European Journal of Operational Research 316 (2024) 473–487

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/eor

Discrete Optimization

The Hamiltonian p-median problem: Polyhedral results and branch-and-cut algorithms

Michele Barbato [a,*], Luís Gouveia [b]

[a] Dipartimento di Informatica "Giovanni Degli Antoni", Università degli Studi di Milano, via Celoria 18, 20133, Milan, Italy
[b] Centro de Matemática, Aplicações Fundamentais e Investigação Operacional (CMAF-CIO), Faculdade de Ciências da Universidade de Lisboa, C6 - Piso 4 Lisboa, 1749-016, Portugal

ARTICLE INFO

Keywords:
Combinatorial optimization
Hamiltonian p-median problem
Valid inequality
Polyhedral study
Branch-and-cut algorithm

ABSTRACT

In this paper we study the Hamiltonian p-median problem, in which we are given an edge-weighted graph and we are asked to determine p vertex-disjoint cycles spanning all vertices of the graph and having minimum total weight. We introduce two new families of valid inequalities for a formulation of the problem in the space of edge variables. Each one of the families forbids solutions to the 2-factor relaxation of the problem that have less than p cycles. The inequalities in one of the families are associated with large cycles of the underlying graph and generalize known inequalities associated with Hamiltonian cycles. The other family involves inequalities for the case with $p = \lfloor n/3 \rfloor$, associated with edge cuts and multi-cuts whose shores have specific cardinalities. We identify inequalities from both families that define facets of the polytope associated with the problem. We design branch-and-cut algorithms based on these families of inequalities and on inequalities associated with 2-opt moves removing sub-optimal solutions. Computational experiments on benchmark instances show that the proposed algorithms exhibit a comparable performance with respect to existing exact methods from the literature. Moreover the algorithms solve to optimality new instances with up to 400 vertices.

1. Introduction

Given a positive integer p and an edge-weighted complete graph G over n vertices, the *Hamiltonian p-median problem* (abbreviated to HpMP) is to find a minimum-weight set of p vertex-disjoint cycles (in this paper a *cycle* is a 2-regular connected subgraph of G) whose vertex-sets partition the vertices of G. The HpMP has been first introduced by Branco and Coelho in the context of location-routing problems with real-world applications covering, among others, "school location, milk stations and depot location for different industrial and commercial purposes" (Branco & Coelho, 1990). A generic application is illustrated by the assignment of p guards to n objects where it is assumed that each guard cycles among the assigned protection objects. Similarly, the assignment of p maintenance/inspection vehicles to maintain/inspect n machines can also be modelled as the HpMP. Besides its applications to the above location-routing problems, other applications of the HpMP and its variations have subsequently been found in cutting problems (Glaab, 2002) and laser multi-scanners (Glaab & Pott, 2000).

The HpMP is related to several other combinatorial problems, among which we mention those relevant for our discussion. Firstly, as already noted in all previous studies on the problem, the HpMP for $p = 1$ corresponds to the well-known *Symmetric Travelling Salesman Problem* (abbreviated to STSP), showing that the HpMP is **NP**-hard if p is part of the input. Similarly, when $n = 3p$ the HpMP corresponds to the *triangle packing problem* which is also known to be **NP**-Hard (Garey & Johnson, 1979). Instead, if p is not fixed the HpMP reduces to the *2-factor problem* (*i.e.,* the problem of determining a minimum weight 2-regular subgraph of a graph G) which is solvable in polynomial time (Cornuéjols & Pulleyblank, 1980).

We point out that the HpMP has appeared in two variants in the literature. In one variant, usually studied in works with formulations using binary edge variables, 2-cycles are not allowed. This may lead to sub-optimal solutions with respect to the other variant, allowing 2-cycle solutions and considered in Branco and Coelho (1990) where the HpMP was initially introduced. The variant in which 2-cycles are allowed is usually modelled with directed models, see *e.g.*, Bektaş, Gouveia, and Santos (2018), although models using undirected edge variables with values in $\{0, 1, 2\}$ might also be derived, following an approach similar to the one described in Benavent and Martínez (2013) for a multi-depot routing problem.

* Corresponding author.
E-mail addresses: michele.barbato@unimi.it (M. Barbato), legouveia@fc.ul.pt (L. Gouveia).

https://doi.org/10.1016/j.ejor.2024.02.032
Received 24 December 2022; Accepted 29 February 2024
Available online 1 March 2024

In this paper we focus on the variant without 2-cycles. We conclude the introduction by illustrating the motivations and contributions of this work and by providing the outline of the paper.

Motivations and contributions of the paper. For several routing and location-routing problems, the fastest resolution methods are those developed from formulations using a minimal set of variables, as, for instance, in the case of the STSP, see, *e.g.,* Gutin and Punnen (2006). Clearly, the performance of such methods strongly depends on the strength of the underlying formulations which, in many cases, involve sets of constraints having exponential size and thus require specialized methods to make them useful from a practical point of view.

Besides the degree constraints for each vertex, the formulations for the HpMP can be viewed as containing two sets of inequalities, one set preventing *more than* p cycles and the other preventing *less than* p cycles. Only a few works have focused on the study of formulations defined in the space of the binary edge variables and on methods devised for such formulations (see, *e.g.,* Gollowitzer, Gouveia, Laporte, Pereira, & Wojciechowski, 2014 and Hupp & Liers, 2013). Moreover, previous studies have reported that the constraints forbidding more than p cycles have a positive impact on the performance of algorithms for the HpMP, also when other sets of variables are used, see, *e.g.*, Bektaş et al. (2018). Conversely, inequalities forbidding less than p cycles and having strong theoretical properties or substantial computational impact have been less investigated. In practice, this phenomenon reflects computationally: in literature it is observed that the HpMP instances with large values of p are harder to solve than those with small values p. Motivated by these observations, this study will focus on inequalities preventing less than p cycles, in the edge-variable space.

More precisely, we first introduce and discuss the new family of "quasi-Hamiltonian" cycle inequalities which generalize inequalities known from Gollowitzer et al. (2014) and Hupp and Liers (2013) and that are associated with "Hamiltonian" cycles. Next, we provide additional families of inequalities, designed for the cases with large values of p (which correspond to several unsolved instances from the literature) and that are related with the cut constraints used in integer-linear programming (ILP) models for the STSP. We will provide conditions under which some of these families induce facets of the HpMP polytope, whose extreme points correspond to the HpMP solutions. The new sets of inequalities are used to design effective branch-and-cut algorithms based on ILP formulations involving only edge variables.

Outline. In Section 2 we review the literature on the HpMP. In Section 3 we provide an ILP formulation for the HpMP and report related results from the literature. In Section 4 we introduce the quasi-Hamiltonian cycle inequalities and also provide sufficient conditions for a subset of the new inequalities to induce facets of the HpMP polytope. In Section 5 we discuss the special cases with $p = \lfloor n/3 \rfloor$. For these cases, we present new valid inequalities associated with cuts and multi-cuts whose shores have specific cardinalities, identify inequalities that are facet-inducing for the HpMP polytope and provide new formulations for the HpMP in which $p = \lfloor n/3 \rfloor$. In Section 6 we describe algorithms to separate all these strengthening inequalities. In Section 7 we describe branch-and-cut algorithms based on the previous results, we compare them to state-of-the-art exact methods for the HpMP and test them on new larger HpMP instances; finally we assess the effectiveness of the proposed inequalities.

2. Literature review

In order to put in context our contributions, and since in this paper we will consider an exact approach for the HpMP, in this section we review recent existing studies on the HpMP referring the reader to the references in the reviewed papers for older contributions. In particular, we do not detail works on heuristic or approximate algorithms for the HpMP. Also, and unless otherwise stated, the works reviewed below are for the variant forbidding 2-cycles.

Formulations and algorithms based on edge variables. Hupp and Liers (2013) focus on an edge-variable formulation for the HpMP and highlight which of its inequalities cannot induce facets for the underlying integer hull. The authors also propose a new family of valid inequalities enforcing at least p cycles in each solution. These inequalities, associated with cycles, are generalized to inequalities associated to cycle covers in Gollowitzer et al. (2014). Additionally, in this latter paper, constraints based on vertex partitions and enforcing at most p cycles in a solution are presented. Finally, the resulting formulation, solely based on edge variables, is tested in a branch-and-cut framework. The above mentioned inequalities based on cycles and partitions will be recalled in Section 3 of our paper.

Formulations and algorithms based on node–depot assignments. Location-routing formulations involving variables associating one depot to each cycle besides the usual edge variables are proposed in Gollowitzer et al. (2014) and one branch-and-cut method based on such a formulation is proposed; Erdoğan, Laporte, and Chía (2016) propose a branch-and-cut algorithm that is based on a formulation enhancing the inequalities given in the previous paper. The underlying formulation produces lower bounds of very good quality, and the resulting branch-and-cut algorithm outperforms the branch-and-cut algorithms presented in Gollowitzer et al. (2014).

Set-partitioning formulations. A set-partitioning formulation was first introduced in Branco and Coelho (1990) for the directed version of the problem. This formulation has been subsequently adapted by Gollowitzer, Pereira, and Wojciechowski (2011) for the variant in which 2-cycles are not allowed. The authors theoretically show that this latter formulation always provides a stronger lower bound than those produced by the formulations based on edge variables and node–depot assignment variables; however the formulation is not computationally tested. Marzouk, Moreno-Centeno, and Üster (2016) test a branch-and-price algorithm based on the set-partitioning formulation given in the earlier work of Gollowitzer et al. (2014). The branch-and-price algorithm considers as a valid column of the model any simple cycle of the graph, relaxing the requirement of its cost minimality (originally present in the set-partitioning formulations of Branco & Coelho, 1990; Gollowitzer et al., 2014). The resulting algorithm solves instances with up to 318 vertices and outperforms the edge-variable formulation given in Gollowitzer et al. (2014) on HpMP instances with values of p larger than $p^\star$, the number of cycles in the 2-factor relaxation of the problem. However it is outperformed by the edge-variable formulation when $p < p^\star$.

A directed formulation. Bektaş et al. (2018) present a branch-and-cut algorithm based on a directed model and study the two variants of the problem, allowing and not allowing 2-cycles. The main idea of the model presented in this work is that it splits the arc variables x_{ij} into three sets depending on whether node i is the depot of a cycle, node j is the depot of a cycle, or none is. This "split" model allows the use of a set of multi-cut inequalities used in Bektaş, Gouveia, and Santos (2017) for a multi-depot routing problem. The branch-and-cut algorithm based on that model solves benchmark instances with up to 171 vertices when 2-cycles are allowed and up to 100 vertices when 2-cycles are forbidden. In the latter case, the algorithm proves the optimality of solutions to benchmark instances previously unsolved and compares well with those proposed in Erdoğan et al. (2016) and Marzouk et al. (2016).

3. Starting formulation and related known results

In this section we review an ILP formulation for the HpMP introduced in Hupp and Liers (2013). Most of notation and terminology used in this paper is standard in combinatorial optimization, hence here we just introduce definitions that are more relevant for the description of the ILP formulation. Let $G = (V, E)$ be a simple undirected complete graph with edge weights $c : E \to \mathbb{R}_+$. For every subgraph S of G we

denote by $V(S)$ and $E(S)$ the vertex- and edge-set of S and we define the *cost* of S as $c(S) := \sum_{e\in E(S)} c_e$; the same notation is extended to subsets of edges and to vectors other than c indexed by the edges of G. A *cycle* of G is a connected 2-regular subgraph of G. The *cardinality* or *length* of a cycle C can be equivalently defined as its vertex- or edge-cardinality, hence we write $|C|$ to refer to either of these quantities. A cycle of length k is also called *k-cycle*. A subgraph of $G = (V, E)$ is *spanning* if its vertex set coincides with V. A 2-regular spanning subgraph of G is composed of vertex-disjoint cycles and it is called a *2-factor* of G. A spanning cycle of G is also called a *Hamiltonian* cycle. Given a partition $S_1, S_2, \dots, S_m$ of V, we define the *multi-cut* $\delta(S_1, S_2, \dots, S_m) = \{\{i, j\} \in E : i \in S_k, j \in S_\ell \text{ with } k \neq \ell\}$. A multi-cut involving only two sets is simply called a *cut*. For the sake of conciseness we also define $\delta(i) := \delta(\{i\}, V \setminus \{i\})$.

