200], Carl Johan
Casselgren³[0000-0002-2741-468X], and Thorsten Koch^{1,2}[0000-0002-1967-0077]

¹ Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany
{kai.hoppmann,mexi,koch}@zib.de

² TU Berlin, Chair of Software and Algorithms for Discrete Optimization, Str. des
17. Juni 135, 10623 Berlin, Germany

³ Linköping University, Department of Mathematics, SE-58183 Linköping, Sweden
{carl.johan.casselgren,oleg.burdakov}@liu.se

Abstract. In this article, we discuss the Length-Constrained Cycle Partition Problem (LCCP). Besides edge weights, the undirected graph in LCCP features an individual critical weight value for each vertex. A cycle partition, i.e., a vertex disjoint cycle cover, is a feasible solution if the length of each cycle is not greater than the critical weight of each of the vertices in the cycle. The goal is to find a feasible partition with the minimum number of cycles. In this article, we discuss theoretical properties, preprocessing techniques, and two mixed-integer programming models (MIP) for LCCP both inspired by formulations for the closely related Travelling Salesperson Problem (TSP). Further, we introduce conflict hypergraphs, whose cliques yield valid constraints for the MIP models. We conclude with a report on computational experiments conducted on (A)TSPLIB-based instances. As an example, we use a routing problem in which a fleet of uncrewed aerial vehicles (UAVs) patrols a set of areas.

1 Motivation

Providing a service to a set of customers on a periodical basis is a recurring task being inherent to many real-world applications. Therefore, the corresponding optimization problems, where it often is the goal to determine the minimum number of agents needed to satisfy the service requirements while taking several resource restrictions into account, represent an area of active research. Besides classical examples, which include the delivery of gasoline to service stations [5] or timetabling in public transport [15,22], problems of this kind frequently arise in the context of routing uncrewed aerial vehicles (UAVs).

UAVs are widely used to execute surveillance tasks, since they can gather information about an area from long distance or high altitude. In particular, they are able to visit areas that are not accessible in any other way. Therefore, they are used to monitor critical infrastructure, such as natural gas pipelines [12], to fight forest-fires [17], or to analyze widespread animal populations [3]. The

Length-Constrained Cycle Partition Problem (LCCP) originates from a routing problem regarding these UAVs.

Given a set of areas $V = \{v_1, \dots, v_n\}$, the goal is to determine the minimum number of UAVs necessary to patrol them, while their individual flying routes have to satisfy three conditions: First, the UAVs must fly tours, which means that a UAV starts and ends its route at the same area and visits all other areas, which are assigned to it, exactly once. We assume that a UAV continues on the same tour after finishing it without any delay. Second, each area is visited by exactly one UAV and therefore contained in exactly one tour. This is to avoid possible interferences resulting from intersections. Third, each area $v_i \in V$ is associated with a critical weight value $T_i \in \mathbb{R}_{\geq 0}$, which is an upper bound on the duration for which it can be left unattended, and a scanning time $S_i \in \mathbb{R}_{\geq 0}$, which is the amount of time a UAV needs to scan it. We require that after scanning v_i for S_i time units, the UAV assigned to it has to return and rescan it within T_i time units.

2 Problem Formulation

For LCCP we are given a complete graph $G = (V, E)$ where $V = \{v_1, \dots, v_n\}$ denotes the set of vertices and $E \subseteq V \times V$ the set of edges. For each vertex $v_i \in V$ we are given a *critical weight* $T_i \in \mathbb{R}_{\geq 0}$ and a *scanning time* $S_i \in \mathbb{R}_{\geq 0}$ with $S_i \leq T_i$. The *edge weight* of $e_{ij} := \{v_i, v_j\} \in E$ is given as $\hat{L}_{ij} \in \mathbb{R}_{\geq 0}$. An LCCP instance is called *metric* if the edge weights obey the triangle inequality.

Next, a cycle in G is a tuple $C_k = (V_k, E_k)$, where $E_k = (e_1, \dots, e_\Omega)$ denotes a closed walk that joins the vertices in V_k while all vertices except for the first and the last one are distinct. We call a cycle C_k *proper* if $|V_k| \geq 2$, a *singleton* if $|V_k| = 1$, and *empty* otherwise. Furthermore, C_k is called *feasible* if $\tau_k \leq T_i$ holds for each $v_i \in V_k$, where

$$\tau_k := \sum_{e_{ij} \in E_k} \hat{L}_{ij} + \sum_{v_i \in V_k} S_i$$

is the *length* of C_k , i.e., if the length of the cycle is not greater than the critical weight values of its vertices. A solution for LCCP is a *cycle partition* $\mathcal{C} = \{C_1, \dots, C_m\}$ of V , i.e., a vertex disjoint cycle cover, and \mathcal{C} is called feasible if all of its cycles are feasible. The goal of LCCP is to determine a feasible cycle partition \mathcal{C} of minimum size w.r.t. the cardinality $|\mathcal{C}|$.

In the following, we assume w.l.o.g. that $S_i = 0$ for all vertices $v_i \in V$, since we can add the scanning times to the edge weights: Let $L_{ij} := \hat{L}_{ij} + \frac{S_i + S_j}{2}$ for $e_{ij} \in E$ and consider a cycle C_k . If it is proper, all vertices have degree two and

$$\tau_k = \sum_{e_{ij} \in E_k} \hat{L}_{ij} + \sum_{v_i \in V_k} S_i = \sum_{e_{ij} \in E_k} \left(\hat{L}_{ij} + \frac{S_i + S_j}{2} \right) = \sum_{e_{ij} \in E_k} L_{ij}.$$

Additionally, all singletons remain feasible by definition, too. Note that a metric LCCP instance remains metric after this transformation, i.e., the triangle in-

equality is preserved. Therefore, we are going to denote an instance of LCCP as a four-tuple (V, E, T, L) with $T \in \mathbb{R}_{\geq 0}^{|V|}$ and $L \in \mathbb{R}_{> 0}^{|E|}$ in the following.

Consider the example LCCP instance in Figure 1a. The critical weight values and the edge weights are attached to the corresponding entities. While Figures 1b and 1c show infeasible solutions, an optimal one is presented in 1d.

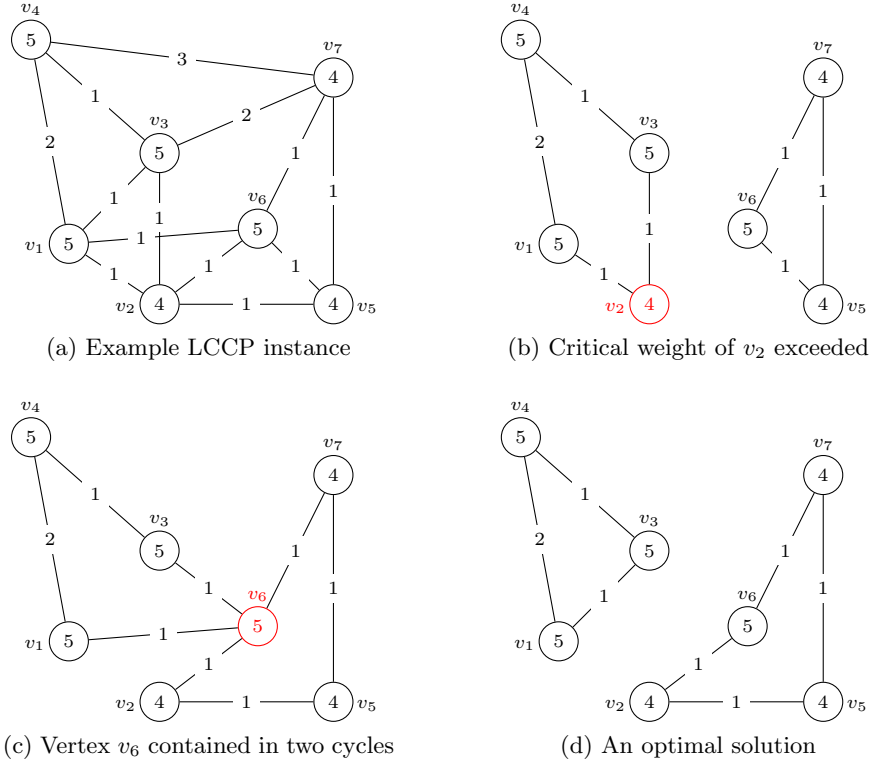


Fig. 1: Example LCCP instance with two infeasible and one optimal solution.

