The Integrality Gap of the Traveling Salesman Problem is 4/3 if the LP Solution Has at Most $n + 8$ Non-Zero Components

Tullio Villa[1,3*], Eleonora Vercesi[2,3], Janos Barta[1], Monaldo Mastrolilli[1,3]

[1]Scuola universitaria professionale della Svizzera italiana, Lugano, Switzerland.
[2]Università della Svizzera italiana, Lugano, Switzerland.
[3]Dalle Molle Institute for Artificial Intelligence USI-SUPSI, Lugano, Switzerland.

*Corresponding author: tullio.villa@supsi.ch;
Contributing authors: eleonora.vercesi@usi.ch; janos.barta@supsi.ch; monaldo.mastrolilli@supsi.ch;

Abstract

We address the classical Dantzig–Fulkerson–Johnson formulation of the symmetric metric Traveling Salesman Problem and study the integrality gap of its linear relaxation, namely the Subtour Elimination Problem (SEP). This integrality gap is conjectured to be $\mathbf{4/3}$. We prove that, when solving a problem on $\boldsymbol{n}$ nodes, if the optimal SEP solution has at most $\boldsymbol{n + 8}$ non-zero components, then the conjecture is true. To establish this result, we devise a new methodology that combines theoretical analysis and computational verification.

Keywords: Symmetric Traveling Salesman Problem, Linear Programming Relaxation, Integrality Gap

1 Introduction

Let $K_n = (V_n, E_n)$ be the complete graph on n nodes, and let $\boldsymbol{c} \in \mathbb{R}^{E_n}$ be a non-negative cost vector. The *Traveling Salesman Problem* (TSP) consists of finding a Hamiltonian tour of minimum total cost. This paper focuses exclusively on the

symmetric formulation, where the graph K_n is undirected, hence we use ij and ji interchangeably to denote the edge connecting nodes i and j. Additionally, we only consider *metric* cost vectors, satisfying the triangle inequalities. Henceforth, we simply use the term TSP to indicate the symmetric metric TSP.

The TSP is typically formulated as an Integer Linear Program (ILP). Among the most prominent formulations is the one of Dantzig-Fulkerson-Johnson [1], whose associated relaxed Linear Problem (LP) is known in the literature as the Subtour Elimination Problem (SEP), or Held-Karp relaxation:

$$\text{TSP}: \min_{\boldsymbol{x}\in P^n_{\text{SEP}}\cap\mathbb{Z}^{E_n}} \sum_{e\in E_n} c_e x_e \ , \qquad \text{SEP}: \min_{\boldsymbol{x}\in P^n_{\text{SEP}}} \sum_{e\in E_n} c_e x_e \ .$$

The polytope on which they are described is the *subtour elimination polytope*:

$$P^n_{\text{SEP}} := \left\{ \boldsymbol{x}\in\mathbb{R}^{E_n} \mid \sum_{e\in\delta(v)} x_e = 2 \ \ \forall v\in V_n, \ \sum_{e\in\delta(S)} x_e \geq 2 \ \ \forall S\in\mathcal{S}, \ 0\leq x_e\leq 1 \ \ \forall e\in E_n \right\} ,$$

where $\mathcal{S} := \{S\subseteq V_n \mid 3\leq |S|\leq n-3\}$ and, for any subset $S\subseteq V_n$, $\delta(S)$ is the set of edges having exactly one node in S[1]. The term *subtour elimination constraints* refers to the second set of constraints in the definition of P^n_{SEP}.

For a given cost vector $\boldsymbol{c}$, we denote the optimal solutions of the integral and relaxed problem by $\text{TSP}(\boldsymbol{c})$ and $\text{SEP}(\boldsymbol{c})$, respectively. The *integrality gap* is the ratio of these two values; it can be evaluated for a single instance cost $\boldsymbol{c}$, as $\alpha(\boldsymbol{c})$, or considered in a worst case scenario, as α:

$$\alpha(\boldsymbol{c}) := \frac{\text{TSP}(\boldsymbol{c})}{\text{SEP}(\boldsymbol{c})} \ , \qquad \alpha := \sup_{\boldsymbol{c} \text{ metric}} \frac{\text{TSP}(\boldsymbol{c})}{\text{SEP}(\boldsymbol{c})} \ .$$

An alternative granularity of the integrality gap can also be defined by focusing on a specific vertex $\boldsymbol{x}\in P^n_{\text{SEP}}$. We denote this by $\text{Gap}(\boldsymbol{x})$, and it is given by[2]:

$$\text{Gap}(\boldsymbol{x}) := \max\left\{ \frac{\text{TSP}(\boldsymbol{c})}{\text{SEP}(\boldsymbol{c})} \ \mid \ \boldsymbol{c} \text{ metric} \ , \ \arg\min \text{SEP}(\boldsymbol{c}) \not\ni \boldsymbol{x} \right\} \ .$$

The exact value of α is currently unknown; in order to determine it, various approaches have been explored, with increasing interest over the past decades. Wolsey [2] was the first to establish a $3/2$ upper bound for α, with an analysis built on the Christofides' classic algorithm [3]. This bound remained untouched for nearly fifty years. The first breakthrough was achieved only in 2021, when Karlin, Klein, and Oveis Gharan proposed a better-than-$3/2$ approximation algorithm for the general TSP [4]. Its analysis [5] lead to the corresponding improvement on the upper bound for the integrality gap: $\alpha\leq 3/2-\varepsilon$, where $\varepsilon\sim 10^{-36}$. The best known lower bound for α is $4/3$,

[1] The brackets denoting singletons are omitted for brevity; this abuse of notation is used throughout this work, when the context does not lead to ambiguity, e.g., $\delta(v)=\delta(\{v\})$.

[2] The quantity Gap is defined as a maximum. Formally, it is more natural to define it as a supremum and then show that it is in fact attained. Indeed, the ratio under consideration is invariant under positive scaling of the cost vector $\boldsymbol{c}$; therefore, costs can be normalized, restricting the optimization to a compact domain. Since the ratio is continuous on this domain, it follows that the supremum is achieved and hence is a maximum.

achieved by families of instances that where already known in 1995, when Goemans [6] formulated the "4/3-conjecture", which asserts that the integrality gap is exactly 4/3. Despite several attempts, nobody was able to disprove the conjecture, nor to settle it conclusively.

Other than these lower and upper bounds, only results for specific subclasses of instances are available in the literature. Benoit and Boyd [7, 8] exploited exhaustive enumeration of the vertices of P^n_{SEP} to compute the exact value of the integrality gap for TSP instances with $n \leq 12$. Schalekamp, Williamson, and van Zuylen [9] conjectured that the maximum integrality gap is attained on *half-integer* instances, that is, cost vectors whose optimal SEP solutions have all the entries in $\{0, 1/2, 1\}$. For this class of instances, the integrality gap was shown to be strictly less than 3/2 by Karlin et al. in [10]. This achievement was improved in [11] and subsequently in the as-yet unpublished work [12]. In recent years, promising lines of research have examined subclasses of SEP solutions $\boldsymbol{x}$ characterized by specific properties of the so called *support graph*, defined as the undirected weighted graph $G_{\boldsymbol{x}} = (V_n, E_{\boldsymbol{x}})$ such that $ij \in E_{\boldsymbol{x}} \Leftrightarrow x_{ij} > 0$, and the weight on edge ij is given by x_{ij}. Notice that the number of edges in the support graph $G_{\boldsymbol{x}}$ is, by definition, the number of non-zero components of $\boldsymbol{x}$. Boyd and Carr [13] proved that the integrality gap is 4/3 when the support graph of the SEP solution contains disjoint 1/2 triangles. For graph-TSP instances[3], Boyd et al. [14] proved that the 4/3-conjecture holds for cubic graphs. Mömke and Svensson [15] improved this result showing that the integrality gap is 4/3 for graph-TSP restricted either to half-integral solutions or to a class of graphs that contains subcubic and claw-free graphs. Boyd and Sebő [16] proved that the integrality gap is at most ~ 1.4286 for instances having as SEP solution one of the so-called Boyd-Carr points (firstly defined in [13]). Jin et al. [17] proved the 4/3-conjecture for instances having as SEP solution cycle-cut points, that is, points for which every non-singleton tight set can be written as the union of two tight sets.