The *incidence vector* of a subgraph S of G is a point $\mathbf{x} \in \{0,1\}^{|E|}$ such that $\mathbf{x}_e = 1$ if and only if e belongs to S. From now on, given a positive integer p, a *solution* to the HpMP will indicate both a set $C_1, C_2, \dots, C_p$ of p vertex-disjoint cycles of G spanning all vertices in V and the corresponding incidence vector. We denote by $\mathcal{X}_p^n$ the *HpMP polytope*, that is, the convex hull of all solutions to the HpMP. Following Gollowitzer et al. (2014) and Hupp and Liers (2013), we model the HpMP as the following ILP involving only *edge variables* defined by constraints (4):

$$\min \sum_{e\in E} c_e x_e$$

$$x(\delta(v)) = 2 \quad \forall v \in V \tag{1}$$

$$x(\delta(S_1, S_2 \dots, S_{p+u})) \geq u + 1$$

$$\forall S_1, \dots, S_{p+u} \text{ partition of } V,\ u \in \left\{1, \dots, \left\lfloor \frac{n-3p}{3} \right\rfloor\right\} \tag{2}$$

$$x(H) \leq n - p \qquad \forall H \text{ Hamiltonian cycle of } G \tag{3}$$

$$x_e \in \{0,1\} \qquad \forall e \in E. \tag{4}$$

The solutions to (1)–(4) are precisely the solutions to the HpMP. Indeed, it is well-known that a point $\mathbf{x} \in \{0,1\}^{|E|}$ satisfies the *degree constraints* (1) if and only if it is the incidence vector of a 2-factor. Moreover, the two families of inequalities (2) and (3) constrain the number of cycles in such 2-factor to be p. More precisely, the *partition inequalities* (2) generalize the well-known cut constraints for the STSP (corresponding to the case $p = u = 1$) and forbid 2-factors with more than p cycles; instead, the *Hamiltonian cycle inequalities* (3) forbid those having less than p cycles by imposing that at least p edges must be removed from any Hamiltonian cycle.

The strength of inequalities of an ILP formulation is related to the dimension of the corresponding faces in the integer hull of the formulation. In particular, the *facets*, that is, the proper faces of highest dimension, are typically in correspondence with strong, and computationally effective, inequalities. According to Hupp and Liers (2013), a result known from Glaab (2000) states that the inequalities (2) for $u = 1$ are facet-defining for $\mathcal{X}_p^n$ when $p \geq 1$ and $n \geq 3p$. In contrast, for generic pairs of n and p it is unknown whether family (3) contains inequalities inducing facets of $\mathcal{X}_p^n$, and it can be shown that no Hamiltonian cycle inequality is facet-defining when $n = 3p$ (a similar result is given in Hupp and Liers (2013), where it is shown that the Hamiltonian cycle inequalities cannot induce facet when $p = 2$).

A first approach to overcome this limitation of the Hamiltonian cycle inequalities is proposed in Gollowitzer et al. (2014) and consists in generalizing the idea underlying the Hamiltonian cycle inequalities by considering support graphs made of 2-factors with less than p cycles (and not only 1 cycle). This leads to the *cycle cover* inequalities associated to any set of vertex-disjoint cycles $C_1, C_2, \dots, C_{p-k}$ with $k \geq 1$ covering V:

$$x(C_1) + x(C_2) + \cdots + x(C_{p-k}) \leq n - k - 1. \tag{5}$$

Inequalities (5) have been utilized in the branch-and-cut algorithm of Gollowitzer et al. (2014) using a heuristic separation (in the same paper it is shown that the separation problem for (5) with $k = 1$ is **NP**-hard). In this paper we follow a different approach, namely we consider inequalities associated to single cycles not necessarily covering all vertices of G. This is the subject of next Section 4.

4. Quasi-Hamiltonian cycle inequalities

In this section we introduce and study the family of *quasi-Hamiltonian cycle* (QHC) inequalities, defined by:

$$x(C) \leq |C| - p + \left\lfloor \frac{n - |C|}{3} \right\rfloor$$

$$\forall C \text{ cycle of } G \text{ such that } n - 3p + 4 \leq |C| \leq n \tag{6}$$

The family of Hamiltonian cycle inequalities (3) coincides with the subfamily of (6) in which C is Hamiltonian because, in that case, $|C| = n$ hence $\left\lfloor \frac{n-|C|}{3} \right\rfloor = 0$. However, (6) also includes inequalities associated with non-spanning cycles of G. Given such a cycle C, the condition $|C| \geq n - 3p + 4$ imposed in (6) guarantees that at most $p - 2$ cycles can be formed using the vertices not in C. This latter aspect is exploited in the proof of the following proposition:

Proposition 1. *Inequalities (6) are valid for $\mathcal{X}_p^n$ for all $p \geq 2$.*

Proof. Let $\mathbf{x}$ be a vertex of $\mathcal{X}_p^n$ and let us assume, by contradiction, that $\mathbf{x}$ violates an inequality (6) associated with a cycle C. Then, since $\mathbf{x}$ is a binary vector and the coefficients in the left-hand side of (6) are identically 1, we have $\mathbf{x}(C) \geq |C| - p + \left\lfloor \frac{n-|C|}{3} \right\rfloor + 1$. Let K be the subgraph of G induced by the edges $e \in E(C)$ such that $\mathbf{x}_e = 1$, so that $\mathbf{x}(C) = |E(K)|$. Then by our hypothesis, we have $|E(K)| \geq |C| - p + \left\lfloor \frac{n-|C|}{3} \right\rfloor + 1$, hence we get that $|C \setminus K| = |C| - |E(K)| \leq p - \left\lfloor \frac{n-|C|}{3} \right\rfloor - 1$. The previous inequality implies that K is obtained from C by removing at most $p - \left\lfloor \frac{n-|C|}{3} \right\rfloor - 1$ edges, therefore, since C is a cycle, K contains at most $p - \left\lfloor \frac{n-|C|}{3} \right\rfloor - 1$ connected components. Hence at most $p - \left\lfloor \frac{n-|C|}{3} \right\rfloor - 1$ cycles containing $V(C)$ can be obtained by using the edges $e \in E$ such that $\mathbf{x}_e = 1$. Let V' be the set of vertices not contained in any of those cycles. Since $|V'| \leq n - |C|$, the vertices in V' can be covered by at most $\left\lfloor \frac{n-|C|}{3} \right\rfloor$ cycles of G. Finally, the total number of cycles in the subgraph K' having $\mathbf{x}$ as incidence vector is the sum of the number of cycles covering $V(K)$ and the number of cycles covering V'. The number of cycles covering $V(K)$ is at most $p - \left\lfloor \frac{n-|C|}{3} \right\rfloor - 1$; the number of cycles covering V' is at most $\left\lfloor \frac{n-|C|}{3} \right\rfloor$. However K' must have p cycles and this is a contradiction. □

We now illustrate several properties of the family of QHC inequalities. First, in the following proposition we provide a link between the QHC and the cycle cover inequalities (the proof is omitted for the sake of brevity):

Proposition 2. *For every $p \geq 3$ and $n \geq 3p$ the QHC inequality (6) defined by a cycle C such that $n - 3p + 4 \leq |C| \leq n - 3$ implies all inequalities (5) defined by the vertex-disjoint cycles $C, C_1, C_2, \dots, C_h$ covering V, where $h = \left\lfloor \frac{n-|C|}{3} \right\rfloor$.*

Observe that the cycle cover inequalities implied by the QHC inequalities, as stated in Proposition 2, are a strict subset of family (5). Moreover, for given C and $h = \left\lfloor \frac{n-|C|}{3} \right\rfloor$ as in Proposition 2 the QHC inequality defined by C only implies a weaker form of the cycle cover inequality associated with less than $h + 1$ cycles. For example, if $p = 4$ and $n-8 \leq |C| \leq n-6$ summing the QHC inequality $x(C) \leq |C| - p + 2$ with $x(C') \leq |C'|$ (with C' covering all vertices not in C) yields $x(C) + x(C') \leq n - p + 2$ which is weaker than (5) associated with cycles C and C'.

Another property of the QHC inequalities is that they strictly tighten the linear programming relaxation of formulation (1)–(4). This result is formally stated and proved in Proposition 10 of Appendix A. In brief,

the proof exhibits fractional points belonging to the linear relaxation of (1)–(4) which violate QHC inequalities associated with cycles of length $n-3p+4$. This analysis made us to investigate the strength of the QHC inequalities associated to cycles of such size, leading to the following proposition, proved in Appendix B:

Proposition 3. *Let $p \geq 3$ and $n \geq 3p+3$. If C is a cycle of odd length $|C| = n-3p+4$, then inequality (6) associated with C is facet-defining for $\mathcal{X}_p^n$.*

We point out that the result of Proposition 3 only provides sufficient conditions for a subfamily of QHC inequalities to be facet-defining for $\mathcal{X}_p^n$ and by no means it excludes that *other* QHC inequalities can have the same property, at least for some specific values of n and p.

5. The HpMP for large values of p

We now focus on the HpMP for large values of p, namely those such that $p = \lfloor n/3 \rfloor$. Several works on the HpMP have reported that these are challenging cases from a computational point of view, see, *e.g.*, Bektaş et al. (2018) and Erdoğan et al. (2016). In this section we provide new sets of inequalities valid for these cases. They are based on the specific structure of the feasible solutions to the HpMP with $p = \lfloor n/3 \rfloor$ and lead to substantially better computational results than those reported in the literature. Moreover, exploiting such inequalities and using them together with particular cases of the QHC inequalities (6) we provide alternative formulations for the instances of the HpMP with $p = \lfloor n/3 \rfloor$.

5.1. Restricted cut and multi-cut constraints

We begin by considering the family of *restricted cut constraints* (RCCs) defined by:

$$x(\delta(S)) \geq 2 \qquad S \subseteq V \text{ such that } |S| = 3k+2 \text{ for some } 1 \leq k \leq n/3. \quad (7)$$

Their name derives from the observation that, except for the cardinality restriction imposed on S, the inequalities in (7) coincides with the cut constraints well-known from STSP formulations.

Proposition 4. *Inequalities (7) are valid for $\mathcal{X}_p^{3p}$ and $\mathcal{X}_p^{3p+1}$ for every $p \geq 2$.*

Proof. Let $S \subseteq V$ be a vertex subset verifying the cardinality condition in (7). If $n = 3p$, every solution is composed uniquely of cycles containing exactly 3 edges. Hence since $|S|$ is not a multiple of 3, the restriction of any solution to the graph induced by S cannot be 2-regular. Then, by the degree constraints (1), at least two edges of the solution belong to the cut $\delta(S)$, that is, $x(\delta(S)) \geq 2$ is valid for $\mathcal{X}_p^{3p}$. If $n = 3p+1$, a similar reasoning applies, by observing that all solutions in this case contain exactly one cycle with 4 edges and the remaining cycles with 3. □

The RCCs (7) prevent less than p cycles, and, when $p \geq 3$, it is not difficult to give examples of integer points not in $\mathcal{X}_p^{3p}$ or $\mathcal{X}_p^{3p+1}$ and which violate one of these constraints.

We now observe that there is no advantage in relaxing the cardinality requirement on $|S|$ in (7). That is, the resulting constraints would be either implied or invalid for $\mathcal{X}_p^{3p}$ and $\mathcal{X}_p^{3p+1}$. To see this, consider the following three cases. If $|S| \leq 2$ then $x(\delta(S)) \geq 2$ is trivially implied by the degree constraints (1). If $|S| = 3k+1$ for some $k \geq 1$ and $n = 3p+1$ then there exists a HpMP solution $\mathbf{x}$ such that $\mathbf{x}(S) = 0$, while if $n = 3p$, then S', the complement of S, is such that $|S'| = 3k'+2$ for some $0 \leq k' \leq n/3$ and $x(\delta(S)) \geq 2$ is equivalent to $x(\delta(S')) \geq 2$. If $|S| = 3k$ then there exists a solution $\mathbf{x}$ to the HpMP with $n = 3p$ (resp. $n = 3p+1$) such that $\mathbf{x}(S) = 0$.

It is well-known that the cut constraints are facet-defining for the STSP polytope. We were able to adapt a proof of this latter result provided in Conforti, Cornuéjols, and Zambelli (2014, p. 300–301) to the RCCs (7) in the case $n = 3p+1$, thus obtaining the following result, whose proof is given in Appendix C.

Proposition 5. *The RCCs (7) are facet-defining for $\mathcal{X}_p^{3p+1}$ if $|S| \geq 5$ and $p \geq 3$.*

A similar result for the case $n = 3p$ is still open, since the proof of Proposition 5 is not easily adapted to that case (see Appendix C for more details).

We point out that the assumption $|S| \geq 5$ in Proposition 5 is not restrictive: indeed, when $n = 3p+1$, the RCCs associated to subsets $S = \{u, v, w, t\}$ of cardinality 4 are implied by the QHC inequalities associated with the three cycles $C_1 = (u, v, w, t, u)$, $C_2 = (u, w, v, t, u)$ and $C_3 = (u, v, t, w, u)$ because summing those inequalities member-wise yields $2x(S) \leq 6$ which is easily seen to be equivalent to $x(\delta(S)) \geq 2$ through the degree constraints (1).