In [14] it was shown that metric LCCP is NP-hard via a reduction from TSP and the corresponding proof demonstrates its close relation to TSP. However, for general edge weights, we can additionally state the next lemma.

Lemma 1. *LCCP is not in APX.*

Proof. Consider the Minimum Vertex Disjoint Cycle Cover Problem (MVDCC), which was shown not to be in APX by Sahni and Gonzalez [21]. MVDCC is defined on an unweighted graph $\tilde{G} = (\tilde{V}, E')$ with E' not necessarily being complete, and it is the goal to determine a minimum vertex disjoint cycle cover. Let \tilde{E} denote the complete edge set on \tilde{V} . Setting $\tilde{T}_i = n$ for all $v_i \in \tilde{V}$, $\tilde{L}_{ij} = 1$

for all $e_{ij} \in E'$ and $\tilde{L}_{ij} = n$ for all $e_{ij} \in \tilde{E} \setminus E'$, we derive a corresponding LCCP instance $(\tilde{V}, \tilde{E}, \tilde{T}, \tilde{L})$. Obviously, any constant approximation algorithm for LCCP would induce a constant approximation algorithm for MVDCC. \square

However, whether metric LCCP is an APX or not remains an open question.

3 Related work

Optimization problems where the vertices of a graph have to be visited under various timing or length conditions are a field of active research. To the best of our knowledge, LCCP was first introduced by Hoppmann et. al. in [14]. They considered the metric case only. In contrast, the paper at hand generalizes the problem considering general edge weights. In the following, we discuss problems, that are closely related to LCCP.

Drucker et al. consider the Cyclic Routing of UAVs (CR-UAV) problem in [7], which is a generalization of LCCP. In CR-UAV, closed walks, which have to start and end at the same vertex but can visit vertices and edges multiple times, have to be determined. Additionally, waiting at vertices is possible and the routes are allowed to intersect. The goal is to determine the minimum number of UAVs necessary to jointly satisfy the critical weight requirements of all vertices. Ho and Ouaknine [13] showed that the corresponding decision problem is PSPACE-complete even in the case of a single UAV. In [9] a solution approach based on solving satisfiability problems w.r.t. a fixed number of UAVs and so-called slots, which correspond to arrival times of UAVs at vertices, is presented. Since the necessary number of slots is not known in advance, the proposed method is incomplete and does not guarantee an optimal solution. However, in [8] a reduction to model-checking is suggested and a complete algorithm, which runs a bounded model checker for detecting feasible solutions and an explicit-state search attempting to prove their absence in parallel, is presented. Asghar et al., who synonymously call the problem Multi-Robot Routing for Persistent Monitoring with Latency Constraints, develop a factor $\mathcal{O}(\log \rho)$ approximation algorithm in [1], where ρ is the ratio of the maximum and the minimum critical weight value. They partition the vertices w.r.t. their critical weights and subsequently solve a Minimum Cycle Cover Problem (MCCP) on each subset.

Given a graph $G = (V, E)$ and some λ , MCCP is to determine the minimum number of cycles that cover the vertex set, such that the length of each cycle is not greater than λ . For the metric version of this NP-hard problem an $\frac{32}{7}$ approximation algorithm is introduced in [25]. Further, as demonstrated and introduced in the proof of Lemma 1 above, the Minimum Vertex Disjoint Cycle Cover Problem (MVDCC) [21] is a problem from combinatorial optimization, for which LCCP can be seen as generalization of.

Besides TSP, of which LCCP can be seen a generalization of, see [14], there are several other well-known combinatorial optimization problems, which are closely related to LCCP. One of them is the Vehicle Routing Problem with Time Windows (VRPTW), see Solomon and Desrosiers [23] or Desrochers et al. [6]

for surveys on the topic. The goal is to determine a collection of routes for a fleet of homogeneous vehicles. The routes have to start and end at a common depot v_0 and jointly visit a given set of customers $\{v_1, \dots, v_n\}$. In doing so, each customer $v_i \in V$ has some service requirement q_i which has to be satisfied within a time window $[l_i, u_i]$ by exactly one of the vehicles. One of the most studied objectives for VRPTW is to minimize the number of necessary vehicles while the accumulated requirements of the customers are not allowed to exceed the capacities Q of the assigned vehicles.

4 Preprocessing

Next, we show how edges, which cannot be part of any feasible cycle, can be identified. Before proving a corresponding condition, we first of all introduce the notion of a completion for an edge.

Definition 1. Let (V, E, T, L) be an LCCP instance and $e_{ij} \in E$. A completion for e_{ij} is a path p_{ij} between v_i and v_j having length $\ell(p_{ij}) := \sum_{e_{st} \in p_{ij}} L_{st}$ such that $\ell(p_{ij}) + L_{ij} \leq T_s$ for all $v_s \in p_{ij}$. It is called a shortest completion if it has minimum length among all completions.

Lemma 2. Let (V, E, T, L) be an LCCP instance, $e_{ij} \in E$, and p_{ij} be a shortest completion for e_{ij} . Then $\ell(p_{ij}) + L_{ij}$ is a tight lower bound on the length of a proper feasible cycle containing e_{ij} .

Proof. Let C_k with length τ_k be a feasible cycle with $e_{ij} \in E_k$. C_k can be split into e_{ij} and a completion \hat{p}_{ij} . Thus,

$$\tau_k = \ell(\hat{p}_{ij}) + L_{ij} \geq \ell(p_{ij}) + L_{ij},$$

and since the cycle induced by p_{ij} and e_{ij} is feasible, the bound is tight. \square

Corollary 1. Let (V, E, T, L) be an LCCP instance. If there exists no completion for some edge e_{ij} , there exists no feasible cycle containing it.

Remark 1. In the metric case, if an edge $e_{ij} \in E$ possesses a completion, then e_{ij} is a shortest completion for itself due to the triangle inequality. Hence, there exists no completion for an edge $e_{ij} \in E$ if $2 \cdot L_{ij} > \min\{T_i, T_j\}$.

The next lemma shows that determining a shortest completion or to detect that no completion exists can be done in polynomial time w.r.t. the size of G .

Lemma 3. Let (V, E, T, L) be an LCCP instance and let $e_{ij} \in E$. Determining a shortest completion for e_{ij} or to prove that no completion exists can be done in $\mathcal{O}(|V||E| + |V|^2 \cdot \log|V|)$.

Proof. Consider Algorithm 1. In each iteration a shortest path p_{ij} between v_i and v_j is computed. Next, we check if the critical weight values of all contained vertices are respected. If this is the case, we have found a shortest completion.