Different approaches have also been attempted to improve the lower bound. In [7, 18–20], families of TSP instances with high integrality gap, asymptotically tending to 4/3 were shown. In [21], the authors proposed an approach to heuristically generate instances with a high integrality gap: no instance with an integrality gap greater than 4/3 was found.

Contribution and outlook

This article extends the contributions of the paper [22] presented at the *27th Conference on Integer Programming and Combinatorial Optimization (IPCO 2026)*.

The work presents both a theoretical contribution and a novel methodology that opens new research directions in the study of the integrality gap. We prove the 4/3-conjecture for all the vertices of P^n_{SEP} whose support graph contains at most $n+8$ edges. Remarkably, this class is rich enough to contain vertices that are not covered by any previously known result in the literature. Figure 1 shows an example of such a vertex.

To establish our result, we define, for each integer k, the family $\mathcal{F}_k$ which comprises, across all polytopes P^n_{SEP} for $n \in \mathbb{N}$, the vertices whose support graphs contain exactly

[3] Graph-TSP is a subclass of metric TSP where the distance between the nodes is computed as the minimum number of edges separating them.

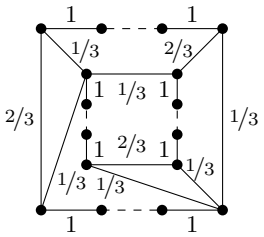

Fig. 1 Example of a vertex considered in this work but not in previous literature. Dashed edges may be replaced by paths of edges of weight 1 of arbitrary length.

$n + k$ edges. Although the family $\mathcal{F}_k$ is infinite, we identify a finite subset $\mathcal{A}_k$ of its elements, which we call *ancestors*, that allows us to make overall considerations on the entire family. We devise the *Gap-Bounding (GB) algorithm* and prove that its application on these ancestors returns upper bounds on the integrality gap of all the costs whose SEP solution is a vertex of $\mathcal{F}_k$. The application of the GB algorithm on the elements of $\mathcal{A}_k$ with $k \leq 8$ yields a computer-aided proof of the desired result.

The methodology presented constitutes in itself a novel way to approach the integrality gap problem, combining theoretical analysis and computational verification. Unlike earlier computational proofs (see [7, 8]) that were limited to enumerating vertices for a fixed dimension n, our framework fixes a small parameter k and derives results valid for *all* n, hereby achieving a form of universality through a computational approach. To the best of our knowledge, this is the first time that a computer-aided proof is implied in bounding the integrality gap for *infinitely many* vertices.

We conclude by noting that the vast majority of instances used in the literature to establish lower bounds on the integrality gap are special cases of our result (see, e.g., [7, 18–20]: counting nodes and edges in the constructions, one may see that the instances in [7, 18, 20] belong to $\mathcal{F}_3$ and the instances in [19] belong to $\mathcal{F}_6$). Other families, however, fall outside the scope of our analysis: the donut instances proposed in [16] have a non-constant surplus of edges over nodes, and the lower bounds on their integrality gap provided in the article converge to $4/3$. Interestingly, for each donut instance, the mentioned lower bound is consistently dominated by the integrality gap of an instance of $\mathcal{F}_3$ with the same number of nodes[4]. This observation aligns with the empirical evidence that, for a small number of nodes $n \leq 12$, the value of the integrality gap of a vertex correlates with the number of edges in its support graph (more details in Section 7). Taken together, these insights motivate the following hypothesis, which highlights the significance of the class of vertices considered in this work.

Few Edges Hypothesis. *For every n, among the vertices of P^n_{SEP}, the maximum integrality gap is realized on a vertex with the minimum possible number of edges in the support graph, that is $n + 3$.*

In summary, our work contributes to three different fronts.

[4]It can be checked comparing the explicit formulas for the lower bounds on the integrality gap in [7] and [16].

1. It establishes the 4/3 conjecture for newly identified infinite classes of vertices.
2. It demonstrates the potential of computer-aided proofs to advance our understanding of the integrality gap across infinite vertex classes, proposing a flexible framework that can be adapted to other contexts.
3. It opens the compelling research direction of investigating the correlation between the integrality gap and the number of edges in the support graph. A proof of the Few Edges Hypothesis, together with the contribution of this work, would result in definitively solving the 4/3 conjecture.

Beyond the results already reported in the conference version [22], the present article contains several substantial additions. First, we apply an improved computational pipeline that actively generates the ancestors in the subtour elimination polytope: this allows us to extend the application of our framework from the previous limit of $n+6$, where vertex lists were taken from the existing literature, to the current frontier $n+8$. Second, we provide a simplified proof of the correctness of the Gap-Bounding algorithm. Finally, we include in full detail all technical arguments omitted from the conference version due to space constraints.

Outline

Section 2 introduces the necessary background from the literature. Section 3 presents the families $\mathcal{F}_k$ and the notion of ancestors. Section 4 provides the theoretical basis for the GB algorithm, which is described in Section 5 together with the proof of its correctness. Section 6 details the computation at the core of our framework, both reporting the results of the execution of the GB algorithm, and illustrating the implementation of the entire computational pipeline. Section 7 explores future research directions originating from this work.

2 Background material

2.1 Preliminaries from graph theory

Before giving the list of definitions used in this work, we clarify that, to avoid confusion, given a graph $G = (V, E)$, the term *node* refers to the elements of V; the term *vertex* is reserved exclusively for the extreme points of P^n_{SEP}.

Definition 2.1 (Hamiltonian walk, Hamiltonian tour) Let G be an undirected graph. A *Hamiltonian walk* is a closed walk in G that visits every node at least once, possibly traversing the same edge multiple times. If every node is visited exactly once, we call the walk *Hamiltonian tour*. For a given walk $\boldsymbol{w}$, $w_{ij} \in \mathbb{N}$ denotes the *multiplicity* of ij in $\boldsymbol{w}$, that is, the number of times the walk $\boldsymbol{w}$ uses the edge ij (the analogous notation $t_{ij} \in \{0, 1\}$ is used for a tour $\boldsymbol{t}$).

For brevity, in what follows the terms *walk* and *tour* always indicate Hamiltonian walks and tours. Moreover, we use the same literal to denote both the walk $\boldsymbol{w}$ (respectively the tour $\boldsymbol{t}$) and its *characteristic vector*, that is, the vector of w_{ij} (respectively t_{ij}) entries. When the underlying graph G is not specified, we assume it is the complete one.

Remark 2.1 Since we have interest in the shortest possible walks $\boldsymbol{w}$, we assume that every edge ij is traversed at most twice, i.e., $w_{ij} \in \{0, 1, 2\}$[5]. Notice that, under this assumption, the number of walks considered on a graph becomes finite.

Definition 2.2 (Metric completion) Let $G = (V, E)$ be a connected graph with $|V| = n$, and let $\boldsymbol{c} \in \mathbb{R}^E$ be a non-negative cost on E. The *metric completion* of $\boldsymbol{c}$ is the metric cost $\boldsymbol{c}^*$ on the complete graph K_n which assigns to each edge $ij \in E_n$ the cost of the shortest path between i and j in G with respect to $\boldsymbol{c}$.

2.2 Preliminaries from the literature on the integrality gap

In this section, we collect from the literature (mainly from [7]) the definitions and results are used in this manuscript.

Theorem 2.2 ([7, 24]) *Let $\boldsymbol{x} \in P^n_{SEP}$ be a vertex. Then $|E_{\boldsymbol{x}}| \leq 2n - 3$.*

Definition 2.3 (1-edge, 1-path, from [7]) Let $\boldsymbol{x} \in P^n_{\text{SEP}}$ and let e be an edge of K_n; e is called 1-*edge* of $\boldsymbol{x}$ if $x_e = 1$. When $\boldsymbol{x}$ is fractional (that is, not a tour), we call 1-*path* of $\boldsymbol{x}$ a maximal path of 1-edges in the support graph $G_{\boldsymbol{x}}$; the nodes of degree 2 in the 1-path are called *internal nodes*, the two remaining nodes are called *end nodes*.