We conclude by observing that when $n = 3p+2$ we cannot re-adapt the RCCs defined before since each feasible solution admits an empty cut for every possible cardinality of the shores. For reasons of space we omit elaborating on this statement. However, an extension of the results in this section to the case $n = 3p+2$ can be done by using more general multi-cut inequalities, as we explain next. To state the general form of these inequalities we first observe that the cuts defining the RCCs (7) can be viewed as multi-cuts involving only two subsets, namely S (with $|S_1| \equiv 2 \pmod 3$) and its complement. The inequalities for $n = 3p+2$ generalize the RCCs to multi-cuts between three subsets, two of which have a restricted cardinality.

That is, we focus on the following family of *restricted 3-cut constraints* (R3CCs), whose validity is proved in Appendix D:

$$x(\delta(S_1, S_2, S_3)) \geq 2$$
$$\forall \text{ partition } S_1, S_2, S_3 \text{ of } V \text{ s.t. } |S_1| \equiv 2 \text{ and } |S_2| \equiv 2 \pmod 3. \quad (8)$$

Proposition 6. *The R3CCs (8) are valid for $\mathcal{X}_p^{3p+2}$ for every $p \geq 2$.*

In the experiments performed in Section 7 there is no instance with $n = 3p+2$ and thus, we have not developed separation algorithms for the inequalities (8).

5.2. Alternative formulations for the cases $p = \lfloor n/3 \rfloor$

One might ask whether the RCCs (7) (when $n = 3p, 3p+1$) or the R3CCs (8) (when $n = 3p+2$) together with the degree constraints (1) and the integrality constraints (4) suffice to give a valid ILP model for the HpMP with $p = \lfloor n/3 \rfloor$. Unfortunately the answer is negative.

We first exhibit a counter-example in the case $n = 3p$. Consider an instance with $p = 3$ and $n = 9$, and a 2-factor of G, composed of one 3-cycle C_3 and one 6-cycle C_6. Its incidence vector satisfies all constraints (7), although it is not feasible for the considered instance because, when $n = 3p$, each feasible solution is composed uniquely of 3-cycles. The latter property also shows that the smallest cycles yielding valid QHC inequalities have 4 edges, hence the corresponding inequalities (6) are $x(C) \leq 2$ for every 4-cycle C of G. Observe that any integer solution satisfying the degree inequalities (1) with less than p cycles would have at least one cycle, C', with more than 3 edges. But such a solution would violate 4-cycle inequalities (6) associated to any 4-cycle D containing 3 consecutive edges from C'. This argument leads to

Proposition 7. *A valid ILP formulation for the HpMP with $n = 3p$ is given by the degree constraints (1), the QHC inequalities (6) for 4-cycles and the integrality constraints (4).*

We consider, now, the case $n = 3p+1$. The following counter-example shows that the RCCs (7) are not sufficient to describe the HpMP solutions: consider an instance with $p = 3$ and $n = 10$; then, the incidence vector of a 2-factor of G made of a 4-cycle C_4 and a 6-cycle C_6, satisfies constraints (7), although it is not feasible for the considered instance. As noted in Section 5.1, when $n = 3p+1$, each feasible solution is composed of cycles with 3 edges and a single cycle with 4 edges.

Moreover, the smallest cycles associated with QHC inequalities (6) have 5 edges, hence the corresponding inequalities are $x(C) \leq 3$ for every 5-cycle C of G. Consider an instance with $n = 16$ and $p = 5$ and a 2-factor composed of 4 vertex-disjoint cycles, each with 4 vertices. Its incidence vector clearly satisfies the degree constraints (1) and the QHC inequalities associated with the smallest cycles, but it is infeasible for the HpMP since it contains less than 5 cycles (and more than one cycle with 4 vertices). Thus, the analogue of Proposition 7 for the case of $n = 3p + 1$ is not valid. However, as the next proposition shows, we obtain a valid formulation if we add the RCCs (7) of the previous Section 5.1.

Proposition 8. *A valid ILP formulation for the HpMP with $n = 3p + 1$ is given by the degree constraints (1), the QHC inequalities (6) associated to 5-cycles, the RCCs (7) and the integrality constraints (4).*

Proof. We only need to show that if $\mathbf{x}$ is integer and satisfies (1), (6) and (7) then it is the incidence vector of a solution to the HpMP when $n = 3p + 1$. From (1), (6) and the integrality constraints, $\mathbf{x}$ is the incidence vector of a 2-factor of G composed of 3- and 4-cycles. Since $n = 3p+1$ such subgraph cannot be composed only of 3-cycles and thus, it must contain at least one 4-cycle. Point $\mathbf{x}$ is the incidence vector of a solution to the problem with $n = 3p+1$ if and only if there is exactly one 4-cycle in such subgraph. Assume that there are two distinct 4-cycles C^1 and C^2. Then, by defining $S = V(C^1) \cup V(C^2)$, the RCC (7) $x(\delta(S)) \geq 2$ is valid for $\mathcal{X}_p^{3p+1}$ (because $|S| \neq 3k, 3k + 1$ for all $1 \leq k \leq p$) and would be violated by $\mathbf{x}$, leading to a contradiction. □

Finally, for the case $n = 3p + 2$ we give a counter-example showing that the R3CCs are not sufficient for a valid formulation of the problem: consider an instance with $p = 4$ and $n = 14$; then the infeasible 2-factor of G given by two 4-cycles and one 6-cycle satisfies all inequalities (8). Combining the R3CCs (8) with the QHC inequalities (6) associated to 6-cycles, we get the following result for $n = 3p + 2$, analogous to Proposition 8. Its proof is given in Appendix D.

Proposition 9. *A valid ILP formulation for the HpMP with $n = 3p + 2$ is given by the degree constraints (1), the QHC inequalities (6) associated to 6-cycles, the R3CCs (8) and the integrality constraints (4).*

6. Separation algorithms

The families of QHC inequalities (6) and RCCs (7) presented in the previous sections have exponential size, hence their inclusion in ILP formulations for the HpMP requires separation. In the following we describe the separation algorithms for the inequalities used in the computational experiments of Section 7. They always receive in input a (generally non-integer) point $x^\star \in [0, 1]^{|E|}$ satisfying the degree constraints (1). Associated with $x^\star$ is its *support graph* $G_{x^\star} = (V, E_{x^\star})$ where $E_{x^\star} = \{e \in E : x_e^\star > 0\}$. Unless differently stated, paths in this section have no repeated vertices, that is, we assume a *path* to be obtained from a cycle by removing a single edge; the *length* of paths is defined as the number of their edges.

Separation of the partition inequalities (2). The generic separation problem for inequalities (2) is **NP**-hard, as shown in Gollowitzer et al. (2014) through a reduction from the minimum k-cut problem. Hence for the separation of inequalities (2) we resort to the same algorithm devised in Gollowitzer et al. (2014). Given $x^\star \in [0, 1]^{|E|}$, the algorithm constructs the support graph $G_{x^\star}$. Next it computes the connected components $S_1, S_2, \ldots, S_\ell$ of $G_{x^\star}$. If $\ell \geq p + 1$ then the sets $S_1, S_2, \ldots, S_\ell$ define a violated inequality (2). The construction of graph $G_{x^\star}$ takes $O(|V|^2)$-time, while computing the connected components takes $O(|V| + |E_{x^\star}|)$-time by depth-first search. We point out that for generic fractional points $x^\star \in [0, 1]^{|E|}$ the above algorithm is heuristic, although it is exact when $x^\star \in \{0, 1\}^{|E|}$.

Separation of the QHC inequalities (6). We are not aware of the complexity of separating generic QHC inequalities (6), although we conjecture it is **NP**-hard. We describe a separation algorithm for the inequalities (6), that is, inequalities of the form $x(C) \leq |C| - p + \left\lfloor \frac{n-|C|}{3} \right\rfloor$ where C is a cycle of length at least $n - 3p + 4$. We consider separately the cases $x^\star \in \{0, 1\}^{|E|}$ and $x^\star \notin \{0, 1\}^{|E|}$.

Separation for $x^\star \in \{0, 1\}^{|E|}$. In this case, the integrality of $x^\star$ implies that it is the incidence vector of a 2-factor of G composed by ℓ cycles $C_1, C_2, \ldots, C_\ell$. If $\ell \geq p$ the algorithm terminates since no QHC inequality is violated in this case (it is easy to prove that a QHC inequality is always satisfied by a 2-factor with at least p cycles). Otherwise, we have $\ell < p$, and we first check if each of these ℓ cycles alone leads to a violated QHC inequality. If that is the case, the corresponding violated QHC inequalities are added to the pool of violated cuts and the separation stops; otherwise we merge subsets of the ℓ cycles into a single cycle until a violated QHC inequality is found. The existence of such an inequality can be shown theoretically from the hypothesis $\ell < p$. Therefore this separation algorithm is exact. Furthermore, recall that the family of QHC inequalities (6) contains the Hamiltonian cycle inequalities (3). Hence, the correctness of the above procedure, together with the correctness of the separation of the partition inequalities (2) on integer points, guarantees that branch-and-cut algorithms separating (2) and (6) always yield an optimal solution to the HpMP. The details on cycle merging are given in Appendix E.1.

Separation for $x^\star \notin \{0, 1\}^{|E|}$. We start by determining the connected components $K_1, K_2, \ldots, K_\ell$ of the support graph $G_{x^\star}$. If $\ell \geq p$, the procedure stops, reporting no violated cut. Otherwise, we look for a minimal subset of components whose total vertex number is at least $n - 3p + 4$. Subsequently, a STSP is heuristically solved on the edge-weighted subgraph of G induced by those vertices and having weight $1 - x_e^\star$ on edge $e \in E$. Next, we check whether the resulting cycle C defines a violated QHC inequality. Additionally, if $|C| > n - 3p + 4$, we search for other violated inequalities by modifying the cycle C according to 2 rules: (1) we sequentially remove vertices guaranteeing that the resulting cycles still define a valid QHC inequality which, if violated, is added to the pool of violated inequalities; (2) as soon as a violated QHC inequality cannot be produced by the process in step (1) we construct cycles of length $n - 3p + 4$ from C by considering all its sub-paths of consecutive $n-3p+4$ vertices and linking their endpoints. The rationale behind rule (2) is that, by Proposition 3, the cycles of length $n-3p+4$ are likely to induce strong QHC inequalities. Finally, the above procedure is repeated for larger subsets of vertices obtained by merging additional connected components from the set $K_1, K_2, \ldots, K_\ell$. All details are given in Appendix E.2.

Separation of RCCs (7). We separate the RCCs (7) for HpMP instances with $n = 3p, 3p+1$ as follows. When $x^\star \in \{0, 1\}^{|E|}$ we use the following exact separation algorithm which runs in $O(n^2)$ time. First we retrieve the cycles $C_1, C_2, \ldots, C_\ell$ composing the graph $G_{x^\star}$. We collect all cycles C such that $|C| \equiv 1 \pmod 3$ in a set K_1 and those such that $|C| \equiv 2 \pmod 3$ in a set K_2. In the case $n = 3p$, if $|K_1| = |K_2| = 0$ no RCC is violated by $x^\star$. Otherwise all RCCs $x(\delta(V(C))) \geq 2$ for $C \in K_1 \cup K_2$ and all RCCs $x(\delta(V(C_1 \cup C_2))) \geq 2$ for $C_1, C_2 \in K_1$ are added to the pool of violated cuts. In the case $n = 3p+1$, no RCC is violated by $x^\star$ if $|K_2| = 0$ and $|K_1| = 1$. Otherwise, all RCCs $\delta(V(C)) \geq 2$ for $C \in K_2$ and all RCCs $\delta(V(C_1 \cup C_2)) \geq 2$ for $C_1, C_2 \in K_1$ are added to the pool of violated cuts.

When $x^\star \notin \{0, 1\}^{|E|}$, we assign the weight $x_e^\star$ to edge e of the support graph $G_{x^\star}$. Then a minimum-weight cut $\delta(S)$ in the resulting edge-weighted graph with S satisfying the cardinality condition of Proposition 4 can be found in polynomial-time, using techniques from submodular minimization under congruency constraints, see Nägele, Sudakov, and Zenklusen (2019) and the discussion in Nägele (2021, Sect. 3.1.2). However, in our implementation we resort to the following simpler heuristic algorithm that works well in practice: we run the exact separation algorithm for the STSP cut constraints implemented in CONCORDE (Applegate et al., 2024) and only add those satisfying the cardinality condition of Proposition 4.