Otherwise, we remove all vertices from G whose critical weight values are smaller than $\ell(p_{ij}) + L_{ij}$ and continue with the next iteration. The removal of vertices does not decrease the length of a shortest path and therefore these vertices can never be part of any completion for e_{ij} . If at some point v_i or v_j is removed, no completion exists. Using the algorithm of Fredman and Tarjan [10], a shortest

Algorithm 1 Shortest completion for edge e_{ij}

```

1: while  $v_i, v_j \in G$  do
2:    $p_{ij} \leftarrow$  Shortest path between  $v_i$  and  $v_j$  in  $G$ 
3:   if  $\ell(p_{ij}) + L_{ij} \leq T_s$  for all  $v_s \in p_{ij}$  then
4:     return  $p_{ij}$ 
5:    $G \leftarrow G - \{v_s \in G \mid \ell(p_{ij}) + L_{ij} > T_s\}$ 
6: return  $\emptyset$ 

```

path in a weighted undirected graph can be computed in $\mathcal{O}(|E| + |V| \cdot \log|V|)$. Further, since in each iteration at least one vertex is removed or the algorithm terminates, we have at most $|V| - 1$ iterations. Thus, Algorithm 1 has a runtime of $\mathcal{O}(|V||E| + |V|^2 \cdot \log|V|)$. \square

The previous result gives rise to Algorithm 2, which is based on Algorithm 1 and deletes all edges from the graph for which no completion exists.

Algorithm 2 Delete edges without completion

```

1: for  $e_{ij}$  in  $E$  in non-descending order w.r.t.  $L_{ij}$  do
2:   if  $e_{ij}$  has no completion then
3:      $G \leftarrow G - e_{ij}$ 

```

Lemma 4. *Let (V, E, T, L) be an LCCP instance. Algorithm 2 deletes all edges without completion from G .*

Proof. Iterating over the edges in non-descending order w.r.t. to their weights ensures that all edges without completion are removed. Assume there is an edge $e_{ij} \in E$ such that another edge e_{kl} is removed from some shortest completion p_{ij} for e_{ij} in a subsequent iteration of Algorithm 2. Due to the ordering of the edges we know that $L_{ij} \leq L_{kl}$ and since e_{ij} is a completion for itself, we have

$$L_{kl} + L_{ij} \leq \ell(p_{ij}) + L_{ij} \leq \ell(e_{ij}) + L_{ij} = L_{ij} + L_{ij} \leq L_{kl} + L_{ij}$$

and therefore e_{ij} is a shortest completion, too. \square

By Lemma 4, applying Algorithm 2 deletes all edges without completion from G . Thus, for all remaining edges there exists a completion and therefore a feasible cycle containing it.

5 Conflict Hypergraphs for LCCP

In this section, we introduce conflict hypergraphs for LCCP. These graphs contain information about subsets of vertices which cannot be contained in common feasible cycles. In particular, we are interested in cliques of these graphs, since they give rise to constraints that can be imposed on the vertices when modelling LCCP as mathematical program.

Definition 2. A hypergraph is a pair $H = (V, E^H)$ with vertex set V and hyperedge set E^H . A hyperedge $e_S \in E^H$ is a subset $S \subseteq V$. H is called a c -uniform hypergraph or c -hypergraph if $|S| = c$ holds for all hyperedges $e_S \in E^H$.

Definition 3. Let $H_c = (V, E_c^H)$ be a c -hypergraph. A hyperclique is a set $U \subseteq V$ such that for each subset $S \subseteq U$ with $|S| = c$ we have $e_S \in E_c^H$.

Definition 4. Let (V, E, T, L) be an instance of LCCP. Its conflict c -hypergraph $H_c = (V, E_c^H)$ has V as vertex set and there is a hyperedge $e_S \in E_c^H$ if no feasible cycle containing all vertices of the subset $S \subseteq V$ with $|S| = c$ exists.

Corollary 2. Let (V, E, T, L) be an LCCP instance and let $U \subseteq V$ be a hyperclique in H_c . A feasible cycle C_k can contain at most $c - 1$ vertices from U , i.e., $|U \cap V_k| \leq c - 1$.

Lemma 5. Let (V, E, T, L) be an LCCP instance and let $U \subseteq V$ be a hyperclique of size $|U| = m$ in H_c . For each feasible cycle partition \mathcal{C} we have $|\mathcal{C}| \geq \lceil \frac{m}{c-1} \rceil$.

Proof. Let \mathcal{C} be a cycle partition with $|\mathcal{C}| < \lceil \frac{m}{c-1} \rceil$. By the pigeonhole principle there exists a cycle containing at least c vertices from U , which is a contradiction to Corollary 2. \square

Corollary 3. Let (V, E, T, L) be an LCCP instance and let $U \subseteq V$ with $|U| = m$ be a maximum clique in H_c . Then $\lceil \frac{m}{c-1} \rceil$ is a lower bound on the size of an optimal cycle partition.

Determining a shortest cycle containing a given subset $S \subseteq V$ of vertices is an NP-hard problem, since TSP can be reduced to it. However, for pairs of vertices, i.e., for $|S| = 2$, it can be done in polynomial time.

Lemma 6. Let (V, E, T, L) be an LCCP instance and let $v_i, v_j \in V$. A smallest feasible cycle w.r.t. the length containing both vertices can be determined in $\mathcal{O}(|V||E| + |V|^2 \cdot \log|V|)$.

Proof. Each feasible cycle containing v_i and v_j can be split into two vertex disjoint, except for the start- and endnode of course, paths p_{ij} and q_{ij} . Hence, we can equivalently determine two such paths minimizing the expression $\ell(p_{ij}) + \ell(q_{ij})$ in order to determine a smallest feasible cycle containing both vertices. This can be done with a modified version of Suurballe's algorithm [24], see Algorithm 3.

Algorithm 3 Shortest feasible cycle containing v_i and v_j

```

1: while  $v_i, v_j \in G$  do
2:    $p_{ij}, q_{ij} \leftarrow$  Two vertex disjoint  $v_i$ - $v_j$ -paths in  $G$  minimizing  $\ell(p_{ij}) + \ell(q_{ij})$ 
3:   if  $\ell(p_{ij}) + \ell(q_{ij}) \leq T_l$  for all  $v_l \in p_{ij} \cup q_{ij}$  then
4:     return Cycle induced by  $p_{ij}$  and  $q_{ij}$ 
5:    $G \leftarrow G - \{v_l \in G \mid \ell(p_{ij}) + \ell(q_{ij}) > T_l\}$ 
6: return  $\emptyset$ 

```

In each iteration two vertex disjoint shortest paths p_{ij} and q_{ij} w.r.t. to the sum of their lengths are determined. We check if the critical weights of the vertices contained in both paths are respected. If this is the case, p_{ij} and q_{ij} induce a shortest feasible cycle containing v_i and v_j . Otherwise, we remove all vertices from G whose critical weights are smaller than $\ell(p_{ij}) + \ell(q_{ij})$ and continue with the next iteration. Since the removal of vertices does not decrease the sum of the lengths of two vertex disjoint paths, the removed vertices cannot be part of any feasible cycle containing v_i and v_j . Furthermore, since at least one vertex is removed after each iteration, there are at most $|V| - 1$ iterations. If v_i or v_j is removed, there exists no feasible cycle containing both vertices and the algorithm terminates. Determining two vertex disjoint paths in a weighted undirected graph can be done in $\mathcal{O}(|E| + |V| \cdot \log|V|)$ using Suurballe's algorithm. Hence, the total running time of Algorithm 3 is $\mathcal{O}(|V||E| + |V|^2 \cdot \log|V|)$. \square

Applying Algorithm 3 to each vertex pair $S := \{v_i, v_j\} \subseteq V$, we can check whether $e_S \in E_2^H$ or not and thereby create the complete conflict graph H_2 .