Theorem 2.3 ([7]) *Let $\boldsymbol{x} \in P^n_{SEP}$ be a fractional vertex. Then $\boldsymbol{x}$ has at least three distinct 1-paths.*

We now introduce a construction presented in [7] and give it a name using the authors' initials.

Definition 2.4 (BB-move, from [7]) Let $\boldsymbol{x} \in P^n_{\text{SEP}}$ and let ab be one of its 1-edges. We call BB-*move* the construction of a new point $\boldsymbol{x}' \in P^{n+1}_{\text{SEP}}$ defined as follows, where v is the new node added ($V_{n+1} = V_n \cup \{v\}$):

$$\begin{aligned} x'_{ab} &= 0\ , & x'_e &= 0 \quad \forall e \in \delta(v) \setminus \{av, vb\}\ , \\ x'_{av} &= x'_{vb} = 1\ , & x'_e &= x_e \quad \forall e \notin \delta(v) \cup \{ab\}\ . \end{aligned}$$

We denote this construction by $\text{BB}(\boldsymbol{x}, ab) := \boldsymbol{x}'$. When it is clear from the context or not strictly necessary, we omit the edge ab in the notation: $\text{BB}(\boldsymbol{x})$.

Theorem 2.4 ([7]) *Let $\boldsymbol{x} \in P^n_{SEP}$ and let ab be one of its 1-edges. Then $\text{BB}(\boldsymbol{x}, ab)$ is a vertex of P^{n+1}_{SEP} if and only if $\boldsymbol{x}$ is a vertex of P^n_{SEP}.*

3 Families of vertices with a given number of edges

In this section, we study the vertices of P^n_{SEP} considering, in their support graph, the relation between the number of edges and the number of nodes.

[5] It is well known (see, e.g. [23]) that, when an edge is used strictly more than twice by a walk, it is possible to get rid of two copies of that edge to obtain a shorter walk.

Lemma 3.1 Let $\boldsymbol{x}$ be a fractional vertex of P^n_{SEP}. Then $|E_{\boldsymbol{x}}| \geq n+3$.

Proof Consider three distinct 1-paths of $\boldsymbol{x}$ (they exist by Theorem 2.3) and let $\tilde{V}$ be the set of their end nodes, so that $|\tilde{V}| = 6$ and $\deg(v) \geq 3 \ \ \forall v \in \tilde{V}$. Then $|E_{\boldsymbol{x}}| = \frac{1}{2}\sum_{v\in V_n} \deg(v) = \frac{1}{2}\big(\sum_{v\in\tilde{V}} \deg(v) + \sum_{v\notin\tilde{V}} \deg(v)\big) \geq \frac{1}{2}\big(6{\cdot}3 + (n-6){\cdot}2\big) = n+3$. □

Our aim now is to describe, for a given k, all possible fractional vertices with n nodes and $n+k$ edges. For this purpose, we define the families of vertices $\mathcal{F}_k$:

$$\mathcal{F}_k := \{\boldsymbol{x} \text{ fractional vertex } \mid \ \boldsymbol{x} \in P^n_{\text{SEP}} \text{ for some } n,\ |E_{\boldsymbol{x}}| = n+k\} \ .$$

Notice that Lemma 3.1 implies $k \geq 3$, hence $\mathcal{F}_1 = \mathcal{F}_2 = \emptyset$. Observe also that a BB-move increases both the number of nodes and edges by 1, thus preserving their difference k. Therefore, in virtue of Theorem 2.4, a vertex is in $\mathcal{F}_k$ if and only if its image under a BB-move is in $\mathcal{F}_k$ as well. This brings us to define the *ancestors*, the vertices of $\mathcal{F}_k$ that can not be obtained as results of a BB-move, that is, vertices without internal nodes in the 1-paths.

Definition 3.1 (Ancestor of order k) An *ancestor of order* k is a vertex $\boldsymbol{x}$ of $\mathcal{F}_k$ with no node of degree 2. We denote the set of ancestors of order k as

$$\mathcal{A}_k := \{\boldsymbol{x} \in \mathcal{F}_k \mid \boldsymbol{x} \text{ has no node of degree } 2\} \ .$$

Definition 3.2 (Successor) Given a vertex $\boldsymbol{x}$, we call *successor* of $\boldsymbol{x}$ any vertex $\boldsymbol{x}'$ obtained by sequential applications of the BB-move.

We can thus recover the whole family $\mathcal{F}_k$ by taking the elements of $\mathcal{A}_k$ and replacing any 1-edge with a 1-path of arbitrary length; $\mathcal{F}_k$ may be seen as the set of successors of $\mathcal{A}_k$. At this point, to give a complete description of $\mathcal{F}_k$, it only remains to retrieve all the elements of $\mathcal{A}_k$.

Lemma 3.2 Let $\boldsymbol{x}$ be an ancestor in $\mathcal{A}_k$ and let n be its number of nodes. Then $k+3 \leq n \leq 2k$.

Proof Theorem 2.2 gives the inequality $n+k = |E_{\boldsymbol{x}}| \leq 2n-3$ which leads to $n \geq k+3$. The upper bound on n is derived from the fact that the minimum degree of the nodes of any ancestor $\boldsymbol{x}$ is 3, thus $n+k = |E_{\boldsymbol{x}}| = \frac{1}{2}\sum_{v\in V_n} \deg(v) \geq \frac{1}{2}{\cdot}3\,n$ which simplifies to $n \leq 2k$. □

Therefore, the elements of $\mathcal{A}_k$ are finite and belong to the polytopes P^n_{SEP} with $n = k{+}3, \dots, 2k$. Building on the approaches of [7] and [25], we devise a procedure to retrieve the complete lists of ancestors $\mathcal{A}_k$ for small $k \leq 8$; the implementation details are described in Section 6.1.

For instance, Figure 2 shows all the ancestors in $\mathcal{A}_4$: as $k=4$, they all belong to the lists of vertices of P^7_{SEP} and P^8_{SEP}. All the vertices of $\mathcal{F}_4$ are either of these shapes or with the 1-edges replaced with 1-paths of arbitrary length. Similarly, simply examining

the structure of the sole ancestor in $\mathcal{A}_3$, one can deduce that the family $\mathcal{F}_3$ consists precisely of those vertices formed by two 1/2-triangles connected by three 1-paths. In virtue of this characterization, one may see that Conjecture 4.1 in [7] supports the Few Edge Hypothesis.

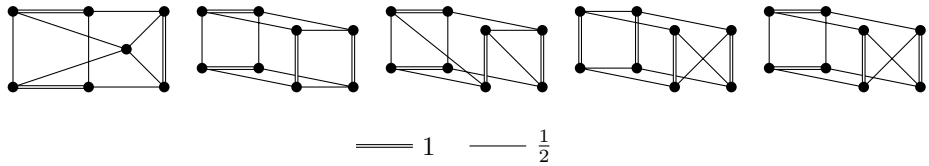

Fig. 2 Vertices of $\mathcal{A}_4$, up to isomorphism.

4 Redefining the Gap problem

In this section, we formulate a relaxed definition of Gap.

Definition 4.1 (Gap^+) Let $\boldsymbol{x} \in P^n_{\mathrm{SEP}}$ be a vertex. The *Gap*$^+$ *of* $\boldsymbol{x}$ is the quantity[6]

$$\mathrm{Gap}^+(\boldsymbol{x}) := \max\left\{ \frac{\mathrm{TSP}(\boldsymbol{c})}{\boldsymbol{c}\boldsymbol{x}} \;\middle|\; \boldsymbol{c} \text{ metric} \right\} .$$

By removing the requirement $\arg\min \mathrm{SEP}(\boldsymbol{c}) \not\ni \boldsymbol{x}$, we obtain the inequality:

$$\mathrm{Gap}^+(\boldsymbol{x}) \geq \mathrm{Gap}(\boldsymbol{x}) \; . \tag{1}$$

Our objective is therefore to provide an upper bound on Gap^+, which consequently yields a valid upper bound for Gap.