Separation of QHC inequalities (6) for the cases $n = 3p$ and $n = 3p + 1$. We describe the approaches used in our branch-and-cut algorithms to separate the QHC inequalities (6) for 4-cycles and 5-cycles, respectively valid for the cases $n = 3p$ and $n = 3p+1$. We first explain the separation algorithms for $n = 3p$. Given $x^\star \in \{0,1\}^{|E|}$, that is, when $x^\star$ is integer, we determine all cycles composing $G_{x^\star}$. Let $C = (v_0, v_1, \dots, v_k, v_0)$ be one such cycle having length at least 4 and for every $i = 0, 1, \dots, k$ consider the cycles $C_i = (v_i, v_{i+1}, v_{i+2}, v_{i+3}, v_i)$, with indices taken modulo $k + 1$: each cycle C_i yields a violated QHC inequality (6) associated to a 4-cycle. All these inequalities are added to the pool of violated cuts. When $x^\star \notin \{0,1\}^{|E|}$, that is, when $x^\star$ is non-integer, we observe that a 4-cycle C_4 yielding a violated QHC inequality necessarily contains two consecutive edges e and f of $G_{x^\star}$ such that $x^\star_e + x^\star_f > 1$ (as otherwise, summing twice all edges of C_4 gives $2x^\star(C_4) \le 4$ which is the QHC associated to C_4 and hence the latter cannot be violated). Therefore for every vertex $v \in V$ we first search for the two distinct vertices $u, w \in V \setminus \{v\}$ maximizing $x^\star_{\{u,v\}} + x^\star_{\{v,w\}}$. If the latter value is greater than 1, vertex v potentially belongs to a 4-cycle yielding a violated QHC inequality. In this case, for every vertex $t \in V \setminus \{u, v, w\}$ we define the two 3-paths $P^1_{v,t} = (t, u, v, w)$ and $P^2_{v,t} = (u, v, w, t)$. If

$$x^\star(P^1_{v,t}) > 2 \text{ or } x^\star(P^2_{v,t}) > 2 \tag{9}$$

we add to the pool of violated cuts the QHC inequality $x(C_{v,t}) \le 2$ where $C_{v,t} = (t, u, v, w, t)$.

The case $n = 3p+1$ is treated by considering 5-cycles and 4-paths in place of the 4-cycles and 3-paths of the above procedure, and by setting to 3 the right-hand side of condition (9). We point out that, while the separation algorithm described above is exact for $x^\star \in \{0,1\}^{|E|}$, it is heuristic for $x^\star \notin \{0,1\}^{|E|}$. Indeed, when $n = 3p$ (resp. $n = 3p+1$) there could be violated QHC inequalities associated with 4-cycles (resp. 5-cycles) even if condition (9) is not satisfied (*e.g.*, taking $x^\star$ of value 0.6 identically on all edges of such cycles). Although this procedure is heuristic, the computational results presented later on indicate that it works well in practice. Furthermore, in preliminary tests, we have considered the inclusion of all violated 4- and 5-cycle inequalities, but that strategy slightly worsened the computational times, hence we maintained the approach described above.

7. Experimental study

The formulations and the inequalities presented in the previous sections are tested in a branch-and-cut framework. We consider three branch-and-cut algorithms:

- algorithm $\mathcal{F}$ based on formulation (1)–(4), the QHC inequalities (6) and the RCCs (7) (when $n = 3p, 3p + 1$);
- algorithm $\mathcal{A}_{3p}$ based on the formulation given in Proposition 7 and valid for the HpMP instances with $n = 3p$. It includes the degree constraints (1), the integrality constraints (4) and the QHC inequalities (6) for 4-cycles;
- algorithm $\mathcal{A}_{3p+1}$ based on the formulation given in Proposition 8 and valid for the HpMP instances with $n = 3p + 1$. It includes the degree constraints (1), the integrality constraints (4) the QHC inequalities (6) for 5-cycles and the RCCs (7).

Except for the degree constraints (1), all inequalities are added dynamically through the separation routines described in Section 6 (when $n = 3p, 3p+1$ the QHC inequalities for 4- and 5-cycles are separated with the specialized routine). Moreover, the algorithms are further enhanced by a set of so-called "restricted 2-opt inequalities" described later.

Each branch-and-cut algorithm above is first compared with those of Bektaş et al. (2018) and Erdoğan et al. (2016) using the benchmark instances used in the literature and subsequently tested on new harder instances introduced in this paper. Finally, we analyse the impact of the proposed inequalities on the performance of the algorithms.

Before presenting the results, we describe the instances used in tests, the restricted 2-opt inequalities (together with the strategy for their separation), a primal heuristic used in the algorithms and some additional implementation details.

Instances. We test the algorithms on a total of 230 HpMP instances whose cost matrices are taken from benchmark STSP instances. For every STSP instance on n vertices, we consider 5 HpMP instances with $p = \left\lfloor \frac{n}{10} \right\rfloor, \left\lfloor \frac{n}{7} \right\rfloor, \left\lfloor \frac{n}{5} \right\rfloor, \left\lfloor \frac{n}{4} \right\rfloor, \left\lfloor \frac{n}{3} \right\rfloor$. We consider three sets of instances (the number of vertices in an instance corresponds to the numerical value at the end of the instance name):

- *literature HpMP instances*: these are obtained from 22 TSPLIB instances (Reinelt, 1991, 1995) with $21 \le n \le 100$. They are used in Bektaş et al. (2018) and Erdoğan et al. (2016) to evaluate the corresponding algorithms, with which the algorithms proposed in this paper are compared;
- *new TSPLIB-based instances*: these are obtained from 9 TSPLIB instances having $42 \le n \le 400$, namely `dantzig42`, `pr124`, `u159`, `brg180`, `rat195`, `d198`, `pr226`, `gil262` and `rd400`;
- *tetrahedral HpMP instances*: these are obtained from 15 "tetrahedron" instances with $67 \le n \le 199$, namely `Tnm58`, `Tnm67`, `Tnm79`, `Tnm88`, `Tnm97`, `Tnm109`, `Tnm118`, `Tnm127`, `Tnm139`, `Tnm148`, `Tnm157`, `Tnm169`, `Tnm178`, `Tnm187` and `Tnm199`. The tetrahedron instances are known to be very difficult Euclidean TSP instances recently introduced in Hougardy and Zhong (2021).

Restricted 2-opt inequalities. In order to speed up the execution of the proposed branch-and-cut algorithms we use a specialized version of the so-called *local search inequalities*, first presented in Lancia, Rinaldi, and Serafini (2015) in the context of the STSP. For the sake of brevity we focus on the version used in this paper, and we refer the reader to Lancia et al. (2015) for the general definition of local search inequalities. More precisely, we consider the following *restricted 2-opt inequalities*:

$$x_{ij} + x_{jk} + x_{k\ell} \le 2 \qquad \forall i, j, k, \ell \in V \text{ s.t. } c_{ij} + c_{jk} + c_{k\ell} > c_{ik} + c_{jk} + c_{j\ell} \tag{10}$$

A HpMP solution violating an inequality of type (10) is necessarily suboptimal, because it contains a cycle where vertices i, j, k and ℓ appear consecutively in this order. Due to the condition on the edge-costs, such a solution can be strictly improved by applying a 2-opt exchange on those vertices. Therefore, although inequalities (10) cut off HpMP solutions, they can be safely employed in algorithms solving the HpMP to optimality. Since the number of inequalities (10) is of the order of $O(n^4)$, we resort to separation to include them during the execution of the algorithms. Preliminary experiments highlighted that enumerating all vertices i, j, k, ℓ indexing (10) is excessively time consuming. Therefore we use the following heuristic procedure. Consider a threshold $\tau \in]0, 1[$, a solution $x^\star$ and all connected components $K_1, K_2, \dots, K_m$ of the graph $G_{x^\star}(\tau)$ obtained from $G_{x^\star}$ by removing all edges e such that $x^\star_e < \tau$. Next, iteratively for $h = 1, 2, \dots, m$, we run the enumerative algorithm only on the vertices of K_h. For each possible configuration of four vertices $i, j, k, \ell \in K_h$ the corresponding inequality (10) is checked for violation and, in the affirmative case, it is subsequently added to the model. In the computational results presented below we choose $\tau = 0.7$ and we separate (10) only on non-integer points. The chosen value for τ reduces the density of the graph where paths of 4 vertices are detected. Moreover it guarantees that every path of 4 vertices of $G_{x^\star}(0.7)$ verifying the edge-cost condition in (10) violates the corresponding inequality, because its right-hand side is equal to 2.

Primal heuristic. We adapt a primal heuristic presented in Gollowitzer et al. (2014). Namely, given a fractional solution $x^\star$ we assign to each edge e of graph $G_{x^\star}$ the weight $x^\star_e$. Then we first compute a minimum-weight spanning tree, next we remove the $p-1$ edges of largest weight thus obtaining p connected components; finally a STSP heuristic is run on each component to determine p cycles. These latter are refined by executing the 2-opt exchanges on the sequences of 4 consecutive vertices, as described in the previous paragraph. The whole process terminates in polynomial time. A detailed description of the heuristic is provided in Appendix E.3.

Other implementation details. The parts of the branch-and-cut algorithms that do not directly include CONCORDE are implemented in C++. The algorithms on graphs have been implemented using the COIN-OR library LEMON (Dezső, Jüttner, & Kovács, 2011). We compiled the code with `g++ 7.4.0` on a Ubuntu 18.04 machine, equipped with a Intel(R) Core(TM) i7-6700K CPU (4.00 GHz) and 32 gigabyte of RAM. The search trees generated by the branch-and-cut algorithms are managed via IBM CPLEX 12.6.3 (IBM, 2016). The primal heuristic described above is run at every node of the enumeration tree. The restricted 2-opt inequalities (10) are added *globally*, that is, once separated they are present in the LPs of all branch-and-cut nodes; all other families of inequalities are added *locally*, that is, they are present only in sub-trees rooted at the branch-and-cut node where they have been separated.

We also use the standard setting of CPLEX for heuristic procedures and for the addition of general-purpose cuts. In all tests we use the "strong branching" strategy of CPLEX (CPLEX parameter `VarSel` set to 3) and the "best bound" tradeoff of CPLEX for the MIP emphasis (CPLEX parameter `MIPEmphasis` set to 3). When not differently specified, all other CPLEX parameters are set to their default values. In particular, due to the presence of CPLEX callbacks for the implementation of cut separation, the branch-and-cut algorithms used in the experiments are run in sequential mode on a single thread. Moreover, when not differently stated, CPLEX is allowed to use its generic-purpose cuts, in their default setting. The same holds for the branch-and-cut algorithms of Bektaş et al. (2018) and Erdoğan et al. (2016) (personal communication). In CPLEX it is not possible to obtain the complete set of generic-purpose cuts that have been separated in an experiment, hence we provide the full list of available cuts in CPLEX in Appendix F.

We point out that we did not run the literature algorithms we compare with, hence their performance is taken from the tables of results given in the respective papers. In particular, the algorithm of Bektaş et al. (2018) was run with a CPU time limit of 3 h (while the algorithms introduced in this paper and the algorithm of Erdoğan et al. (2016) have a CPU time limit of 1 h). More importantly, the results of Bektaş et al. (2018) and Erdoğan et al. (2016) were obtained on machines slower than the one used in our experiments, namely on a machine equipped with an Intel Core i7-4790 3.6 GHz processor and 8 gigabyte of RAM in the former case, and on a Lenovo T440p laptop with an i7 2.50 GHz CPU and 8 gigabyte RAM in the latter case. We used the website https://www.cpubenchmark.net/ to evaluate the three CPUs. The comparison reported that the single thread rating of the CPU of the machine used to run algorithms $\mathcal{F}$, $\mathcal{A}_{3p}$ and $\mathcal{A}_{3p+1}$ is 11% higher than that of Bektaş et al. (2018) and 25% higher than that of Erdoğan et al. (2016).[1]

Online repository. In this paper we present only the main experimental results and, in several cases, we focus on qualitative analyses. The tables with all detailed results are provided for reproducibility purposes at the online repository (Barbato & Gouveia, 2023).

7.1. Results of algorithm $\mathcal{F}$: Comparison with the literature

We compare algorithm $\mathcal{F}$ with the two best branch-and-cut algorithms in the literature, namely those of Bektaş et al. (2018) and Erdoğan et al. (2016). We do not consider in this comparison the branch-and-price algorithm of Marzouk et al. (2016) whose performance is comparable to the branch-and-cut algorithm of Bektaş et al. (2018) (see the discussion therein). When comparing $\mathcal{F}$ with the literature algorithms, we consider both the quality of lower bounds obtained from $\mathcal{F}$ and the performance of $\mathcal{F}$ in reaching integer optimality.

[1] The detailed comparison is available at the URL: https://www.cpubenchmark.net/compare/2226vs2565vs2219/Intel-i7-4790-vs-Intel-i7-6700K-vs-Intel-i7-4710MQ.