Remark 2. In each iteration of Algorithm 3, we delete the vertices contained in the set $\{v_l \in G \mid \ell(p_{ij}) + \ell(q_{ij}) > T_l\}$ from G . This implies there does not exist a feasible cycle containing $S := \{v_i, v_j, v_l\} \subseteq V$ for each deleted vertex v_l . Hence, the corresponding hyperedges e_S are contained in H_3 .

6 LCCP and Vertex Coloring

LCCP and its conflict hypergraphs are closely related to vertex coloring. Recall the definition of the chromatic number.

Definition 5. *Let H be a hypergraph. The chromatic number $\chi(H)$ is the minimum number of colors needed to color the vertices such that no hyperedge is monochromatic, i.e., each hyperedge contains vertices of at least two colors.*

Lemma 7. *Let (V, E, T, L) be an LCCP instance and consider the hypergraph $H := (V, \bigcup_{r=2}^n E_r^H)$, i.e., the union of its c -conflict graphs. $\chi(H)$ is a lower bound on the size of an optimal cycle partition.*

Proof. The cycles of an optimal cycle partition are feasible by definition. Hence, no subset of vertices contained in a common cycle forms a hyperedge in H . Therefore, assigning the same color to all vertices in a common cycle results in a feasible coloring for H . \square

Corollary 4. *Let (V, E, T, L) be an LCCP instance and let H_c be its corresponding conflict c -hypergraph. Then $\chi(H_c)$ is a lower bound on the size of an optimal cycle partition.*

For metric LCCP, we can prove the following correspondence lemma.

Lemma 8. *Let (V, E, T, L) be an LCCP instance and consider the hypergraph $H := (V, \bigcup_{r=2}^n E_r^H)$. The size of a minimum cycle partition is equal to the chromatic number H . In particular, there exists a one-to-one mapping of feasible cycle partitions and feasible colorings of H .*

Proof. Given a feasible coloring of H , there exist a feasible cycle C containing all vertices of the same color. W.l.o.g. we can assume that this cycle C does not contain vertices of any other colour, since we can remove them due to the triangle inequality. Therefore, the colors induce a feasible cycle partition which has the same size as the coloring. On the other hand, given a feasible cycle partition, assigning all vertices of a feasible cycle the same color yields a feasible coloring by definition. \square

Lemma 8 does not hold for the general case as the example LCCP instance in Figure 2 demonstrates. H_2 has no edges, since for any two vertices v_i, v_j with $i, j \in \{1, 2, 3, 4\}$ and $i < j$ we have that $\{v_i, v_j, v_5\}$ is a feasible cycle. Accordingly, for H_3 we have $E_3^H = \{\{v_1, v_2, v_3\}, \{v_1, v_2, v_4\}, \{v_1, v_3, v_4\}, \{v_2, v_3, v_4\}\}$, since all cycles containing three vertices from $V - v_5$ contain at least two edges with weight 3. For the same reason, H_4 and H_5 are complete. Now, assigning v_1 and v_2 as well as v_3, v_4 , and v_5 common colors yields a feasible coloring for H and we have $\chi(H) = 2$. However, a minimum cycle partition is given by $\mathcal{C} = \{(\{v_1\}, \emptyset), (\{v_2\}, \emptyset), (\{v_3, v_4, v_5\}, (\{v_3, v_4\}, \{v_4, v_5\}, \{v_5, v_3\}))\}$ with $|\mathcal{C}| = 3$.

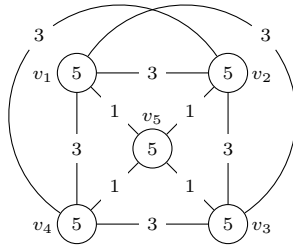


Fig. 2: LCCP instance for which Lemma 8 does not hold.

7 A Cheapest Insertion Heuristic for LCCP

Next, we present a Cheapest Insertion Heuristic (CI) for LCCP inspired by the homonymous TSP heuristic [20]. The basic version is stated in Algorithm 4.

In each iteration, a vertex $v_x \in V$ having minimum critical weight is selected, i.e., $v_x = \operatorname{argmin}_{v_i \in V} T_i$. If v_x is the only vertex in G or if there exists no adjacent vertex with which it forms a feasible cycle, it is returned as singleton, see lines 9 to 13. Otherwise, we continue with such a shortest feasible cycle consisting of two vertices. In the following, we repeatedly determine two vertices v_a and v_b , which are adjacent in C_k , and a vertex $v_c \in V \setminus V_k$ such that the insertion of v_c between v_a and v_b would yield a minimum increase in the total cycle length. If the augmented cycle length does not exceed T_x , we insert v_c and continue to search for more suitable vertex triples, see lines 18 to 24. Otherwise, we remove V_k from G and continue with the construction of the next cycle. The algorithm terminates when the graph is empty and each vertex is contained in a cycle.

Algorithm 4 Basic CI for LCCP

```

1:  $k \leftarrow 0$ 
2: while  $V \neq \emptyset$  do
3:    $k \leftarrow k + 1$ 
4:    $C_k \leftarrow \text{CREATE FEASIBLE CYCLE}(V, E, K, L)$ 
5:    $G \leftarrow G - V_k$ 
6: return  $C_1, \dots, C_k$ 
7:
8: function  $\text{CREATE FEASIBLE CYCLE}(V, E, K, L)$ :
9:    $v_x \leftarrow \operatorname{argmin}_{v_i \in V} T_i$ 
10:   $V_x \leftarrow \{v_y \in N(v_x) \mid T_x \leq 2 \cdot L_{xy} \text{ and } T_y \leq 2 \cdot L_{xy}\}$ 
11:  if  $V_x = \emptyset$  then
12:     $C_k \leftarrow (\{v_x\}, \emptyset)$ 
13:    return  $C_k$ 
14:
15:   $v_y \leftarrow \operatorname{argmin}_{v_i \in V_x} L_{xi}$ 
16:   $C_k \leftarrow (\{v_x, v_y\}, (\{v_x, v_y\}, \{v_y, v_x\}))$ 
17:   $\tau_k \leftarrow 2 \cdot L_{xy}$ 
18:  while  $V \setminus C_k \neq \emptyset$  do
19:     $(v_a, v_b, v_c) \leftarrow$  vertices  $v_a$  and  $v_b$  being adjacent in  $C_k$  and  $v_c \in V \setminus V_k$ 
20:      minimizing  $\Delta\tau := -L_{ab} + L_{ac} + L_{cb}$ 
21:    if  $\tau_k + \Delta\tau > T_x$  then
22:      return  $C_k$ 
23:     $C_k \leftarrow C_k$  with  $v_c$  inserted between  $v_a$  and  $v_b$ 
24:     $\tau_k \leftarrow \tau_k + \Delta\tau$ 
25:
26:  return  $C_k$ 

```

The basic version can be refined in line 22. Instead of returning C_k directly, any exact or heuristic algorithm for TSP can be applied to the subgraph of G induced by V_k . If a tour of smaller length is found, i.e., τ_k can be decreased, further vertices may possibly be inserted. In that case, the algorithm would restart the while-loop in line 18.

Furthermore, the following observation motivates the introduction of an additional postprocessing routine: When Algorithm 4 terminates, cycle C_m , which was created last, often has small length and contains only a small number of vertices having large critical weights. Thus, we try to extend C_m using vertices from $L := \bigcup_{i=1}^{m-1} V_k$, i.e., vertices contained in the other cycles. If no extension is possible, we return the current solution. Otherwise, we extend C_m in a similar manner as in lines 18 to 24, but in contrast, we have to dynamically adjust the minimum critical weight value of the cycle after each insertion and to check whether or not $\tau_k + \Delta\tau \leq T_c$ when determining a suitable vertex v_c . After extending C_m , we rerun the whole Algorithm 4 on $G - V_m$. The described procedure is repeated until $L = \emptyset$.