To this end, the following lemma studies the behavior of Gap^+ under the application of the BB-move.

Lemma 4.1 Let $\boldsymbol{x} \in P^n_{\mathrm{SEP}}$. Then $\mathrm{Gap}^+(\,\mathrm{BB}(\boldsymbol{x})\,) \geq \mathrm{Gap}^+(\boldsymbol{x})$. That is, the BB-move is Gap^+-increasing.

Proof Let $\boldsymbol{x}_0 \in P^n_{\mathrm{SEP}}$ be a vertex with a 1-edge ab. Let $\boldsymbol{x}_1 := \mathrm{BB}(\boldsymbol{x}_0, ab) \in P^{n+1}_{\mathrm{SEP}}$, with v the node added and av, vb the two 1-edges originated by the BB-move. Consider a metric cost $\boldsymbol{c}^0$ which realizes the Gap^+ on $\boldsymbol{x}_0$, namely $\mathrm{Gap}^+(\boldsymbol{x}_0) = \frac{\mathrm{TSP}(\boldsymbol{c}^0)}{\boldsymbol{c}^0\boldsymbol{x}_0}$. We construct the metric $\boldsymbol{c}^1$ on V_{n+1} starting from $\boldsymbol{c}^0$ and adding the node v at distance 0 form a: $c^1_{av} = 0$, $c^1_{vu} = c^0_{au}$ for all nodes $u \in V_n \setminus a$, and $c^1_e = c^0_e$ for all edges $e \in E_n$. Clearly $\boldsymbol{c}^1$ is metric and $\boldsymbol{c}^1\boldsymbol{x}_1 = \boldsymbol{c}^0\boldsymbol{x}_0$.

We first show that $\mathrm{TSP}(\boldsymbol{c}^0) \geq \mathrm{TSP}(\boldsymbol{c}^1)$. Let $\boldsymbol{t}_0$ be a tour on K_n that is optimal for $\mathrm{TSP}(\boldsymbol{c}^0)$, i.e., $\mathrm{TSP}(\boldsymbol{c}^0) = \boldsymbol{c}^0\boldsymbol{t}_0$. Let $\boldsymbol{t}^+_0$ be the tour on K_{n+1} which retraces the steps of $\boldsymbol{t}_0$

[6] As for Gap, the quantity Gap^+ can be defined as a supremum and one can then prove that it is in fact attained as a maximum.

but visits v immediately after a, that is, if the sequence of nodes visited by $\boldsymbol{t}_0$ is $a, u_2, \ldots, u_n$, then the sequence of nodes visited by $\boldsymbol{t}_0^+$ is $a, v, u_2, \ldots, u_n$. It is immediate to verify that $\boldsymbol{c}^1\boldsymbol{t}_0^+ = \boldsymbol{c}^0\boldsymbol{t}_0$, since $\boldsymbol{t}_0^+$ uses the same edges of $\boldsymbol{t}_0$ except for au_2, replaced by av and vu_2, which involve the same cost: $c^1_{av} + c^1_{vu_2} = 0 + c^0_{au_2}$. Therefore $\mathrm{TSP}(\boldsymbol{c}^1) \leq \boldsymbol{c}^1\boldsymbol{t}_0^+ = \boldsymbol{c}^0\boldsymbol{t}_0 = \mathrm{TSP}(\boldsymbol{c}^0)$.

On the other hand, $\mathrm{TSP}(\boldsymbol{c}^1) \geq \mathrm{TSP}(\boldsymbol{c}^0)$. Indeed, given an optimal tour $\boldsymbol{t}_1$ for $\mathrm{TSP}(\boldsymbol{c}^1)$, we can cut out v to get a tour $\boldsymbol{t}_1^-$ on the nodes of V_n, of cost $\boldsymbol{c}^0\boldsymbol{t}_1^- \leq \boldsymbol{c}^1\boldsymbol{t}_1$, by triangle inequality; thus $\mathrm{TSP}(\boldsymbol{c}^0) \leq \boldsymbol{c}^0\boldsymbol{t}_1^- \leq \boldsymbol{c}^1\boldsymbol{t}_1 = \mathrm{TSP}(\boldsymbol{c}^1)$.

The two inequalities together yield $\mathrm{TSP}(\boldsymbol{c}^1) = \mathrm{TSP}(\boldsymbol{c}^0)$, finally proving $\mathrm{Gap}^+(\boldsymbol{x}_1) \geq \frac{\mathrm{TSP}(\boldsymbol{c}^1)}{\boldsymbol{c}^1\boldsymbol{x}_1} = \frac{\mathrm{TSP}(\boldsymbol{c}^0)}{\boldsymbol{c}^0\boldsymbol{x}_0} = \mathrm{Gap}^+(\boldsymbol{x}_0)$. □

Remark 4.2 The two definitions Gap and Gap^+ are actually different: there exist vertices $\boldsymbol{x}$ realizing the strict inequality $\mathrm{Gap}(\boldsymbol{x}) < \mathrm{Gap}^+(\boldsymbol{x})$, as illustrated in Figure 3. The relaxed version Gap^+ has been introduced because the previous lemma, which is of crucial relevance throughout our work, fails for Gap: the vertex in Figure 3 serves as a counterexample for this fact as well. The reason behind this discrepancy is the following. An optimal cost $\boldsymbol{c}$ achieving $\mathrm{Gap}^+(\boldsymbol{x})$ need not lie in the optimal normal cone of $\boldsymbol{x}$ (i.e., one may have $\arg\min \mathrm{SEP}(\boldsymbol{c}) \not\ni \boldsymbol{x}$). Accordingly, while it is easy to modify $\boldsymbol{c}$ and obtain a cost for $\textsc{bb}(\boldsymbol{x})$ attaining at least an equal Gap^+ (as shown in the previous proof), the same modification may yield an unfeasible cost for $\mathrm{Gap}(\textsc{bb}(\boldsymbol{x}))$, not belonging to the normal cone of $\textsc{bb}(\boldsymbol{x})$. Nevertheless, despite the dissimilar pointwise behavior, both definitions lead to the same global limit: α. Since every cost belongs to the normal cone of at least one vertex, the integrality gap may be obtained as:

$$\begin{aligned} \alpha &= \sup\left\{\mathrm{Gap}(\boldsymbol{x}) \;\middle|\; \boldsymbol{x} \in P^n_{\mathrm{SEP}} \text{ vertex for some } n \in \mathbb{N}\right\} \\ &= \sup\left\{\mathrm{Gap}^+(\boldsymbol{x}) \;\middle|\; \boldsymbol{x} \in P^n_{\mathrm{SEP}} \text{ vertex for some } n \in \mathbb{N}\right\} . \end{aligned}$$

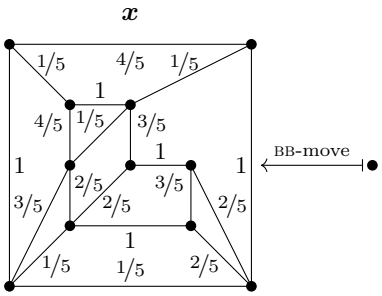

Fig. 3 Example of a vertex $\boldsymbol{x} \in P^{11}_{\mathrm{SEP}}$ exhibiting the differences between Gap and Gap^+ described in Remark 4.2. Here, $\mathrm{Gap}(\boldsymbol{x}) < \mathrm{Gap}^+(\boldsymbol{x})$ (1.1091 < 1.1111). Moreover, a BB-move on the 1-edge pointed by the arrow yields the vertex $\textsc{bb}(\boldsymbol{x})$ which satisfies $\mathrm{Gap}(\textsc{bb}(\boldsymbol{x})) < \mathrm{Gap}(\boldsymbol{x})$ (1.1000 < 1.1091).

When considering a vertex $\boldsymbol{x} \in P^n_{\mathrm{SEP}}$, the relevant information for the realization of the Gap^+ is found on the edges of the support graph, as formalized below.