Quality of lower bounds. We consider two lower-bounding approaches related to algorithm $\mathcal{F}$: its *LP bound*, denoted $\mathcal{L}(\mathcal{F})$, which is the linear relaxation value of formulation (1)–(4), and its *root node bound*, denoted $\mathcal{R}(\mathcal{F})$, which additionally benefits from CPLEX generic cuts and other features related with the integrality of variables ($\mathcal{R}(\mathcal{F})$ is obtained by stopping CPLEX at the root node, that is, setting its parameter `NodeLim` to 1). In both approaches the inequalities are separated using the routines described in Section 6. Since most of such separation routines are heuristic, the results only approximate the real value of the LP bound and root node bound associated with the formulation and the valid inequalities underlying $\mathcal{F}$.

The instance-wise results used in the lower-bound analysis are provided at the online repository (Barbato & Gouveia, 2023) while here we perform a qualitative analysis. The scatter plots of Fig. 1 compare $\mathcal{L}(\mathcal{F})$ and $\mathcal{R}(\mathcal{F})$ with the lower bounds obtained from the formulations of Bektaş et al. (2018) and Erdoğan et al. (2016). The comparison metric is the relative optimality gap (in percentage) computed as $100(UB^{\star} - LB)/UB^{\star}$, where $UB^{\star}$ is the best upper bound known for that instance. In each scatter plot the plus signs and the diamond markers correspond to the 20 literature HpMP instances with $58 \leq n \leq 76$ while the circle markers correspond to the 35 literature HpMP instances with $n = 99{,}100$. The x-axis of the scatter plots on the left column (resp. right column) reports the relative optimality gap of the lower bounds obtained in Bektaş et al. (2018) (resp. Erdoğan et al., 2016); the y-axis of the scatter plots on the upper row (resp. lower row) reports the relative optimality gap of $\mathcal{R}(\mathcal{F})$ (resp. $\mathcal{L}(\mathcal{F})$). The points below the dotted blue diagonal (coinciding with the orthant bisector) correspond to instances for which $\mathcal{F}$ produces better lower bounds than the literature methods.

From Fig. 1 we immediately see that $\mathcal{L}(\mathcal{F})$ is dominated by the lower bounds obtained in Bektaş et al. (2018) and Erdoğan et al. (2016). Conversely, $\mathcal{R}(\mathcal{F})$ yields better lower bounds than the algorithm of Bektaş et al. (2018) for most of instances and is comparable with the lower-bounding approach of Erdoğan et al. (2016) (although this latter slightly dominates $\mathcal{R}(\mathcal{F})$). Also, we note that larger relative optimality gaps of all four lower-bounding procedures mostly coincide with the larger instances, although this outcome is more evident for $\mathcal{L}(\mathcal{F})$ and the lower bounds of Bektaş et al. (2018). The main conclusion of the above qualitative comparison is the following:

Experimental Observation 1. *The generic cuts generated by CPLEX are crucial to obtain strong lower bounds from formulation $\mathcal{F}$. In particular, $\mathcal{R}(\mathcal{F})$ yields lower bounds comparable with those of Erdoğan et al. (2016) (which are the best ones overall) and less sensitive to the instance size than those of Bektaş et al. (2018).*

We have also made experiments using single families of CPLEX cuts alone in order to see their effect on the lower bounds. The detailed results are reported in Appendix F. In general, there are only two families of CPLEX cuts yielding better lower bounds than $\mathcal{L}(\mathcal{F})$, namely the "cover" cuts and the "mixed-integer rounding" (MIR) cuts of CPLEX. Using the cover cuts alone produces better lower bounds than using the MIR cuts alone. In almost all cases, however, both these families produce worse lower bounds than those of $\mathcal{R}(\mathcal{F})$. Therefore we conclude that the good results of $\mathcal{R}(\mathcal{F})$ are obtained from non-trivial combinations of generic cuts of CPLEX.

Branch-and-cut performance. We now compare algorithm $\mathcal{F}$ and the algorithms of Bektaş et al. (2018) and Erdoğan et al. (2016) in terms of exact resolution of HpMP instances. For the comparison we use the set of 55 literature HpMP instances already considered in Bektaş et al. (2018) and Erdoğan et al. (2016), having $58 \leq n \leq 100$.

The results are reported in Table 1. The first three columns of the table contain the name of the underlying TSPLIB instances, the number $p^{\star}$ of cycles composing their 2-factor relaxation and the number p of cycles required in the corresponding HpMP instances. In the subsequent columns, for each algorithm we report the optimal value and the time

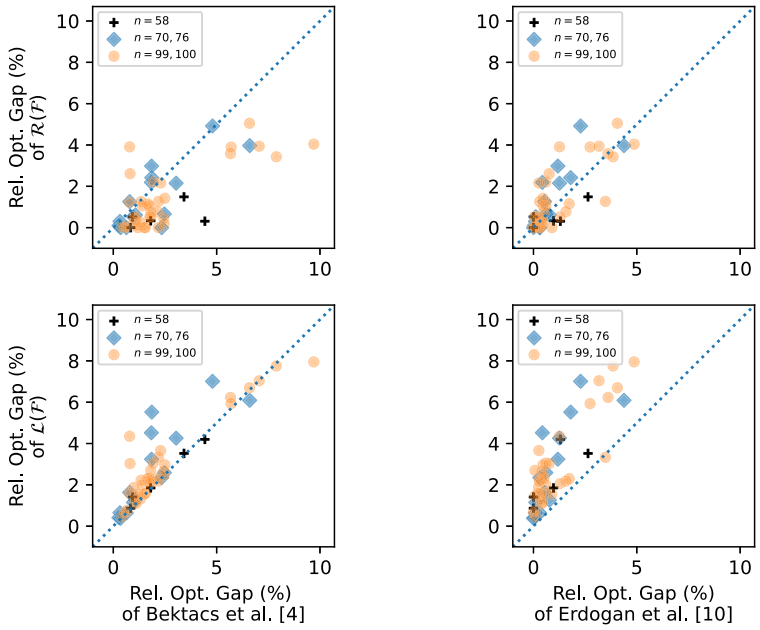


Fig. 1. Scatter plots comparing $\mathcal{L}(\mathcal{F})$ and $\mathcal{R}(\mathcal{F})$ with the lower bounds of the HpMP formulations given in Bektaş et al. (2018) and Erdoğan et al. (2016). In each scatter plot, the points represent instances whose x- and y-values correspond to the relative optimality gaps of the lower bounds specified by the axes labels. Points below the diagonal (dotted blue line) indicate instances where $\mathcal{F}$ produces better lower bounds. The three different markers indicate instance size ranges (plus signs for $n = 58$, diamonds for $n = 70, 76$ and circles for $n = 99{,}100$).

needed to reach optimality (in CPU seconds) or, if the instance is unsolved within the time limit, the pair $[UB, LB]$ of best upper and lower bound along with the relative optimality gap upon termination. We recall that the relative optimality gap is expressed in percentage and computed as $100(UB-LB)/UB$. Since the considered formulations fail to solve to optimality several instances, the last line of Table 1 additionally reports the number of instances solved by each algorithm.

We begin with an outcome which is common to all three considered algorithms:

Experimental Observation 2. *The literature HpMP instances are more difficult for all considered algorithms when $p > p^{\star}$ and especially if $p = \lfloor n/3 \rfloor$.*

We point out that results similar to those of the Experimental Observation 2 were observed already in Bektaş et al. (2018) and Erdoğan et al. (2016).

Next, we analyse the relative performance of the algorithms. To account for the difference between the machine used in this paper and those used in Bektaş et al. (2018) and Erdoğan et al. (2016), for each instance we highlight in boldface the CPU time of algorithm $\mathcal{F}$ if it is at least 35% smaller than the CPU times of both algorithms of Bektaş et al. (2018) and Erdoğan et al. (2016) (time limits are treated as infinite); similarly, we highlight in boldface the CPU time obtained by the algorithms of Bektaş et al. (2018) and Erdoğan et al. (2016) on a given instance if, reducing it by 35%, yields a value smaller than the CPU time of $\mathcal{F}$ on the same instance. Then the results of Table 1 lead to the following:

Experimental Observation 3. *On the literature HpMP instances, algorithm $\mathcal{F}$ performs generally better than the algorithms of Bektaş et al. (2018) and Erdoğan et al. (2016). In particular, $\mathcal{F}$ is faster on the 30 largest instances in this set, having $n = 100$. For every $58 \leq n \leq 100$, $\mathcal{F}$ is especially effective on the literature HpMP instances with $p \in \{\lfloor n/10 \rfloor, \lfloor n/7 \rfloor, \lfloor n/3 \rfloor\}$.*

We explain the last outcome of the Experimental Observation 3 as follows: when $p \in \{\lfloor n/10 \rfloor, \lfloor n/7 \rfloor\}$ the partition inequalities forbidding more than p cycles are more relevant and we have an effective separation routine for such inequalities thus, obtaining a good performance overall; when $p = \lfloor n/3 \rfloor$, algorithm $\mathcal{F}$ benefits from the combination of RCCs and QHC associated with 4- or 5-cycles, which are separated by a specialized routine.

To intuitively illustrate the advantage of $\mathcal{F}$ over the algorithms of Bektaş et al. (2018) and Erdoğan et al. (2016), in Figs. 2(a) and 2(b) we provide qualitative analyses of their performances, based on the results of Table 1.

Fig. 2(a) plots the performance profiles of the three algorithms, indicating how many instances have been solved to optimality (values on the y-axis) within a given CPU time (values on the x-axis, in logarithmic scale). Only instances solved within 3600 CPU seconds (our time limit) are considered. The performance profiles of Fig. 2(a) lead to the following:

Experimental Observation 4. *Algorithm $\mathcal{F}$ solves to optimality the largest amount of HpMP instances within very short time thresholds (around 20 s); the algorithm of Bektaş et al. (2018) and $\mathcal{F}$ solve comparable amounts of instances within time thresholds of at least 100 s and they are always more effective than the algorithm of Erdoğan et al. (2016).*

Fig. 2(b) shows two boxplots, obtained by considering the 42 instances solved to optimality by *all* algorithms. The left boxplot (resp. the right boxplot) describes the distribution of the ratios of the CPU times of $\mathcal{F}$ over the CPU times of the algorithm of Bektaş et al. (2018) (resp. Erdoğan et al., 2016). The points in the figure are the ratio values on single instances. A point with y-value lower than 1 indicates that, on the corresponding instance, $\mathcal{F}$ was faster than the competing algorithm. Except for a few outliers, all quartiles of the two boxplots are below 0.4 and the median is below 0.1 in both cases. That is, we have:

Experimental Observation 5. *In most of the instances the CPU time of $\mathcal{F}$ was less than 1/10 of the CPU times of the algorithms of Bektaş et al. (2018) and Erdoğan et al. (2016). The advantage of $\mathcal{F}$ is more evident when compared to the algorithm of Erdoğan et al. (2016).*

Another remark concerns the impact of CPLEX cuts on the performance of algorithm $\mathcal{F}$. We have tested a variation of algorithm $\mathcal{F}$ not using any of the general-purpose cuts of CPLEX. The detailed results are

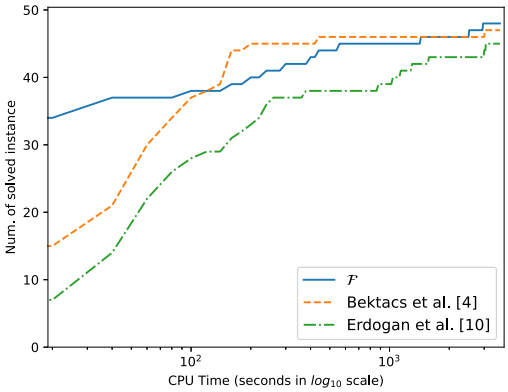
Table 1
Comparison of branch-and-cut algorithm $\mathcal{F}$ with those of Bektaş et al. (2018) and Erdoğan et al. (2016) on the literature HpMP instances with $58 \leq n \leq 100$. For each algorithm we report the optimal value and the time (in CPU seconds) to solve the instances, or the final pair of UB and LB along with the relative optimality gap (in percentage) upon termination.