Although the next lemma follows from the fact that LCCP is not in APX, we give a concrete proof that demonstrates the possible shortsightedness of CI.

Lemma 9. *Cheapest Insertion for LCCP has no constant approximation ratio.*

Proof. Consider the metric LCCP instance on the complete graph $G = (V, E)$ with $|V| = n = 2k^2$ vertices. For each $v_i \in \{v_1, \dots, v_k\} =: V_1$ let $T_i = 2k$ and for each $v_i \in V \setminus V_1 =: V_2$ let $T_i = 2k^2 - k$. Additionally, let $L_{ij} = 2$ for each edge in $e_{ij} \in V_1 \times V_1$ and let $L_{ij} = 1$ otherwise. An optimal solution consists of two cycles: One cycle containing all nodes in V_1 and the other cycle containing all nodes in V_2 . However, the heuristic produces a solution with k cycles, each featuring one node from V_1 and $2k - 1$ nodes from V_2 . Hence, Cheapest Insertion for LCCP does not admit a constant approximation ratio. \square

8 Mixed Integer Programming Models for LCCP

In this section we present two mixed integer programming models (MIPs) for LCCP which are inspired by two formulations for TSP. To define them, we consider the induced directed graph $G = (V, A)$, whose arc set features two directed arcs $a_{ij}, a_{ji} \in A$ for each edge $e_{ij} \in E$ in the following. Both arcs are assigned the same weight as the corresponding edge. Before discussing the differences between the models, we first describe variables and constraints they have in common.

For each potential proper feasible cycle C_k we introduce a binary variable u_k with $k \in \{1, \dots, \lfloor \frac{n}{2} \rfloor\} := \mathcal{K}$ indicating whether it contains any vertices or not. Additionally, there is a nonnegative continuous variable τ_k representing its length. Next, for each vertex $v_i \in V$ we introduce a binary variable y_i indicating whether the vertex forms a singleton or not. Further, for each vertex $v_i \in V$ and each potential proper cycle C_k we introduce a binary variable z_i^k indicating whether $v_i \in V_k$ or not and analogously for each arc $a_{ij} \in A$ a binary variable x_{ij}^k indicating whether $a_{ij} \in E_k$ or not.

$$\min \sum_{v_i \in V} y_i + \sum_{k \in \mathcal{K}} u_k \quad (1)$$

$$\text{s.t.} \quad y_i + \sum_{k \in \mathcal{K}} z_i^k = 1 \quad \forall v_i \in V \quad (2)$$

$$z_i^k \leq u_k \quad \forall v_i \in V, \forall k \in \mathcal{K} \quad (3)$$

$$\sum_{a_{ij} \in N^+(v_i)} x_{ij}^k = z_i^k \quad \forall v_i \in V, \forall k \in \mathcal{K} \quad (4)$$

$$\sum_{a_{ji} \in N^-(v_i)} x_{ji}^k = z_i^k \quad \forall v_i \in V, \forall k \in \mathcal{K} \quad (5)$$

$$\sum_{a_{ij} \in A} L_{ij} x_{ij}^k = \tau_k \quad \forall k \in \mathcal{K} \quad (6)$$

$$T_i + (M_k - T_i)(1 - z_i^k) \geq \tau_k \quad \forall v_i \in V, \forall k \in \mathcal{K} \quad (7)$$

$$u_k \in \{0, 1\} \quad \forall k \in \mathcal{K} \quad (8)$$

$$y_i \in \{0, 1\} \quad \forall v_i \in V \quad (9)$$

$$z_i^k \in \{0, 1\} \quad \forall v_i \in V, \forall k \in \mathcal{K} \quad (10)$$

$$x_{ij}^k \in \{0, 1\} \quad \forall a_{ij} \in A, \forall k \in \mathcal{K} \quad (11)$$

$$\tau_k \in \mathbb{R}_{\geq 0} \quad \forall k \in \mathcal{K} \quad (12)$$

The objective function (1) aims at minimizing the total number of cycles, i.e., the sum of singletons and proper cycles. Constraint (2) ensures that each vertex either forms a singleton or is assigned proper cycle. If a vertex v_i is assigned proper cycle C_k , then (3) accounts for that and v_i has to have an outgoing and an ingoing arc, which is ensured by constraints (4) and (5), respectively. Next, constraints (6) keep track of the cycle lengths, while (7) ensures that the critical weights values of all vertices are respected, where M_k denotes the k -th biggest critical weight among all vertices.

The formulation above guarantees that each vertex is contained in exactly one cycle and that all critical weight constraints are fulfilled. Nevertheless, as it often occurs when designing MIP formulations for problems related to TSP, we have to take care of possible subtours. Hence, we extend the basic formulation in two different ways: One uses a modified version of subtour elimination constraints and the other one follows the idea of Miller, Tucker and Zemlin of assigning an order to the vertices in order to prohibit subtours, as introduced in [14].

8.1 Subtour Elemination Constraints

Subtour elimination constraints for TSP ensure that between any two nonempty sets of vertices there are at least two arcs connecting them. However, in contrast to TSP, we do not know in advance which vertices form a common cycle in LCCP. Thus, we cannot directly apply the classic subtour elimination constraints and therefore introduce constraints

$$\sum_{\substack{v_i, v_j \in S_1: \\ a_{ij} \in A}} x_{ij}^k + \sum_{\substack{v_i, v_j \in S_2: \\ a_{ij} \in A}} x_{ij}^k \leq |S_1| + |S_2| - 2 \quad \forall S_1, S_2 \subset V, S_1, S_2 \neq \emptyset \quad (13)$$

$$S_1 \cap S_2 = \emptyset, \forall k \in \mathcal{K}$$

instead. Assume that a subset of the vertices assigned to the set representing V_k in the basic MIP formulation above form two proper cycles C_k^1 and C_k^2 . In that case, constraint (13) with $S_1 = V_k^1$ and $S_2 = V_k^2$ is violated, since

$$\sum_{\substack{v_i, v_j \in C_k^1: \\ a_{ij} \in A}} x_{ij}^k + \sum_{\substack{v_i, v_j \in C_k^2: \\ a_{ij} \in A}} x_{ij}^k = |V_k^1| + |V_k^2| = |S_1| + |S_2|.$$

Conversely, if the vertices form one proper cycle, no sets S_1 and S_2 exist such that the corresponding constraint is violated. Hence, the MIP consisting of (1) - (12) and constraints (13), which we call SEC in the following, models LCCP.

8.2 Miller-Tucker-Zemlin Formulation

A second way to avoid subtours in TSP is the Miller-Tucker-Zemlin (MTZ) formulation [16]. Each vertex is assigned a positive weight while the starting vertex has value zero. For each pair of consecutive vertices in a tour the weights must increase except for the last and the starting vertex. Again, a straightforward use for LCCP is not possible, since we cannot fix starting vertices for the cycles in advance. Thus, for each $k \in \mathcal{K}$ and each vertex $v_i \in V$ we introduce additional binary variables $s_i^k \in \{0, 1\}$ indicating whether v_i is the starting vertex of cycle C_k or not. Weight variables $w_i^k \in \mathbb{Z}_{\geq 0}$ together with constraints

$$\sum_{i \in V} s_i^k = u_k \quad \forall k \in \mathcal{K} \quad (14)$$

$$s_i^k \leq z_i^k \quad \forall v_i \in V, \forall k \in \mathcal{K} \quad (15)$$

$$\sum_{v_i \in V} z_i^k - u_k \geq w_i^k \quad \forall v_i \in V, \forall k \in \mathcal{K} \quad (16)$$

$$w_i^k - w_j^k + |V| \cdot (x_{ij}^k - s_j^k) \leq |V| - 1 \quad \forall a_{ij} \in A, \forall k \in \mathcal{K} \quad (17)$$

$$s_i^k \in \{0, 1\} \quad \forall v_i \in V, \forall k \in \mathcal{K} \quad (18)$$

$$w_i^k \in \mathbb{Z}_{\geq 0} \quad \forall v_i \in V, \forall k \in \mathcal{K}. \quad (19)$$

do then model the idea explained above applied to LCCP. Constraints (14) determine a starting vertex for each cycle, which also has to be part of it due to constraint (15). Furthermore, the necessary weight values are bounded by (16) for each set $k \in \mathcal{K}$. Finally, constraints (17) are the Miller-Tucker-Zemlin constraints as explained above. Thus, the MIP model consisting of (1) - (12) together with (14) - (19) models LCCP and we denote it by MTZ in the following.