Lemma 4.3 Let $\boldsymbol{x}$ be a vertex of P^n_{SEP} and let $\boldsymbol{c} \in \arg\max \mathrm{Gap}^+(\boldsymbol{x})$. Let $\boldsymbol{c}_{|E_{\boldsymbol{x}}}$ be the restriction of $\boldsymbol{c}$ on the edges of the support graph $G_{\boldsymbol{x}}$, and consider its metric completion $\tilde{\boldsymbol{c}} := (\boldsymbol{c}_{|E_{\boldsymbol{x}}})^*$ (see Definition 2.2). Then $\tilde{\boldsymbol{c}} \in \arg\max \mathrm{Gap}^+(\boldsymbol{x})$.

Proof By definition, $\tilde{\boldsymbol{c}}$ is metric and $\tilde{\boldsymbol{c}}_{|E_{\boldsymbol{x}}} = \boldsymbol{c}_{|E_{\boldsymbol{x}}}$, thus $\tilde{\boldsymbol{c}}\boldsymbol{x} = \boldsymbol{c}\boldsymbol{x}$. Moreover $\tilde{\boldsymbol{c}} \geq \boldsymbol{c}$ componentwise, hence $\mathrm{TSP}(\tilde{\boldsymbol{c}}) \geq \mathrm{TSP}(\boldsymbol{c})$. This gives $\frac{\mathrm{TSP}(\tilde{\boldsymbol{c}})}{\tilde{\boldsymbol{c}}\boldsymbol{x}} \geq \frac{\mathrm{TSP}(\boldsymbol{c})}{\boldsymbol{c}\boldsymbol{x}} = \mathrm{Gap}^+(\boldsymbol{x})$. The other verse of the inequality, $\frac{\mathrm{TSP}(\tilde{\boldsymbol{c}})}{\tilde{\boldsymbol{c}}\boldsymbol{x}} \leq \mathrm{Gap}^+(\boldsymbol{x})$, holds simply because $\mathrm{Gap}^+(\boldsymbol{x})$ is the maximum of such fractions. This proves $\frac{\mathrm{TSP}(\tilde{\boldsymbol{c}})}{\tilde{\boldsymbol{c}}\boldsymbol{x}} = \mathrm{Gap}^+(\boldsymbol{x})$. □

Remark 4.4 In virtue of the previous lemma, we henceforth assume that every cost associated with a vertex $\boldsymbol{x}$ is of the form $(\boldsymbol{c}_{|E_{\boldsymbol{x}}})^*$, that is, it is the metric completion of a cost defined on the support graph of $\boldsymbol{x}$. Under this convention, it is sufficient to specify costs only on the edges in $E_{\boldsymbol{x}}$. In addition, we may replace tours by walks on the support graph: this can be done by substituting any edge $ij \notin E_{\boldsymbol{x}}$ used by the tour with the shortest path (on the support graph) connecting i and j.

From this perspective, the Gap^+ quantity admits the equivalent characterization:

$$\mathrm{Gap}^+(\boldsymbol{x}) = \max\left\{ \frac{\mathrm{TSP}_{G_{\boldsymbol{x}}}(\boldsymbol{c})}{\boldsymbol{c}\boldsymbol{x}} \;\middle|\; \boldsymbol{c} \text{ non negative cost on } G_{\boldsymbol{x}} \right\}, \tag{2}$$

where $\mathrm{TSP}_{G_{\boldsymbol{x}}}(\boldsymbol{c})$ denotes the problem of finding a Hamiltonian walk on $G_{\boldsymbol{x}}$ of minimum $\boldsymbol{c}$ cost. The equivalence $\mathrm{TSP}_{G_{\boldsymbol{x}}}(\boldsymbol{c}) = \mathrm{TSP}(\boldsymbol{c})$ follows directly from Lemma 4.3 together with Remark 4.4. Moreover, although no explicit assumption is required on $\boldsymbol{c}$, metricity is enforced a posteriori in every maximal argument. To see this, consider an optimal cost $\boldsymbol{c}$ attaining $\mathrm{Gap}^+(\boldsymbol{x})$ and suppose that an edge ij is assigned a cost larger than that of a shortest path P between i and j; replacing $\boldsymbol{c}_{ij}$ by the $\boldsymbol{c}$-cost of P leaves $\mathrm{TSP}_{G_{\boldsymbol{x}}}$ unchanged, while strictly decreases the denominator $\boldsymbol{c}\boldsymbol{x}$, thereby yielding a larger ratio and contradicting maximality.

Similarly to the approach of Benoit and Boyd, in [7], we can design a linear problem which computes $\mathrm{Gap}^+(\boldsymbol{x})$. For a given a vertex $\boldsymbol{x}$, we define the LP denoted $\mathrm{OPT}^+(\boldsymbol{x})$ as follows[7]:

$$\begin{aligned}
&\mathrm{OPT}^+(\boldsymbol{x}): \\
&\text{minimize} && \sum_{ij \in E_{\boldsymbol{x}}} x_{ij} c_{ij} && && (3)\\
&\text{subject to:} && \sum_{ij \in E_{\boldsymbol{x}}} w_{ij} c_{ij} \;\geq 1 && \forall \boldsymbol{w} \text{ walk on } G_{\boldsymbol{x}}\,, && (4)\\
& && c_{ij} \geq 0 && \forall ij \in E_{\boldsymbol{x}}. && (5)
\end{aligned}$$

In the optimization process, the constraints of $\mathrm{OPT}^+(\boldsymbol{x})$ normalize $\mathrm{TSP}_{G_{\boldsymbol{x}}}(\boldsymbol{c}) = 1$; minimizing $\boldsymbol{c}\boldsymbol{x}$ is then equivalent to maximizing $1/\boldsymbol{c}\boldsymbol{x} = \mathrm{TSP}_{G_{\boldsymbol{x}}}(\boldsymbol{c})/\boldsymbol{c}\boldsymbol{x}$, thus

$$\frac{1}{\mathrm{OPT}^+(\boldsymbol{x})} = \mathrm{Gap}^+(\boldsymbol{x})\,. \tag{6}$$

[7] Note that, in the previous work [26], this LP was denoted $\mathrm{OPT}^{\mathrm{II}}$, as it arose in a more elaborate argument.

5 The Gap-Bounding algorithm

This section is entirely devoted to finding a way to bound the Gap^+ for all the vertices of a given family $\mathcal{F}_k$. To this end, we aim to contain the increase in Gap^+ originated by an iterative application of a BB-move: our goal is to provide a lower bound on OPT^+, which results in an upper bound on Gap^+, by (6).

We begin by outlining the central idea of our approach. Assume that we are given a vertex $\boldsymbol{x}_0$ alongside with an optimal solution of $\text{OPT}^+(\boldsymbol{x}_0)$ and a corresponding optimal solution $\boldsymbol{\mu}^0$ of the dual problem $\mathcal{D}\,\text{OPT}^+(\boldsymbol{x}_0)$. When considering a successor $\boldsymbol{x}'$ of $\boldsymbol{x}_0$, we intend to construct a feasible dual solution $\boldsymbol{\mu}'$ for $\mathcal{D}\,\text{OPT}^+(\boldsymbol{x}')$ starting from $\boldsymbol{\mu}^0$, so that the corresponding objective value results in a lower bound for $\text{OPT}^+(\boldsymbol{x}')$, as guaranteed by duality theory. Clearly, the crucial point in this procedure is to find a "good-enough" feasible dual solution, so that the bound obtained is meaningful (e.g., the trivial dual solution of all zeros yields 0 as objective value, which is a worthless bound for $\text{OPT}^+ = 1/\text{Gap}^+$). Moreover, for every vertex $\boldsymbol{x}$, we combine equations (1), (6), and duality theory to obtain

$$\text{Gap}(\boldsymbol{x}) \leq \text{Gap}^+(\boldsymbol{x}) = \frac{1}{\text{OPT}^+(\boldsymbol{x})} = \frac{1}{\mathcal{D}\,\text{OPT}^+(\boldsymbol{x})} \leq \frac{1}{L}\ , \tag{7}$$

where L is any lower bound on $\mathcal{D}\,\text{OPT}^+(\boldsymbol{x})$. In our case, L will be a bound on the objective value attained by $\boldsymbol{\mu}'$ (elaborated from $\boldsymbol{\mu}^0$). This strategy is visually represented in Figure 4: it will be the skeleton of the Gap-Bounding algorithm, discussed in detail in Section 5.1.