Instance	$p^\star$	p	$\mathcal{F}$		Bektaş et al. (2018)		Erdoğan et al. (2016)	
			Final Bound	Time (s) and Gap(%)	Final Bound	Time (s) and Gap(%)	Final Bound	Time (s) and Gap(%)
brazil58	12	5	21 744.00	**0.73**	21 744.00	12.00	21 744.00	78.90
		8	21 289.00	**0.14**	21 289.00	6.00	21 289.00	36.95
		11	21 080.00	**0.01**	21 080.00	5.00	21 080.00	5.14
		14	21 221.00	**0.99**	21 221.00	2.00	21 221.00	4.72
		19	22 635.00	**0.76**	22 635.00	71.00	22 635.00	31.13
st70	12	7	638.22	**0.41**	638.22	5.00	638.22	18.11
		10	632.54	**0.14**	632.54	4.00	632.54	12.56
		14	630.90	**0.08**	630.90	3.00	630.90	8.66
		17	636.19	**0.71**	636.19	9.00	636.19	11.16
		23	694.49	**192.41**	694.49	3739.00	694.49	1137.77
eil76	4	7	542.95	**0.30**	542.95	10.00	542.95	20.97
		10	545.02	**2.14**	545.02	38.00	545.02	18.60
		15	552.15	89.30	552.15	**58.00**	552.15	207.04
		19	[571.05, 560.36]	1.87%	563,96	**133.00**	563.95	**371.35**
		25	[613.72, 594.59]	3.12%	[612.85, 590.40]	3.66%	601.71	**1025.73**
pr76	8	7	101 401.33	**0.14**	101 401.00	5.00	101 401.33	25.29
		10	101 779.42	8.85	101 779.00	8.00	101 779.42	224.40
		15	103 663.31	224.94	103 663.00	**34.00**	[103822.35, 103097.47]	0.70%
		19	104 481.75	287.79	104 482.00	**7.00**	104 481.75	**45.62**
		25	110 073.94	38.56	110 074.00	**12.00**	110 073.94	867.49
rat99	8	9	1209.09	**0.72**	1209.09	12.00	1209.14	90.16
		14	1224.10	387.77	1224.10	**23.00**	[1249.35, 1217.48]	2.55%
		19	[1263.26, 1239.29]	1.90%	1245,16	**43.00**	[1264.52, 1236.82]	2.19%
		24	[1348.19, 1250.53]	7.24%	1273,23	**44.00**	[1276.13, 1261.17]	1.17%
		33	1373.37	**16.11**	1373.37	3003.00	[1373.37, 1334.28]	2.85%
kroA100	13	10	19 900.87	**3.29**	19 900.90	151.00	19 900.87	2993.41
		14	19 637.52	**0.74**	19 637.50	95.00	19 637.52	40.47
		20	19 868.64	**5.46**	19 868.60	30.00	19 868.64	57.24
		25	20 279.51	435.69	20 279.50	**51.00**	20 279.51	**77.87**
		33	22 303.23	**2504.77**	[23591.00, 21498.00]	8.87%	[22303.23, 21761.87]	2.43%
kroB100	20	10	20 823.12	**2.84**	20 823.10	145.00	20 823.12	1575.86
		14	20 762.88	**1.67**	20 762.90	143.00	20 762.88	1292.72
		20	20 660.05	**0.39**	20 660.00	75.00	20 660.05	114.70
		25	20 786.92	**3.12**	20 786.90	16.00	20 786.92	34.89
		33	[23679.36, 22624.32]	4.46%	[24968.40, 22204.60]	11.07%	[22923.42, 22412.71]	2.2%
kroC100	13	10	19 923.30	**2.35**	19 923.30	98.00	19 923.30	93.61
		14	19 938.84	**6.58**	19 938.84	67.00	19 938.84	77.78
		20	20 135.00	**23.12**	20 135.00	55.00	20 135.00	229.62
		25	20 427.96	543.10	20 428.00	**436.00**	20 427.96	**197.60**
		33	22 465.73	**2942.86**	[23759.00, 21536.60]	9.35%	[22465.73, 21559.53]	4.03%
kroD100	14	10	20 270.57	**0.14**	20 270.60	48.00	20 270.57	50.50
		14	20 267.23	**0.08**	20 267.20	42.00	20 267.23	46.87
		20	20 457.00	**14.88**	20 457.00	197.00	20 457.00	254.33
		25	20 671.19	155.73	20 671.20	**160.00**	20 671.19	**154.50**
		33	[22402.88, 22039.19]	1.62%	[22439.70, 21720.30]	3.21%	[22238.56, 22011.87]	1.02%
kroE100	12	10	20 766.43	**0.09**	20 766.40	25.00	20 766.43	28.92
		14	20 777.69	**0.42**	20 777.70	37.00	20 777.69	28.45
		20	20 937.39	**4.11**	20 937.40	65.00	20 937.39	51.43
		25	21 174.94	**38.75**	21 174.90	92.00	21 174.94	62.60
		33	22 782.98	**1429.21**	[22843.60, 22470.10]	1.64%	22 782.98	3054.13
rd100	14	10	7524.08	**2.66**	7524.08	147.00	7524.08	177.19
		14	7500.44	**0.15**	7500.44	57.00	7500.44	42.96
		20	7537.98	**13.76**	7537.98	101.00	7537.98	149.61
		25	7555.83	**8.43**	7555.83	42.00	7555.83	51.30
		33	[8149.65, 8125.84]	0.29%	[8211.20, 7919.09]	3.56%	[8131.25, 7996.03]	1.66%
Tests solved				48/55		48/55		45/55

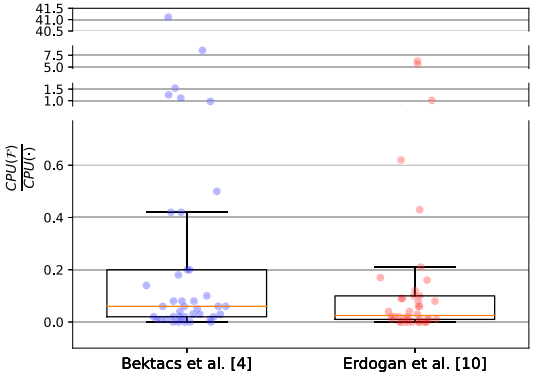
provided in Appendix F, while here we report the main conclusions. On the largest instances considered in this section (having $n \geq 100$) algorithm $\mathcal{F}$ is substantially faster than its variation without CPLEX cuts. On the contrary, the CPU times reported by the two algorithm variations is similar on the small- or medium-size instances. Overall, the variation without CPLEX cuts solves 3 instances less than $\mathcal{F}$, all having $n = 100$. Interestingly, the unsolved instances always have $n = 3p + 1$. We point out that the branch-and-cut algorithms tested in Bektaş et al. (2018) and Erdoğan et al. (2016) also use CPLEX cuts, hence the comparison made above is fair.

Finally we point out that, as a consequence of the effectiveness of $\mathcal{F}$, we have provided new optimal solutions to literature HpMP instances:

Experimental Observation 6. *Algorithm $\mathcal{F}$ solves to optimality instances* `kroA100` *and* `kroC100` *with $p = 33$, previously unsolved in the literature. Overall $\mathcal{F}$ solves to optimality 48 out of 55 literature HpMP instances*



(a) Performance profiles of algorithm $\mathcal{F}$ and of the branch-and-cut algorithms of [4] and [10] on literature HpMP instances with $58 \leq n \leq 100$. Each value on the y-axis is the number of instances solved within the time reported on the x-axis (in logarithmic scale).



(b) Boxplots (quartiles and median) of the CPU time ratios between algorithm $\mathcal{F}$ and the algorithms of [4] and [10] on literature HpMP instances with $58 \leq n \leq 100$ solved by all these algorithms. The y-axis has been "broken" to account for some outliers.

Fig. 2. Qualitative comparison of algorithm $\mathcal{F}$ with the branch-and-cut algorithms of Bektaş et al. (2018) and Erdoğan et al. (2016).

with $n \geq 58$, that is, 3 more than the algorithm of Erdoğan et al. (2016) and the same amount of the algorithm of Bektaş et al. (2018).

7.2. Results of algorithm $\mathcal{F}$: New instances

In this subsection we report on the performance of algorithm $\mathcal{F}$ in solving the new benchmark instances to integer optimality. First, we present the results obtained on the new TSPLIB-based instances. The results are presented in Table 2, formatted as Table 1. In this case, we highlight in boldface the CPU times of instances solved to integer optimality.

Overall, algorithm $\mathcal{F}$ solves 22 out of the 45 considered HpMP instances. For every $n \geq 195$, it finds the optimum of at least one HpMP instance. Most of the solved instances have $p \in \{\lfloor n/10 \rfloor, \lfloor n/7 \rfloor\}$ (6 occurrences each) followed by those with $p \in \{\lfloor n/5 \rfloor, \lfloor n/3 \rfloor\}$ (4 occurrences each). The most difficult instances in this set are those based on the TSPLIB instance `brg180`: on all of them, $\mathcal{F}$ reaches the time limit with a relative optimality gap greater than the 80%.

Next, we perform a similar analysis on the tetrahedral HpMP instances. The results of the experiments are reported in Table 3, following the same format as Table 2. Algorithm $\mathcal{F}$ solves to optimality 28 out of 75 tetrahedral instances. In particular, it solves to optimality all instances having $p = \lfloor n/3 \rfloor$ while the resolution of the instances with $p = \lfloor n/4 \rfloor, \lfloor n/5 \rfloor$ always reaches the time limit of one hour with a nonzero relative optimality gap. Instances with $p = \lfloor n/10 \rfloor, \lfloor n/7 \rfloor$ are always solved to optimality when $58 \leq n \leq 97$.

From the results of Tables 2 and 3 we draw the following conclusions:

Experimental Observation 7. *The new HpMP instances are generally more difficult to solve than the benchmark instances used in Bektaş et al. (2018) and Erdoğan et al. (2016). On the set of new TSPLIB-based instances, algorithm $\mathcal{F}$ is effective when p is low ($p \in \{\lfloor n/10 \rfloor, \lfloor n/7 \rfloor\}$) for all values of n. Algorithm $\mathcal{F}$ is also effective when $p = \lfloor n/3 \rfloor$ and $n \leq 200$ on both sets of new TSPLIB-based and tetrahedral HpMP instances.*

Our explanation of the effectiveness of $\mathcal{F}$ on the new instances with $p = \lfloor n/3 \rfloor$ is that the RCCs (7) have a strong impact. In particular, this is evident on the tetrahedral HpMP instances which are more difficult than the TSPLIB-based instances: the tetrahedral instances with $p > p^\star$ are mostly unsolved, except when the RCCs are involved in the formulation.

7.3. Results of algorithms $\mathcal{A}_{3p}$ and $\mathcal{A}_{3p+1}$

We now analyse algorithms $\mathcal{A}_{3p}$ and $\mathcal{A}_{3p+1}$ which are valid for the cases $n = 3p$ and $n = 3p+1$, respectively. We split the discussion into two parts: results on the literature instances and results on the new HpMP instances.

Tables 4 and 5 report on the former: Table 4 refers to the 8 literature HpMP instances with $n = 3p$, while Table 5 refers to the 10 literature HpMP instances $n = 3p+1$. For each considered algorithm, the values in the corresponding column are either the CPU time to reach optimality or the final relative optimality gap (in case of unsolved instances). Hyphens indicate unavailable results while boldface values indicate the best performance.

The results are summarized in the following:

Experimental Observation 8. *Algorithms $\mathcal{A}_{3p}$ and $\mathcal{A}_{3p+1}$ outperform the algorithms of Bektaş et al. (2018) and Erdoğan et al. (2016) on the literature HpMP instances with $n = 3p, 3p+1$: with a few exceptions (instances `eil76` and `pr76`) they both reduce by at least one order of magnitude the CPU times reported in Bektaş et al. (2018) and Erdoğan et al. (2016). Remarkably, $\mathcal{A}_{3p+1}$ solves instance `rd100`, previously unsolved by algorithm $\mathcal{F}$ and those of Bektaş et al. (2018) and Erdoğan et al. (2016).*

Next, we consider the results on the new HpMP instances, reported in Table 6 (for $n = 3p$) and Table 7 (for $n = 3p+1$). In this case we compare $\mathcal{A}_{3p}$ and $\mathcal{A}_{3p+1}$ with algorithm $\mathcal{F}$. Note that the upper part of Table 7 refers to the new TSPLIB-based instances having $n = 3p+1$, while its lower part refers to the tetrahedral HpMP instances (which all have $n = 3p+1$).

For both cases $n = 3p$ and $n = 3p+1$ the algorithms based on the alternative formulations solve the same instances as algorithm $\mathcal{F}$. We note that, while there are 5 unsolved new TSPLIB-based instances, all tetrahedral instances are solved to optimality by the considered algorithms. We explain this outcome by the large size of the new TSPLIB-based instances. Overall we have:

Experimental Observation 9. *On instances with $n = 3p$, algorithms $\mathcal{F}$ and $\mathcal{A}_{3p}$ perform similarly. On the other hand, on instances with $n = 3p+1$ algorithm $\mathcal{A}_{3p+1}$ is consistently better than $\mathcal{F}$.*

7.4. Impact of valid inequalities

In this section we discuss how the inequalities introduced in this paper affect the performance of the branch-and-cut algorithms. We

Table 2
Performance of the branch-and-cut algorithm $\mathcal{F}$ on the set of new TSPLIB-based instances. The values reported in the last column are times in CPU seconds employed by $\mathcal{F}$ to solve the instances, or the final relative optimality gaps for unsolved instances.