8.3 Symmetry Breaking Inequalities

The solution space of the two MIP models can be highly symmetric. Given any feasible solution, all permutations of the proper cycle indices respecting constraints (7) are feasible. Assume w.l.o.g. that the vertices are ordered non-increasingly by their critical weights. Then, inequalities

$$z_i^k \leq \sum_{j=1}^{i-1} z_j^{k-1} \quad \forall v_i \in V, \forall k \in \mathcal{K} \setminus \{1\} \quad (20)$$

ensure that only the permutation with the cycles sorted by the minimum index of the contained vertices remains feasible.

8.4 Conflict Clique Inequalities

Let (V, E, T, L) be an LCCP instance and let H_c be its conflict c -hypergraph. Let $U \subseteq V$ with $|U| = m$ be a hyperclique in H_c . By Corollary 2 we derive that

$$\sum_{v_i \in U} z_i^k \leq c - 1 \quad \forall k \in \mathcal{K} \quad (21)$$

are valid inequalities for LCCP. In addition, the size of each clique induces a lower bound on the size of an optimal solution by Corollary 3, i.e.,

$$\sum_{v_i \in V} y_i + \sum_{k \in \mathcal{K}} u_k \geq \lceil \frac{m}{c-1} \rceil \quad (22)$$

is a valid inequality, too. The following lemma demonstrates, that none of the conflict hypergraphs is redundant w.r.t. conflict clique constraints.

Lemma 10. *Let $c \geq 2$. There exists an LCCP instance (V, E, T, L) and a feasible solution for the LP-relaxation of the corresponding MIP models, which is cut of by conflict clique constraints that can only be derived from the conflict c -hypergraph. Further, the lower bound induced by the size of a maximum hyperclique of this c -hypergraph is the only tight one, i.e., it is the only one equal to the size of an optimal solution.*

Proof. Consider the generic LCCP instance consisting of the complete graph on $2c-1$ vertices where $T_i = 2c-1$ for each vertex $v_i \in V$ and $L_{ij} = 2$ for each edge $e_{ij} \in E$. Obviously, for each subset $S \subseteq V$ with $|S| < c$ there exist a feasible cycle, e.g. the cycle induced by any permutation of the vertices of S and the corresponding edges. Thus, all conflict d -hypergraphs with $d < c$ are empty, i.e., do not contain any hyperedge. On the other hand, if $|S| \geq c$ there does not exist any feasible cycle and all conflict d -hypergraphs with $d \geq c$ are complete.

A feasible solution for the LP-relaxations of the MIP models is given by $u_k = \frac{1}{2}$, $z_j^k = \frac{1}{2}$, $x_{ij}^k = \frac{1}{2}$ for all e_{ij} with $j = i + 1$ together with $e_{2c-1,1}$, and $\tau_k = 2c - 1$ for $k \in \{1, 2\}$, while all remaining variables are zero. For model

MTZ we additionally have $s_1^1 = s_1^2 = \frac{1}{2}$. Since the conflict d -hypergraphs with $d \geq c$ are complete, V is the maximum hyperclique in each of them. Therefore, we have that the corresponding conflict clique constraints (21) for $k \in \{1, 2\}$

$$\sum_{v_i \in V} z_i^k = \frac{1}{2}|V| = c - \frac{1}{2} \leq d - 1,$$

are violated if and only if $d = c$. Furthermore, the lower bound (22) on the size of a smallest cycle partition is given by

$$\lceil \frac{2c-1}{c-1} \rceil \geq \frac{2c-1}{c-1} > \frac{2c-2}{c-1} = 2$$

for the conflict c -hypergraph, while for $d > c$ we have

$$\lceil \frac{2c-1}{d-1} \rceil \leq \lceil \frac{2c-1}{c} \rceil = \lceil 2 - \frac{1}{c} \rceil = 2.$$

Since an optimal cycle partition consists of three cycles, e.g., any cycle induced by the vertex sets $V_1 := \{v_1, \dots, v_{c-1}\}$ and $V_2 := \{v_c, \dots, v_{2c-2}\}$ as well as the singleton $V_3 := \{v_{2c-1}\}$, we know that the bound for $d = c$ is tight. \square

9 Computational experiments

In this section, we describe and present the computational experiments, which we conducted in order to test both MIP approaches, and analyze the results.

9.1 Computational Setup

We ran our experiments on a cluster of machines composed of Intel Xeon Gold 5122 @ 3.60GHz CPUs with 96GB of RAM. All algorithms were implemented in Python and we used the corresponding interface of the Branch-and-Bound based solver Gurobi v9.0 [11] with a time limit of 6 hours.

9.2 Instances

For our computational experiments, we generated two sets of test instances based on the 28 instances from the TSPLIB and 14 instances from the ATSP LIB all having 100 or less vertices [19,18]. The ATSP LIB instances are defined on a directed graph and the corresponding arc weights are given as an asymmetric matrix. For our LCCP instances, we chose the weights in the upper triangular part of these matrices.

Furthermore, let τ^* denote the length of an optimal tour for the corresponding TSP or ATSP instance. As mentioned, we created two sets of LCCP test instances: For the first test set we assigned each vertex a random integer in the interval $[\frac{\tau^*}{6}, \frac{\tau^*}{2}]$, while for the second test set we assigned an integer in the interval $[\frac{\tau^*}{8}, \frac{\tau^*}{4}]$ as critical weight.

9.3 Algorithmic Setup

First of all, we ran Algorithm 2 and removed all edges, which cannot be contained in feasible cycles. Next, we computed a feasible solution by running the Cheapest Insertion Heuristic and thereby derived a tighter bound on the necessary size of \mathcal{K} . Here, we used the mentioned refinement step applying an exact algorithm for TSP and the described post-processing routine. The TSP is solved with the MIP formulation from Dantzig, Fulkerson and Johnson [4]. Afterwards, we determined the conflict 2-hypergraph as in Algorithm 3 and calculated all maximal cliques using the algorithm of Cazals and Karande [2]. The corresponding constraints (21) as well as the lower bound (22) and the symmetry breaking constraints (20) were added to both MIP models before the start of the solution process. Furthermore, we determined a subset $\tilde{E}_3^H \subseteq E_3^H$ of the hyperedges of H_3 as follows: For each vertex triple $v_i, v_j, v_l \in V$ we checked whether the sum of the shortest paths between the three nodes is larger than $\min\{T_i, T_j, T_l\}$. Additionally, we added all the edges according to Remark 2. All mentioned calculations together were performed in less than 2 minutes for each instance.

While the corresponding variables and constraints for the MTZ model were added before the start of the solving process, the subtour elimination constraints (13) for the SEC model were separated during the solving process. Let $\mathcal{C} = \{C_1, \dots, C_m\}$ be a solution for the current fomulation. If a cycle C_k contains subtours, we add the corresponding SEC constraint (13) for every pair of distinct subtours and each $k \in \mathcal{K}$.