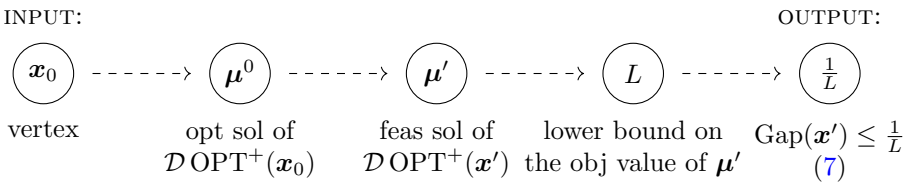

Fig. 4 Flowchart of the pipeline applied to bound Gap for a successor $\boldsymbol{x}'$ of $\boldsymbol{x}_0$.

Given a vertex $\boldsymbol{x}$, the dual problem $\mathcal{D}\,\text{OPT}^+(\boldsymbol{x})$ is derived as follows:

$$\begin{aligned}
&\mathcal{D}\,\text{OPT}^+(\boldsymbol{x}):\\
&\text{maximize} && \sum_{\boldsymbol{w}\text{ walk on }G_{\boldsymbol{x}}} \mu_{\boldsymbol{w}} && && (8)\\
&\text{subject to:} && \sum_{\boldsymbol{w}\text{ walk on }G_{\boldsymbol{x}}} w_e\mu_{\boldsymbol{w}} \leq x_e && \forall e \in E_{\boldsymbol{x}}, && (9)\\
& && \mu_{\boldsymbol{w}} \geq 0 && \forall \boldsymbol{w}\text{ walk on }G_{\boldsymbol{x}}. && (10)
\end{aligned}$$

This LP and its feasible solutions will be of central relevance throughout the remainder of the section.

5.1 Bounding the Gap on successors

The goal of this section is to derive, for a generic successor $\boldsymbol{x}'$ of $\boldsymbol{x}_0$, a feasible solution of $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}')$, starting from an optimal solution of $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_0)$; this serves as a bound on $\mathrm{Gap}(\boldsymbol{x}')$ (see the flowchart in Figure 4). To preserve the continuity of the argument, some technical claims are only stated; formal proofs are deferred to Section 5.2.

Let $\boldsymbol{x}_0$ be a vertex of P^n_{SEP} and let $\boldsymbol{\mu}^0 \in \arg\max \mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_0)$. We begin by considering a 1-edge ab of $\boldsymbol{x}_0$ and applying d consecutive BB-moves (see Definition 2.4): we insert nodes $a_1, \ldots, a_d$ and obtain, by Theorem 2.4, a vertex $\boldsymbol{x}_1$ with the 1-path $aa_1 \ldots a_d b$ in place of the 1-edge ab. We aim to design a feasible solution for $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_1)$.

An assignment $\boldsymbol{\mu}^1$ for $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_1)$ is indexed by the walks on $G_{\boldsymbol{x}_1}$. We construct walks $\boldsymbol{w}^1$ on $G_{\boldsymbol{x}_1}$, starting from the ones on $G_{\boldsymbol{x}_0}$, and assign weights to the corresponding components of $\boldsymbol{\mu}^1$. In this perspective, we divide the walks on $G_{\boldsymbol{x}_0}$ into three families, according to how many times the edge ab is traversed: we define, for $m = 0, 1, 2$, the collections $\mathcal{W}^{ab}_m := \{\boldsymbol{w} \text{ walk} \mid w_{ab} = m\}$.

- If $\boldsymbol{w}^0 \in \mathcal{W}^{ab}_0$, we construct $d+1$ walks on $G_{\boldsymbol{x}_1}$ as follows: for each $k = 0, \ldots, d$, the walk $\boldsymbol{w}^1_k$ retraces $\boldsymbol{w}^0$ but, when it arrives at a, it deviates to pass through $a_1, a_2, \ldots, a_k$ and back, and when it arrives at b, it deviates to $a_d, a_{d-1}, \ldots, a_{k+1}$ and back. We set $\mu^1_{\boldsymbol{w}^1_k} := \frac{1}{d+1}\mu^0_{\boldsymbol{w}^0}$ for all $k = 0, \ldots, d$.
- If $\boldsymbol{w}^0 \in \mathcal{W}^{ab}_1$, we construct the walk $\boldsymbol{w}^1$ on $G_{\boldsymbol{x}_1}$ that retraces $\boldsymbol{w}^0$ but replaces ab with $aa_1, a_1a_2, \ldots, a_{d-1}a_d, a_d b$. We set $\mu^1_{\boldsymbol{w}^1} := \mu^0_{\boldsymbol{w}^0}$.
- If $\boldsymbol{w}^0 \in \mathcal{W}^{ab}_2$, we construct the walk $\boldsymbol{w}^1$ on $G_{\boldsymbol{x}_1}$ that retraces $\boldsymbol{w}^0$ but replaces the double passage on ab by passing twice through $aa_1, a_1a_2, \ldots, a_{d-1}a_d, a_d b$. We set $\mu^1_{\boldsymbol{w}^1} := \mu^0_{\boldsymbol{w}^0}$.

Figure 5 illustrates the new assignment $\boldsymbol{\mu}^1$ in the case $d = 2$.

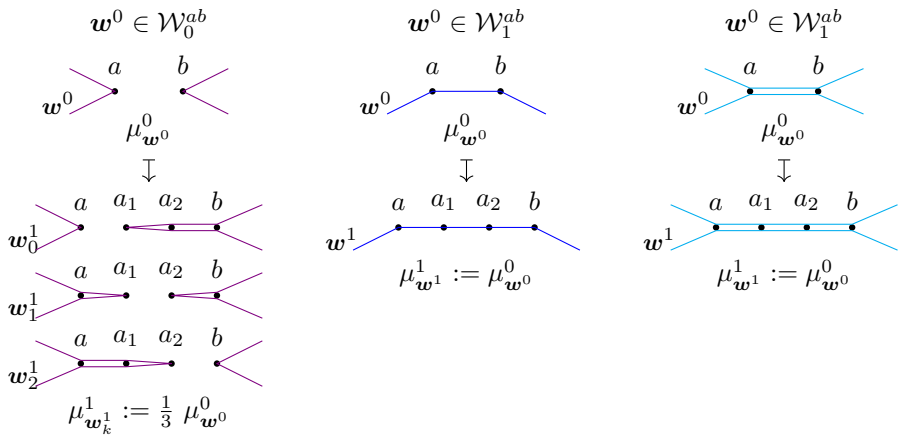

Fig. 5 Variables $\boldsymbol{\mu}^1$ for $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_1)$ constructed from variables $\boldsymbol{\mu}^0$ for $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_0)$. Here, $\boldsymbol{x}_1$ is obtained from $\boldsymbol{x}_0$ by replacing the 1-edge ab with a 1-path with $d = 2$ internal nodes.

Claim 5.1 (i) The objective value is the same for $\boldsymbol{\mu}^1$, in $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_1)$, and $\boldsymbol{\mu}^0$, in $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_0)$.

(ii) For every edge $e \in G_{\boldsymbol{x}_1}$ not in the 1-path $aa_1 \dots a_d b$, the value of the corresponding constraint (9) is the same for $\boldsymbol{\mu}^1$, in $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_1)$, and $\boldsymbol{\mu}^0$, in $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_0)$.