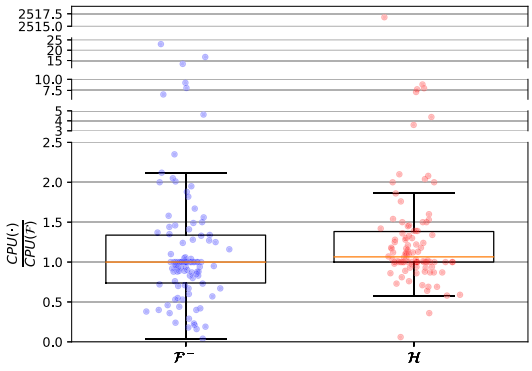
Instance	$p^\star$	p	$\mathcal{F}$	
			Final Bound	Time (s) and Gap(%)
dantzig42	8	4	648.00	**0.01**
		6	647.00	**0.01**
		8	646.00	**0.00**
		10	654.00	**0.55**
		14	796.00	**0.62**
pr124	23	12	52 479.47	**32.64**
		17	52 210.02	**255.94**
		24	51 487.56	**2.22**
		31	51 830.28	**5.26**
		41	57 672.38	**221.92**
u159	20	15	41 238.46	**2.77**
		22	41 208.78	**16.54**
		31	[41849.59, 41789.29]	0.14%
		39	[43377.72, 42189.09]	2.74%
		53	47 320.58	**129.42**
brg180	15	18	[10910.00, 1852.78]	83.02%
		25	[99410.00, 1923.09]	98.07%
		36	[190040.00, 2049.61]	98.92%
		45	[312280.00, 2090.00]	99.33%
		60	[88790.00, 2739.33]	96.91%
rat195	7	19	[2326.51, 2326.51]	**2978.33**
		27	[2493.26, 2337.32]	6.25%
		39	[3034.78, 2360.93]	22.20%
		48	[3524.14, 2388.50]	32.22%
		65	2678.97	**1595.81**
d198	32	19	[12100.43, 11997.56]	0.85%
		28	11 910.72	**1.45**
		39	11 927.47	**103.62**
		49	[12332.59, 12005.21]	2.65%
		66	[17046.55, 16615.30]	2.53%
pr226	43	22	[60693.96, 58030.75]	4.39%
		32	[58033.26, 57433.26]	1.03%
		45	57 177.55	**0.95**
		56	[108704.74, 57177.55]	47.40%
		75	[240399.60, 72872.36]	69.69%
gil262	30	26	2260.32	**26.86**
		37	2263.31	**167.72**
		52	[2283.96, 2277.61]	0.28%
		65	[2996.28, 2284.06]	23.77%
		87	[3597.48, 2426.12]	32.56%
rd400	51	40	14 675.53	**39.96**
		57	14 684.15	**496.84**
		80	[17587.95, 14717.43]	16.32%
		100	[24026.88, 14765.26]	38.55%
		133	[31933.62, 15576.99]	51.22%
Tests solved				22/45

begin by measuring the impact of the restricted 2-opt inequalities (10) and of the QHC inequalities (6), and we do so by comparing $\mathcal{F}$ with its following variants: (*i*) algorithm $\mathcal{F}^-$ obtained from $\mathcal{F}$ by excluding the restricted 2-opt inequalities (10); (*ii*) algorithm $\mathcal{H}$ obtained from $\mathcal{F}$ by imposing that only Hamiltonian inequalities (3) are added during the branch-and-cut process. Such inequalities are separated by modifying the routine for the general QHC inequalities as follows: if the current solution $x^\star$ is integer, we merge *all* cycles composing the graph $G_{x^\star}$ associated to $x^\star$; if $x^\star$ is not integer, we heuristically solve a STSP on the weighted *complete* graph on vertex set V (that is, here, unlike the separation of general QHC inequalities, edges $e \in E$ with $x^\star_e = 0$ are also considered), and the resulting Hamiltonian cycle inequality associated to the STSP solution is checked for violation.

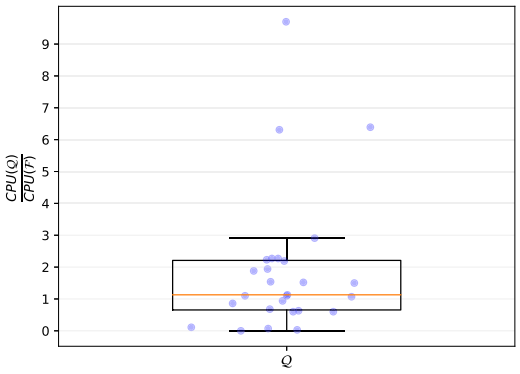
We test $\mathcal{F}$, $\mathcal{F}^-$ and $\mathcal{H}$ on all 230 H*p*MP instances described in Section 7. The instance-wise results of each algorithm are provided at the online repository (Barbato & Gouveia, 2023). Out of 230 H*p*MP instances, algorithm $\mathcal{F}$ solves 153 instances, algorithm $\mathcal{F}^-$ 149 instances and $\mathcal{H}$ 124 instances. Algorithm $\mathcal{F}^-$ reaches optimality on instances Tnm127 with $p = 12$ and Tnm139 with $p = 13$, unsolved by $\mathcal{F}$ and $\mathcal{H}$; algorithm $\mathcal{H}$ solves instance u159 with $p = 31$ unsolved by $\mathcal{F}$ and $\mathcal{F}^-$. With just one exception, the instances solved by $\mathcal{F}$ and not by $\mathcal{H}$ have $n = 3p, 3p + 1$, indicating that the QHC inequalities (6) associated with 4- or 5-cycles make algorithm $\mathcal{F}$ effective on those instances (recall that the RCCs are used in both $\mathcal{H}$ and $\mathcal{F}$).

In Fig. 3(a), we also provide a qualitative comparison of the CPU times of the three algorithms. The left-hand boxplot (resp. right-hand boxplot) shows the quartiles of the ratios of the CPU times of $\mathcal{F}^-$ (resp. $\mathcal{H}$) over the CPU times of $\mathcal{F}$ on the set of instances solved by *all* three algorithms (we additionally remove instances on which all three algorithms reported a resolution time of ~ 0 CPU seconds).

We see that, in the left-hand boxplot, the median (orange line) is precisely 1, meaning that $\mathcal{F}^-$ is at least as fast as $\mathcal{F}$ on about the 50% of instances (outliers excluded); instead, in the right-hand boxplot the median is higher than 1 and the lower quartile is precisely on 1,



(a) Boxplots (quartiles and median) of the CPU time ratios between algorithms $\mathcal{F}^-$ and $\mathcal{H}$ and algorithm $\mathcal{F}$ on the HpMP instances solved by all these algorithms. The y-axis has been "broken" to account for some outliers.



(b) Boxplot (quartiles and median) of the CPU time ratios between algorithms $\mathcal{Q}$ and $\mathcal{F}$ on the HpMP instances with $n = 3p, 3p+1$ solved by both algorithms.

Fig. 3. Qualitative comparison of algorithms $\mathcal{F}$, $\mathcal{F}^-$, $\mathcal{H}$ and $\mathcal{Q}$.

Table 3
Performance of the branch-and-cut algorithm $\mathcal{F}$ on the set of new tetrahedral HpMP instances. The values reported in the last column are times in CPU seconds employed by $\mathcal{F}$ to solve the instances or the final relative optimality gaps for unsolved instances.

Instance	$p^\star$	p	$\mathcal{F}$	
			Final Bound	Time (s) and Gap(%)
Tnm58	6	5	622 837.00	**2.95**
		8	613 845.00	**0.23**
		11	644 854.00	**1203.26**
		14	[677644.00, 660649.51]	2.51%
		19	720 292.00	**108.64**
Tnm67	7	6	773 270.00	**0.66**
		9	776 200.00	**7.26**
		13	[807704.00, 804905.40]	0.35%
		16	[866845.00, 816886.47]	5.76%
		22	903 164.00	**264.37**
Tnm79	8	7	927 604.00	**1.87**
		11	935 895.00	**24.31**
		15	[1038050.00, 962722.76]	7.26%
		19	[1230303.00, 980223.41]	20.33%
		26	1099608.00	**1529.92**
Tnm88	9	8	1076347.00	**6.43**
		12	1088995.00	**124.28**
		17	[1331268.00, 1115562.96]	16.20%
		22	[1481114.00, 1154703.29]	22.04%
		29	1279322.00	**620.48**
Tnm97	10	9	1228291.00	**15.47**
		13	1237110.00	**980.26**
		19	[1428613.00, 1280680.04]	10.35%
		24	[1782537.00, 1317702.15]	26.07%
		32	1456219.00	**532.54**
Tnm109	11	10	1387882.00	**194.37**
		15	[1447302.00, 1392115.66]	3.81%
		21	[1852239.00, 1437167.45]	22.40%
		27	[2131628.00, 1483382.85]	30.41%
		36	1641179.00	**179.84**
Tnm118	12	11	1537256.00	**42.84**
		16	[1561573.00, 1545577.49]	1.02%
		23	[2110338.00, 1596207.58]	24.36%
		29	[2991620.00, 1643500.31]	45.06%
		39	1819253.00	**45.22**
Tnm127	13	12	[1693543.00, 1680070.07]	0.79%
		18	[1848827.00, 1698162.53]	8.14%
		25	[2447140.00, 1755479.78]	28.26%
		31	[3344000.00, 1802990.12]	46.08%
		42	1998017.00	**35.85**

(*continued on next page*)

Table 3 (*continued*).

Instance	$p^\star$	p	$\mathcal{F}$	
			Final Bound	Time (s) and Gap(%)
Tnm139	14	13	[1874050.00, 1841952.83]	1.71%
		19	[1993687.00, 1855215.64]	6.94%
		27	[3644041.00, 1915452.43]	47.43%
		34	[3933351.00, 1972414.72]	49.85%
		46	2190466.00	**153.30**
Tnm148	15	14	[2051988.00, 1974848.15]	3.75%
		21	[2381756.00, 2011729.14]	15.53%
		29	[3886308.00, 2076360.43]	46.57%
		37	[4260302.00, 2146405.10]	49.61%
		49	2369686.00	**197.10**
Tnm157	16	15	[2181003.00, 2121773.35]	2.71%
		22	[2916679.00, 2162573.69]	25.85%
		31	[4198962.00, 2237908.65]	46.70%
		39	[4248684.00, 2307293.69]	45.69%
		52	2552529.00	**284.80**
Tnm169	17	16	[2349577.00, 2275189.58]	3.16%
		24	[2829992.00, 2330369.28]	17.65%
		33	[5128941.00, 2404038.24]	53.12%
		42	[6438722.00, 2482983.37]	61.43%
		56	2747867.00	**310.38**
Tnm178	18	17	[2593570.00, 2418482.30]	6.75%
		25	[3056457.00, 2481839.83]	18.80%
		35	[5131854.00, 2565729.59]	50.00%
		44	[5719971.00, 2646260.39]	53.73%
		59	2929777.00	**278.56**
Tnm187	19	18	[2644569.00, 2567982.49]	2.89%
		26	[3605184.00, 2634282.91]	26.93%
		37	[6596939.00, 2724064.37]	58.70%
		46	[7043438.00, 2803553.27]	60.19%
		62	3113091.00	**185.50**
Tnm199	20	19	[3611309.00, 2720407.23]	24.66%
		28	[5322946.00, 2792067.99]	47.54%
		39	[6780480.00, 2883225.78]	57.47%
		49	[8252541.00, 2969311.00]	64.01%
		66	3298277.00	**115.23**
Tests solved				28/75

Table 4
Performance of algorithm $\mathcal{A}_{3p}$ on literature HpMP instances with $n = 3p$. Comparison with the algorithms of Bektaş et al. (2018) and Erdoğan et al. (2016). The values reported in the columns corresponding to the algorithms are times in CPU seconds, or relative optimality gaps for unsolved instances. Hyphens indicate unavailable results.