Additionally, we heuristically separated constraints from cliques in our subset of hyperedges of H_3 for both models during the solving process. After solving the LP relaxation of each Branch-and-Bound node, we determined hypercliques in $\tilde{H}_3 := (V, \tilde{E}_3^H)$ using a greedy routine: Let \tilde{z} denote the vector of z -variable values in the current LP-solution and consider the subgraph induced by the vertex set $\{v_i \in V \mid \tilde{z}_i^k > 0\}$ for some $k \in \mathcal{K}$. By using a greedy algorithm on the hyperedges w.r.t. \tilde{z} we first of all compute a maximal clique in this subgraph. Afterwards, we extend this clique to a maximal clique U in \tilde{H}_3 by adding suitable vertices for which $\tilde{z}_i^k = 0$. The corresponding constraint (21) is added to the model and cuts off the current LP-solution if $\sum_{v_i \in U} \tilde{z}_i^k > 2$.

9.4 Results

The results of our computational experiments can be found in Tables 1 to 4 in the Appendix. Besides the instance name, the number of removed edges due to preprocessing is shown for each instance. Furthermore, while the value of the solution found by the heuristic is given in column UB (CI heur), the lower bound from the size of a maximum clique in the H_2 is written in column LB (H_2 -clique). Finally, for both MIP approaches the upper and the lower bound at the end of the solving process are shown, which was either reached when the problem was solved or when the time limit was reached, which is indicated by TL. Note that the number of vertices of an instance is encapsulated in the instance name. In

the following remarks, we consider instances with less than 30 vertices as small, less than 60 vertices as medium-sized, and all others as large.

The results for the TSPLIB-based test sets can be found in Table 1 for critical weights in $[\frac{\tau^*}{8}, \frac{\tau^*}{4}]$ and in Table 2 for critical weights in $[\frac{\tau^*}{6}, \frac{\tau^*}{2}]$, respectively. For the first test set, we are able to solve all but two medium-sized instances. Thereby, model SEC is able to solve three more instances than model MTZ and is faster on all medium-sized instances. For the second test set, the results are similar: SEC is able to solve one more instance than MTZ and is faster on all instances. However, compared to the first test set, it takes longer to solve the medium sized instances as well as four of them were not solved. This is probably due to the larger critical weights, which allow for more degree of freedom when it comes to the creation of feasible cycles.

Compared to the results in [14], where only model MTZ was tested, we could improve the running times of this algorithmic solution approach. This is mainly due to the new version of the CI heuristic, which generates better incumbents on many instances, and because of the additional cutting planes from H_3 . Further, we are able to prove optimality for 6 additional instances on the first and 4 additional instances on the second test set and could improve the lower bounds for most of the other instances as well.

The results for the ATSP LIB-based test sets can be found in Table 3 for critical weights in $[\frac{\tau^*}{8}, \frac{\tau^*}{4}]$ and in Table 4 for critical weights in $[\frac{\tau^*}{6}, \frac{\tau^*}{2}]$, respectively. The results we see in Table 3 are similar to the ones for the corresponding metric test instance set from Table 1. We are able to solve instances with up to 44 vertices using model SEC, while model MTZ could prove optimality only for the three smallest instances. However, for critical weights in $[\frac{\tau^*}{6}, \frac{\tau^*}{2}]$ model SEC was able to solve only two instances and model MTZ three instances, see Table 4. Again, due to the bigger critical weights, there is more degree of freedom when it comes to creating feasible cycles. Nevertheless, this combined with general edge weights seems to make the problem a lot harder to solve.

10 Conclusion and Outlook

In this article we introduced LCCP for arbitrary nonnegative edge weights generalizing the results presented in [14]. Further, we extended and improved the proposed CI heuristic and developed a new MIP formulation based on a variant of subtour elimination constraints. Additionally, we introduced the concept of conflict hypergraphs for LCCP and showed their close relation to Vertex Coloring. We are able to solve test instances for metric LCCP of small and medium size and improve upon the results shown in [14]. The same holds for general LCCP instances, although they seem to be more difficult to solve.

Acknowledgements

The work of Kai Hoppmann, Gioni Mexi, and Thorsten Koch has been conducted in the Research Campus MODAL funded by the German Federal Ministry of Education and Research (BMBF) (fund number 05M14ZAM). Carl Johan Casselgren was supported by a grant from the Swedish Research Council (2017-05077).

References

1. Asghar, A.B., Smith, S.L., Sundaram, S.: Multi-Robot Routing for Persistent Monitoring with Latency Constraints. arXiv preprint arXiv:1903.06105 (2019)
2. Cazals, F., Karande, C.: A note on the problem of reporting maximal cliques. *Theoretical Computer Science* **407**(1-3), 564–568 (2008)
3. Chamoso, P., Raveane, W., Parra, V., González, A.: UAVs applied to the counting and monitoring of animals. In: *Ambient Intelligence - Software and Applications*, pp. 71–80. Springer (2014)
4. Dantzig, G., Fulkerson, R., Johnson, S.: Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America* **2**(4), 393–410 (1954)
5. Dantzig, G.B., Ramser, J.H.: The Truck Dispatching Problem. *Management Science* **6**(1), 80–91 (1959)
6. Desroches, M., Lenstra, J., Savelbergh, M., Soumis, F.: Vehicle Routing with Time Windows: Optimization and Approximation. *Vehicle routing: Methods and Studies*, B. L. Golden and A. A. Assad (eds.). North-Holland, Amsterdam, pp. 65–84 (1988)
7. Drucker, N., Penn, M., Strichman, O.: Cyclic routing of unmanned air vehicles. *Information Systems Engineering Technical Reports*. IE/IS-2014-02 (2014)
8. Drucker, N., Ho, H.M., Ouaknine, J., Penn, M., Strichman, O.: Cyclic-routing of Unmanned Aerial Vehicles. *Journal of Computer and System Sciences* **103**, 18–45 (2019)
9. Drucker, N., Penn, M., Strichman, O.: Cyclic routing of unmanned aerial vehicles. In: *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. pp. 125–141. Springer (2016)
10. Fredman, M.L., Tarjan, R.E.: Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. *J. ACM* **34**(3), 596–615 (Jul 1987). <https://doi.org/10.1145/28869.28874>, <http://doi.acm.org/10.1145/28869.28874>
11. Gurobi Optimization, L.: Gurobi Optimizer Reference Manual, Version 9.0.0. <http://www.gurobi.com> (2019)
12. Hausamann, D., Zirnig, W., Schreier, G.: Monitoring of gas transmission pipelines - A customer driven civil UAV application. In: *ODAS Conference* (2003)
13. Ho, H.M., Ouaknine, J.: The cyclic-routing UAV problem is PSPACE-complete. In: *International Conference on Foundations of Software Science and Computation Structures*. pp. 328–342. Springer (2015)
14. Hoppmann, K., Mexi, G., Burdakov, O., Casselgren, C.J., Koch, T.: Minimum Cycle Partition with Length Requirements. In: *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Springer (2020)
15. Liebchen, C., Möhring, R.H.: The Modeling Power of the Periodic Event Scheduling Problem: Railway Timetables - and Beyond. In: *Algorithmic Methods for Railway Optimization*, pp. 3–40. Springer (2007)