(iii) For every edge $e \in G_{\boldsymbol{x}_1}$ in the 1-path $aa_1 \dots a_d b$, the value of the corresponding constraint (9) in $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_1)$ is bounded by a constant $C(\boldsymbol{x}_0, ab)$ entirely determined by $\boldsymbol{\mu}^0$, and, in particular, independent of d:

$$C(\boldsymbol{x}_0, ab) := 2 \sum_{\boldsymbol{w}^0 \in \mathcal{W}_0^{ab}} \mu^0_{\boldsymbol{w}^0} + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_1^{ab}} \mu^0_{\boldsymbol{w}^0} + 2 \sum_{\boldsymbol{w}^0 \in \mathcal{W}_2^{ab}} \mu^0_{\boldsymbol{w}^0} \ .$$

The constant $C(\boldsymbol{x}_0, ab)$ may be used to rescale $\boldsymbol{\mu}^1$ and get the feasible assignment $\bar{\boldsymbol{\mu}}^1 := \frac{1}{C(\boldsymbol{x}_0, ab)} \boldsymbol{\mu}^1$ for $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_1)$. Since $C(\boldsymbol{x}_0, ab)$ depends only on $\boldsymbol{x}_0$, it works for all the vertices obtained by expanding the 1-edge ab of $\boldsymbol{x}_0$ to a 1-path of arbitrary length.

We now need to extend the construction for a generic successor $\boldsymbol{x}'$ of $\boldsymbol{x}_0$. To this end, we repeat the previous operation for all the 1-edges $e_1, \dots, e_p$ of $\boldsymbol{x}_0$, sequentially expanding them to 1-paths with $d_1, \dots, d_p$ internal nodes. Let $\boldsymbol{x}_1, \dots, \boldsymbol{x}_p$ be the vertices obtained along this process, leading to $\boldsymbol{x}_p = \boldsymbol{x}'$; accordingly, let $\boldsymbol{\mu}^1, \dots, \boldsymbol{\mu}^p$ be the assignments of dual variables originated from $\boldsymbol{\mu}^0$. The iterative application of Claim 5.1 immediately yields the following.

Claim 5.2 (i) The objective value is the same for $\boldsymbol{\mu}^p$, in $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_p)$, and $\boldsymbol{\mu}^0$, in $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_0)$.

(ii) For every fractional edge $e \in G_{\boldsymbol{x}_p}$, the value of the corresponding constraint (9) is the same for $\boldsymbol{\mu}^p$, in $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_p)$, and $\boldsymbol{\mu}^0$, in $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_0)$.

It only remains to prove the analogous of fact (iii) in Claim 5.1 for our sequential construction: we need upper bounds for the values attained by $\boldsymbol{\mu}^p$ on the constraints (9) corresponding to the 1-edges. To this aim, we recursively compute the required constants at each step $h = 1, \dots, p$ of the procedure, starting from the preceding vertex $\boldsymbol{x}_{h-1}$:

$$C(\boldsymbol{x}_{h-1}, e_h) := 2 \sum_{\boldsymbol{w}^{h-1} \in \mathcal{W}_0^{e_h}} \mu^{h-1}_{\boldsymbol{w}^{h-1}} + \sum_{\boldsymbol{w}^{h-1} \in \mathcal{W}_1^{e_h}} \mu^{h-1}_{\boldsymbol{w}^{h-1}} + 2 \sum_{\boldsymbol{w}^{h-1} \in \mathcal{W}_2^{e_h}} \mu^{h-1}_{\boldsymbol{w}^{h-1}} \ .$$

The final detail to show is that the factors $C(\boldsymbol{x}_{h-1}, e_h)$ are determined neither by the lengths of the 1-paths $d_1, \dots, d_p$, nor by the order in which the 1-edges are expanded: they can be computed directly from the starting vertex $\boldsymbol{x}_0$.

Claim 5.3 $C(\boldsymbol{x}_{h-1}, e_h) = C(\boldsymbol{x}_0, e_h)$.

At this stage of the procedure, we are equipped with an assignment $\boldsymbol{\mu}^p$ for $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_p) = \mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}')$. As observed along the process, $\boldsymbol{\mu}^p$ itself may result

unfeasible for the dual problem; however, it suffices to appropriately rescale the variables to ensure that the constraints (9) associated with the 1-edges are satisfied. This is achieved simply by correcting the assignment by the quantity:

$$C^*(\boldsymbol{x}_0) := \max_{h=1,\dots,p} \{C(\boldsymbol{x}_0, e_h)\} = \max_{h=1,\dots,p} \{C(\boldsymbol{x}_{h-1}, e_h)\} \ .$$

The claims of this section guarantee that the rescaled assignment $\boldsymbol{\mu}^* := \frac{1}{C^*(\boldsymbol{x}_0)} \cdot \boldsymbol{\mu}^p$ is feasible and attains the objective value $\frac{1}{C^*(\boldsymbol{x}_0)} \cdot \mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_0)$, which can not be larger than the optimal value of $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}')$. Therefore, we can rewrite (7) as follows, $\mathrm{Gap}^+(\boldsymbol{x}') = \frac{1}{\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}')} \leq \frac{C^*(\boldsymbol{x}_0)}{\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}_0)} = C^*(\boldsymbol{x}_0) \cdot \mathrm{Gap}^+(\boldsymbol{x}_0)$, which simplifies into

$$\mathrm{Gap}^+(\boldsymbol{x}') \ \leq \ C^*(\boldsymbol{x}_0) \cdot \mathrm{Gap}^+(\boldsymbol{x}_0) \ . \tag{11}$$

Crucially, this upper bound only depends on the initial vertex $\boldsymbol{x}_0$, and not on the structure of the successor $\boldsymbol{x}'$. Notice that we expect $C^*(\boldsymbol{x}_0)$ to be greater than or equal to 1, otherwise we would obtain $\mathrm{Gap}^+(\boldsymbol{x}') < \mathrm{Gap}^+(\boldsymbol{x}_0)$, contradicting Lemma 4.1.

We are finally able to design the Gap-Bounding (GB) algorithm as Algorithm 1, with the purpose of computing an upper bound on the Gap^+ of *all* the successors of a given input vertex.

Algorithm 1 Gap-Bounding algorithm

1: INPUT: $\boldsymbol{x} \in P^n_{\mathrm{SEP}}$ vertex
2: Solve $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x})$ and retrieve an optimal assignment $\{\mu_{\boldsymbol{w}}\}_{\boldsymbol{w}\text{ walk on }G_{\boldsymbol{x}}}$.
3: **for** each e 1-edge of $\boldsymbol{x}$ **do**
4: $\quad C(\boldsymbol{x}, e) \ \leftarrow \ 2 \sum_{\boldsymbol{w} \in \mathcal{W}_0^e} \mu_{\boldsymbol{w}} + \sum_{\boldsymbol{w} \in \mathcal{W}_1^e} \mu_{\boldsymbol{w}} + 2 \sum_{\boldsymbol{w} \in \mathcal{W}_2^e} \mu_{\boldsymbol{w}}$
5: **end for**
6: $C^*(\boldsymbol{x}) \ \leftarrow \ \max\{C(\boldsymbol{x}, e)\}_{e\text{ 1-edge of }\boldsymbol{x}}$
7: $\mathrm{Gap}^+(\boldsymbol{x}) \ \leftarrow \ 1/\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x})$
8: **return** $C^*(\boldsymbol{x}) \cdot \mathrm{Gap}^+(\boldsymbol{x})$

Theorem 5.4 *The Gap-Bounding algorithm, on input a vertex $\boldsymbol{x} \in P^n_{SEP}$, returns a value $GB(\boldsymbol{x})$ which is an upper bound for the gap of all the successors $\boldsymbol{x}'$ of $\boldsymbol{x}$: $Gap(\boldsymbol{x}') \leq GB(\boldsymbol{x})$.*

Proof This entire section is the proof of equation (11): for any successor $\boldsymbol{x}'$ of $\boldsymbol{x}$, it holds $\mathrm{Gap}^+(\boldsymbol{x}') \leq C^*(\boldsymbol{x}) \cdot \mathrm{Gap}^+(\boldsymbol{x}) = \mathrm{GB}(\boldsymbol{x})$. Equation (1) completes the argument. □

We remark that both the GB algorithm and Theorem 5.4 apply to all generic vertices $\boldsymbol{x}$, without requiring that they be ancestors. This fact can be exploited, as shown in the following lemma.