Instance ($n = 3p$)	Time (s) and Gap(%)		
	$\mathcal{A}_{3p}$	Bektaş et al. (2018)	Erdoğan et al. (2016)
gr21	**0.00**	–	0.45
gr24	**0.01**	–	0.24
swiss42	**0.02**	1.00	1.12
att48	**6.52**	5.83%	285.90
gr48	**0.84**	208.00	24.25
hk48	**0.12**	118.00	10.04
eil51	**1.23**	1701.00	50.96
rat99	**25.75**	3003.00	2.85%
Tests solved	8/8	5/6	7/8

Table 5
Performance of algorithm $\mathcal{A}_{3p+1}$ on literature HpMP instances with $n = 3p + 1$. Comparison with the algorithms of Bektaş et al. (2018) and Erdoğan et al. (2016). The values reported in the columns corresponding to the algorithms are times in CPU seconds, or relative optimality gaps for unsolved instances.

Instance ($n = 3p + 1$)	Time (s) and Gap(%)		
	$\mathcal{A}_{3p+1}$	Bektaş et al. (2018)	Erdoğan et al. (2016)
brazil58	**0.94**	71.00	31.13
st70	**247.32**	3739.00	1137.77
eil76	2.61%	3.66%	**1025.73**
pr76	17.62	**12.00**	867.49
kroA100	**2112.57**	8.87%	2.43%
kroB100	**0.88%**	11.07%	2.23%
kroC100	3.50%	9.35%	4.03%
kroD100	**0.90%**	3.21%	1.02%
kroE100	**880.03**	1.64%	3054.13
rd100	**2164.67**	3.56%	1.66%
Tests solved	6/10	3/10	5/10

meaning that $\mathcal{H}$ is slower than $\mathcal{F}$ on about the 75% of instances (outliers excluded). The points drawn in Fig. 3(a) represent the ratios computed on single instances. In the right-hand boxplot, only two points are below 0.5, while there are 15 such points in the left-hand boxplot, meaning that $\mathcal{F}^-$ is substantially more effective than $\mathcal{H}$ in at least halving the CPU time of $\mathcal{F}$. Overall we get the following:

Experimental Observation 10. *The QHC inequalities have a strong impact on the HpMP formulation of Section 3; this is especially true when considering the difficult HpMP instances having $n = 3p, 3p + 1$, for which we separate the QHC inequalities associated with 4- and 5-cycles by using a dedicated separation routine; the restricted 2-opt inequalities are also useful, although their impact does not seem to be as decisive.*

We finally study the impact of the RCCs. We test algorithm $\mathcal{Q}$, obtained from $\mathcal{F}$ by excluding the RCCs, on all instances having $n = 3p, 3p + 1$ and we compare $\mathcal{Q}$ and $\mathcal{F}$. The instance-wise results are provided at the online repository (Barbato & Gouveia, 2023). Here we report that on the 13 instances with $n = 3p$, algorithms $\mathcal{Q}$ and $\mathcal{F}$ exhibit similar performances in terms of CPU times and number of solved instances (10 solved by $\mathcal{Q}$ and 11 solved by $\mathcal{F}$). Instead, $\mathcal{F}$ performs

Table 6
Performance of algorithm $\mathcal{A}_{3p}$ on new instances with $n = 3p$. Comparison with algorithm $\mathcal{F}$. The values reported in the columns corresponding to the algorithms are CPU times in seconds, or relative optimality gaps for unsolved instances.

Instance ($n = 3p$)	Time (s) and Gap(%)	
	$\mathcal{A}_{3p}$	$\mathcal{F}$
dantzig42	0.75	0.62
u159	489.10	**129.42**
brg180	97.77%	**96.91%**
rat195	**1314.25**	1595.81
d198	39.46%	2.53%
Tests solved	3/5	3/5

Table 7
Performance of algorithm $\mathcal{A}_{3p+1}$ on new instances with $n = 3p + 1$. Comparison with algorithm $\mathcal{F}$. The first part of the table refers to the new TSPLIB-based instances, the second part to the instances constructed from those of Hougardy and Zhong (2021). The values reported in the columns corresponding to the algorithms are CPU times in seconds, or relative optimality gaps for unsolved instances.

Instance ($n = 3p + 1$)	Time (s) and Gap(%)	
	$\mathcal{A}_{3p+1}$	$\mathcal{F}$
pr124	**66.79**	**221.92**
pr226	**52.63%**	69.69%
gil262	**25.51%**	32.56%
rd400	54.57%	**51.22%**
Tnm58	165.46	**108.64**
Tnm67	**202.07**	264.37
Tnm79	**1118.19**	1529.92
Tnm88	754.37	**620.48**
Tnm97	**333.12**	532.54
Tnm109	**89.76**	179.84
Tnm118	**34.37**	45.22
Tnm127	52.75	**35.85**
Tnm139	**131.13**	153.30
Tnm148	**105.31**	197.10
Tnm157	**136.76**	284.80
Tnm169	**108.93**	310.38
Tnm178	**108.6**	278.56
Tnm187	**150.62**	185.50
Tnm199	**65.63**	115.23
Tests solved	16/19	16/19

substantially better than $\mathcal{Q}$ on the 29 instances with $n = 3p + 1$, since it solves to optimality 23 instances while $\mathcal{Q}$ solves 18 instances.

Considering only the subset of 27 instances with $n = 3p, 3p+1$ solved to optimality by both algorithms, we show the distribution of the ratios of the CPU times of algorithm $\mathcal{Q}$ over the CPU times of algorithm $\mathcal{F}$ in the boxplot of Fig. 3(b), similar to those of Fig. 3(a). The median of the ratios is above 1 and the upper quartile is above 2, meaning that $\mathcal{Q}$ is slower than $\mathcal{F}$ on most of the tested instances (mostly having $n = 3p+1$). At the same time the position of the lower quartile below 1 indicates that $\mathcal{Q}$ can be faster than $\mathcal{F}$ on a non-negligible amount of instances. This is due to the good performance of $\mathcal{Q}$ on the instances with $n = 3p$. As a conclusion we have:

Experimental Observation 11. *The RCCs (7) have a strong impact on the performance of $\mathcal{F}$ on the instances with $n = 3p + 1$.*

8. Conclusion and future research

In this paper we have studied the Hamiltonian p-median problem, a location-routing problem in which we need to determine a minimum-weight set of p cycles spanning all vertices of an edge-weighted graph. For this problem we have proposed an integer linear programming formulation based on edge variables and two families of strengthening inequalities: a family of quasi-Hamiltonian cycle inequalities, valid for the general problem, and a family of restricted (multi-)cut constraints, valid for instances with $p = \lfloor n/3 \rfloor$. We have shown that both families contain inequalities inducing facets of the integer hull of the edge-variable formulation; moreover, we have shown that branch-and-cut algorithms based on the edge-variable formulation and the proposed inequalities are effective in tackling problem instances: on 110 instances the branch-and-cut algorithms exhibit comparable or better performance than that of state-of-the-art algorithms for the Hamiltonian p-median problem and have been able to solve three benchmark instances previously unsolved; finally the algorithms have been tested on 120 new problem instances (most of which of larger sizes than literature instances) solving 50 of them. The largest instances solved to optimality have 400 vertices.

Open topics of research. Our study has also led to some interesting research directions that we would like to investigate as a future work. The first one is to give sufficient and necessary conditions for the quasi-Hamiltonian cycle inequalities and the restricted cut and multi-cut constraints to be facet-inducing. The second research direction concerns the generalization of the principle behind the quasi-Hamiltonian cycle inequalities to their cycle covering counterpart, with an emphasis on their polyhedral properties and on their effective inclusion in branch-and-cut algorithms. A third research direction concerns the effectiveness of the covers cuts and mixed-integer rounding cuts of CPLEX in improving the linear relaxation of the edge-variable formulation, as observed in Section 7.1: a natural question is whether these two general-purpose families can be specialized to take into account the specific structure of the solutions of the Hamiltonian p-median problem. Finally, we would like to focus on the restricted (multi-)cut inequalities from two standpoints: first, designing effective separation algorithms for the restricted 3-cut constraints (valid for the instances of the Hamiltonian p-median problem with $n = 3p + 2$); second, generalizing the restricted (3-)cut constraints to generic values of p so to use them in alternative formulations for the general Hamiltonian p-median problem.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The list of instances and the tables with the details of the experimental results used in this study are available at the online repository Barbato and Gouveia (2023).

Acknowledgements

The authors are grateful to the three anonymous reviewers whose suggestions improved the paper.

Funding

This research was partially supported by Portuguese National Funding from FCT - Fundação para a Ciência e a Tecnologia, under projects PTDC/MAT-NAN/2196/2014 and UIDB/04561/2020, and partially supported by project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ejor.2024.02.032.

References

Applegate, D. L., Bixby, R. E., Chvátal, V., Cook, W., Espinoza, D. G., Goycoolea, M., et al. (2024). Concorde TSP solver. https://www.math.uwaterloo.ca/tsp/concorde/index.html.

Barbato, M., & Gouveia, L. (2023). Experimental results of: The Hamiltonian p-median problem: Polyhedral results and branch-and-cut algorithms. http://dx.doi.org/10.13130/RD_UNIMI/UDEATS.

Bektaş, T., Gouveia, L., & Santos, D. (2017). New path elimination constraints for multi-depot routing problems. *Networks*, *70*(3), 246–261.

Bektaş, T., Gouveia, L., & Santos, D. (2018). Revisiting the Hamiltonian *p*-median problem: A new formulation on directed graphs and a branch-and-cut algorithm. *European Journal of Operational Research*.

Benavent, E., & Martínez, A. (2013). Multi-depot multiple TSP: a polyhedral study and computational results. *Annals of Operations Research*, *207*(1), 7–25.

Branco, I., & Coelho, J. D. (1990). The Hamiltonian *p*-median problem. *European Journal of Operational Research*, *47*(1), 86–95.

Conforti, M., Cornuéjols, G., & Zambelli, G. (2014). *Graduate texts in mathematics*: *vol. 271*, *Integer programming*.

Cornuéjols, G., & Pulleyblank, W. R. (1980). Perfect triangle-free 2-matchings. In V. J. Rayward-Smith (Ed.), *Combinatorial optimization II* (pp. 1–7). Springer.

Dezső, B., Jüttner, A., & Kovács, P. (2011). LEMON–an open source C++ graph template library. *Electronic Notes in Theoretical Computer Science*, *264*(5), 23–45.

Erdoğan, G., Laporte, G., & Chía, A. M. R. (2016). Exact and heuristic algorithms for the Hamiltonian *p*-median problem. *European Journal of Operational Research*, *253*(2), 280–289.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. New York, NY, USA: W. H. Freeman & Co.

Glaab, H. (2000). Eine Variante des Traveling-Salesman-Problems mit mehreren Handlungsreisenden: Modelle, Algorithmen und Anwendung. Shaker.

Glaab, H. (2002). A new variant of a vehicle routing problem: Lower and upper bounds. *European Journal of Operational Research*, *139*(3), 557–577.

Glaab, H., & Pott, A. (2000). The Hamiltonian *p*–median problem. *The Electronic Journal of Combinatorics*, *7*(1), R42.

Gollowitzer, S., Gouveia, L., Laporte, G., Pereira, D. L., & Wojciechowski, A. (2014). A comparison of several models for the Hamiltonian *p*-median problem. *Networks*, *63*(4), 350–363.

Gollowitzer, S., Pereira, D. L., & Wojciechowski, A. (2011). New models for and numerical tests of the Hamiltonian *p*-median problem. In *International conference on network optimization* (pp. 385–394). Springer.

Gutin, G., & Punnen, A. P. (2006). *The traveling salesman problem and its variations, volume 12*. Springer Science & Business Media.

Hougardy, S., & Zhong, X. (2021). Hard to solve instances of the euclidean traveling salesman problem. *Mathematical Programming Computation*, *13*, 51–74.

Hupp, L., & Liers, F. (2013). A polyhedral study of the Hamiltonian *p*-median problem. *Electronic Notes in Discrete Mathematics*, *41*, 213–220.

IBM (2016). ILOG CPLEX 12.6 User Manual. http://public.dhe.ibm.com/software/products/Decision_Optimization/docs/pdf/usrcplex.pdf. (Accessed 01 November 2016).

Lancia, G., Rinaldi, F., & Serafini, P. (2015). Local search inequalities. *Discrete Optimization*, *16*, 76–89.

Marzouk, A. M., Moreno-Centeno, E., & Üster, H. (2016). A branch-and-price algorithm for solving the Hamiltonian *p*-median problem. *INFORMS Journal on Computing*, *28*(4), 674–686.

Nägele, M. (2021). *Efficient methods for congruency-constrained optimization*. PhD thesis, ETH Zurich.

Nägele, M., Sudakov, B., & Zenklusen, R. (2019). Submodular minimization under congruency constraints. *Combinatorica*, *39*(6), 1351–1386.

Reinelt, G. (1991). TSPLIB—A traveling salesman problem library. *ORSA Journal on Computing*, *3*(4), 376–384.

Reinelt, G. (1995). TSPLIB: a library of sample instances for the TSP (and related problems) from various sources and of various types. URL: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.