16. Miller, C.E., Tucker, A.W., Zemlin, R.A.: Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)* **7**(4), 326–329 (1960)
17. Ollero, A., Martínez de Dios, J.R., Merino, L.: Unmanned aerial vehicles as tools for forest-fire fighting. *Forest Ecology and Management* **234**(1), S263 (2006)
18. Reinelt, G.: TSPLIB - A Traveling Salesman Problem Library. *ORSA Journal on Computing* **3**(4), 267–384 (1991)
19. Reinelt, G.: TSPLIB and ATSP LIB instances (accessed July 27, 2020), <http://comopt.ifl.uni-heidelberg.de/software/TSPLIB95/>
20. Rosenkrantz, D.J., Stearns, R.E., Lewis, II, P.M.: An Analysis of Several Heuristics for the Traveling Salesman Problem. *SIAM Journal on Computing* **6**(3), 563–581 (1977)
21. Sahni, S., Gonzalez, T.: P-complete approximation problems. *Journal of the ACM (JACM)* **23**(3), 555–565 (1976)
22. Serafini, P., Ukovich, W.: A Mathematical Model for Periodic Scheduling Problems. *SIAM Journal on Discrete Mathematics* **2**(4), 550–581 (1989)
23. Solomon, M.M., Desrosiers, J.: Survey Paper - Time Window Constrained Routing and Scheduling Problems. *Transportation science* **22**(1), 1–13 (1988)
24. Suurballe, J.W., Tarjan, R.E.: A quick method for finding shortest pairs of disjoint paths. *Networks* **14**(2), 325–336 (1984)
25. Yu, W., Liu, Z., Bao, X.: New approximation algorithms for the minimum cycle cover problem. *Theoretical Computer Science* **793**, 44–58 (2019)

Appendix

Table 1: Results for TSPLIB with critical weights from $[\frac{\tau^*}{8}, \frac{\tau^*}{4}]$

Instance name	Removed edges	UB (CI heur)	LB (H_2 -clique)	MTZ			SEC		
				UB	LB	time (sec)	UB	LB	time (sec)
burma14	67	6	6	6	6	1	6	6	1
ulysses16	65	6	5	6	6	1	6	6	1
gr17	96	8	7	8	8	1	8	8	1
gr21	141	8	6	8	8	4	8	8	3
ulysses22	127	7	6	7	7	4	7	7	2
gr24	172	7	7	7	7	1	7	7	1
fri26	188	8	6	8	8	6	8	8	10
bayg29	238	10	7	8	8	42	8	8	23
bays29	235	8	6	8	8	36	8	8	21
dantzig42	478	9	7	9	9	1552	9	9	575
swiss42	451	9	7	9	8	TL	9	9	284
att48	605	9	6	8	8	4563	8	8	2893
gr48	536	9	5	9	8	TL	9	8	TL
hk48	548	10	6	9	8	TL	9	9	14417
eil51	522	9	5	9	8	TL	9	9	19242
berlin52	515	9	6	9	9	2956	9	9	2260
brazil58	649	9	5	8	7	TL	9	7	TL
st70	1074	10	5	10	7	TL	10	7	TL
eil76	780	10	5	10	6	TL	10	6	TL
pr76	925	10	6	10	7	TL	10	7	TL
gr96	1587	10	5	10	6	TL	10	7	TL
rat99	1673	10	5	10	6	TL	10	6	TL
kroA100	2141	10	5	10	6	TL	10	5	TL
kroB100	1979	10	5	10	6	TL	10	6	TL
kroC100	2183	11	6	11	7	TL	11	6	TL
kroD100	1962	11	6	11	6	TL	11	6	TL
kroE100	2092	11	5	11	6	TL	11	6	TL
rd100	1817	11	5	11	6	TL	11	6	TL

Table 2: Results for TSPLIB with critical weights from $[\frac{\tau^*}{6}, \frac{\tau^*}{2}]$

Instance name	Removed edges	UB (CI hour)	LB (H_2 -clique)	MTZ			SEC		
				UB	LB	time (sec)	UB	LB	time (sec)
burma14	40	5	4	5	5	1	5	5	1
ulysses16	39	4	4	4	4	1	4	4	1
gr17	61	5	4	5	5	1	5	5	1
gr21	86	6	3	5	5	2	5	5	1
ulysses22	66	5	4	5	5	4	5	5	1
gr24	53	6	4	5	5	11	5	5	3
fri26	105	6	5	6	6	87	6	6	9
bayg29	103	6	4	5	5	30	5	5	13
bays29	111	6	5	6	6	32	6	6	14
dantzig42	243	7	4	6	6	20686	6	6	15081
swiss42	160	7	4	6	6	13111	6	6	6833
att48	299	8	5	6	5	TL	6	6	4939
gr48	197	7	4	7	5	TL	7	6	TL
hk48	282	7	4	6	6	18843	6	6	3656
eil51	128	7	3	7	5	TL	7	5	TL
berlin52	264	7	4	7	6	TL	7	6	TL
brazil58	227	6	3	6	4	TL	6	5	TL
st70	218	7	3	7	5	TL	7	5	TL
eil76	185	7	3	7	5	TL	7	5	TL
pr76	208	7	3	7	5	TL	7	5	TL
gr96	406	7	3	7	5	TL	7	5	TL
rat99	587	7	3	7	5	TL	7	5	TL
kroA100	803	8	3	8	5	TL	8	5	TL
kroB100	643	8	3	8	5	TL	8	5	TL
kroC100	856	7	3	7	5	TL	7	5	TL
kroD100	635	7	3	7	5	TL	7	5	TL
kroE100	678	7	3	7	5	TL	7	5	TL
rd100	446	8	4	8	4	TL	8	5	TL

Table 3: Results ATSP LIB with critical weights from $[\frac{\tau^*}{8}, \frac{\tau^*}{4}]$

Instance name	Removed edges	UB (CI hour)	LB (H_2 -clique)	MTZ			SEC		
				UB	LB	time (sec)	UB	LB	time (sec)
br17	114	6	6	6	6	1	6	6	1
ftv33	284	9	7	8	8	120	8	8	137
ftv35	275	9	6	8	8	205	8	8	149
ftv38	332	10	6	9	8	TL	9	9	2614
p43	226	4	2	4	3	TL	3	3	1398
ftv44	441	9	6	9	8	TL	9	9	3297
ftv47	468	11	5	11	8	TL	11	8	TL
ry48p	459	9	5	9	8	TL	9	8	TL
ft53	177	11	3	11	6	TL	11	6	TL
ftv55	563	10	5	10	7	TL	10	7	TL
ftv64	580	10	5	10	7	TL	10	7	TL
ft70	0	9	1	9	6	TL	9	5	TL
ftv70	565	11	5	11	6	TL	11	7	TL
kro124p	744	10	3	10	5	TL	10	5	TL

Table 4: Results ATSP LIB with critical weights from $[\frac{\tau^*}{6}, \frac{\tau^*}{2}]$

Instance name	Removed edges	UB (CI hour)	LB (H_2 -clique)	MTZ			SEC		
				UB	LB	time (sec)	UB	LB	time (sec)
br17	92	5	5	5	5	1	5	5	1
ftv33	97	7	4	7	6	TL	7	6	TL
ftv35	67	6	3	5	5	14852	6	5	TL
ftv38	70	7	3	7	5	TL	6	5	TL
p43	196	3	2	3	3	1	3	3	1
ftv44	112	8	3	8	5	TL	8	6	TL
ftv47	169	7	3	7	5	TL	7	6	TL
ry48p	162	7	3	7	5	TL	7	5	TL
ft53	12	7	2	7	4	TL	7	4	TL
ftv55	145	8	4	8	5	TL	8	5	TL
ftv64	130	8	3	8	5	TL	8	5	TL
ft70	0	7	1	7	4	TL	7	4	TL
ftv70	72	7	2	7	4	TL	7	5	TL
kro124p	64	7	2	7	4	TL	7	4	TL