Lemma 5.5 Let $\boldsymbol{x} \in P^n_{\mathrm{SEP}}$ be a vertex and let $\boldsymbol{x}_0$ be its ancestor. Then $\mathrm{GB}(\boldsymbol{x})$ gives a bound for the gap of all the successors $\boldsymbol{x}'$ of $\boldsymbol{x}_0$ (even for $\boldsymbol{x}'$ that are not successors of $\boldsymbol{x}$): $\mathrm{Gap}(\boldsymbol{x}') \leq \mathrm{GB}(\boldsymbol{x})$.

Proof Let $\boldsymbol{x}'$ be a successor of $\boldsymbol{x}_0$. Let $e_1, \ldots, e_p$ be the 1-edges of $\boldsymbol{x}_0$ and let $d_1, \ldots, d_p$ and $d'_1, \ldots, d'_p$ be the lengths of the corresponding 1-paths of $\boldsymbol{x}$ and $\boldsymbol{x}'$, respectively. Consider a common successor of both $\boldsymbol{x}$ and $\boldsymbol{x}'$, e.g. the vertex $\tilde{\boldsymbol{x}}$ whose 1-paths have lengths $\max(d_1, d'_1), \ldots, \max(d_p, d'_p)$. Then $\mathrm{Gap}^+(\boldsymbol{x}') \leq \mathrm{Gap}^+(\tilde{\boldsymbol{x}}) \leq \mathrm{GB}(\boldsymbol{x})$ by Lemma 4.1 and Theorem 5.4. □

5.2 Technical proofs

This section merely contains the proofs previously omitted for readability.

Proof of Claim 5.1 The proof of fact (i) is the following straightforward computation:

$$\begin{aligned}
\sum_{\boldsymbol{w}^1 \text{ walk on } G_{\boldsymbol{x}_1}} \mu^1_{\boldsymbol{w}^1} &= \sum_{\boldsymbol{w}^0 \in \mathcal{W}_0^{ab}} \Big(\sum_{k=0}^{d} \mu^1_{\boldsymbol{w}^1_k} \Big) + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_1^{ab}} \mu^1_{\boldsymbol{w}^1} + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_2^{ab}} \mu^1_{\boldsymbol{w}^1} \\
&= \sum_{\boldsymbol{w}^0 \in \mathcal{W}_0^{ab}} (d{+}1) \frac{1}{d{+}1} \mu^0_{\boldsymbol{w}^0} + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_1^{ab}} \mu^0_{\boldsymbol{w}^0} + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_2^{ab}} \mu^0_{\boldsymbol{w}^0} \\
&= \sum_{\boldsymbol{w}^0 \text{ walk on } G_{\boldsymbol{x}_0}} \mu^0_{\boldsymbol{w}^0} \ .
\end{aligned}$$

To prove fact (ii), we observe that, for an edge e not in the 1-path $aa_1 \ldots b$, the multiplicity of the new walks $\boldsymbol{w}^1$ is the same of the ones they were originated from, i.e., $w^1_e = w^0_e$. The corresponding constraint (9), accordingly, maintains the same value:

$$\begin{aligned}
\sum_{\boldsymbol{w}^1 \text{ walk on } G_{\boldsymbol{x}_1}} w^1_e \, \mu^1_{\boldsymbol{w}^1} &= \sum_{\boldsymbol{w}^0 \in \mathcal{W}_0^{ab}} \Big(\sum_{k=0}^{d} (w^1_k)_e \, \mu^1_{\boldsymbol{w}^1_k} \Big) + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_1^{ab}} w^1_e \, \mu^1_{\boldsymbol{w}^1} + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_2^{ab}} w^1_e \, \mu^1_{\boldsymbol{w}^1} \\
&= \sum_{\boldsymbol{w}^0 \in \mathcal{W}_0^{ab}} \Big((d{+}1) \ (w^0_e \ \frac{1}{d{+}1} \mu^0_{\boldsymbol{w}^0}) \Big) + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_1^{ab}} w^0_e \, \mu^0_{\boldsymbol{w}^0} + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_2^{ab}} w^0_e \, \mu^0_{\boldsymbol{w}^0} \\
&= \sum_{\boldsymbol{w}^0 \text{ walk on } G_{\boldsymbol{x}_0}} w^0_e \, \mu^0_{\boldsymbol{w}^0} \ .
\end{aligned}$$

Finally, for an edge e of the 1-path $aa_1 \ldots a_d b$, we prove fact (iii) by explicit computation:

$$\begin{aligned}
\sum_{\boldsymbol{w}^1 \text{ walk on } G_{\boldsymbol{x}_1}} w^1_e \, \mu^1_{\boldsymbol{w}^1} &= \sum_{\boldsymbol{w}^0 \in \mathcal{W}_0^{ab}} \Big(\sum_{k=0}^{d} (w^1_k)_e \, \mu^1_{\boldsymbol{w}^1_k} \Big) + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_1^{ab}} w^1_e \, \mu^1_{\boldsymbol{w}^1} + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_2^{ab}} w^1_e \, \mu^1_{\boldsymbol{w}^1} \\
&= \sum_{\boldsymbol{w}^0 \in \mathcal{W}_0^{ab}} \Big((1{\cdot}0 + d{\cdot}2) \, \frac{1}{d{+}1} \mu^0_{\boldsymbol{w}^0} \Big) + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_1^{ab}} 1{\cdot}\mu^0_{\boldsymbol{w}^0} + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_2^{ab}} 2{\cdot}\mu^0_{\boldsymbol{w}^0} \\
&\leq \ 2 \sum_{\boldsymbol{w}^0 \in \mathcal{W}_0^{ab}} \mu^0_{\boldsymbol{w}^0} \ + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_1^{ab}} \mu^0_{\boldsymbol{w}^0} \ + \ 2 \sum_{\boldsymbol{w}^0 \in \mathcal{W}_2^{ab}} \mu^0_{\boldsymbol{w}^0} \ .
\end{aligned}$$

The upper bound computed depends solely on the assignment $\boldsymbol{\mu}^0$: the dependency on d was eliminated in the last inequality by bounding $\frac{d}{d+1}$ by 1. □

Proof of Claim 5.2 It follows directly from the iterative application of Claim 5.1. □

Proof of Claim 5.3 Before delving into the main argument of the proof, we introduce a notation that prevents ambiguities. For $m = 0, 1, 2$, we use $\mathrm{E}_m^{e_h}(\boldsymbol{w}^{h-1})$ (eventually with subscript k when $m = 0$) to denote the new walk(s) obtained when extending $\boldsymbol{w}^{h-1}$ (E for *extend*) to pass through the d_h new nodes added in the expansion of e_h to a 1-path, as illustrated in Figure 5. The new structures associated with $\boldsymbol{x}_h$ are thus constructed as follows:

- $\boldsymbol{w}^{h-1} \in \mathcal{W}_0^{e_h}$ originates d_h+1 walks with dual variables $\mathrm{E}_0^{e_h}(\boldsymbol{w}^{h-1})_k$, for $k = 0, \dots, d_h$, $\mu^h_{\mathrm{E}_0^{e_h}(\boldsymbol{w}^{h-1})_k} := \frac{1}{d_h+1}\mu^{h-1}_{\boldsymbol{w}^{h-1}}$;
- $\boldsymbol{w}^{h-1} \in \mathcal{W}_1^{e_h}$ originates the walk $\mathrm{E}_1^{e_h}(\boldsymbol{w}^{h-1})$, with dual variable $\mu^h_{\mathrm{E}_1^{e_h}(\boldsymbol{w}^{h-1})} := \mu^{h-1}_{\boldsymbol{w}^{h-1}}$;
- $\boldsymbol{w}^{h-1} \in \mathcal{W}_2^{e_h}$ originates the walk $\mathrm{E}_2^{e_h}(\boldsymbol{w}^{h-1})$, with dual variable $\mu^h_{\mathrm{E}_2^{e_h}(\boldsymbol{w}^{h-1})} := \mu^{h-1}_{\boldsymbol{w}^{h-1}}$.

We also use the indicator function $\mathbf{1}_W(w)$, which evaluates to 1 if $w \in W$, and 0 otherwise, so that we can compactly rewrite $C(\boldsymbol{x}_{h-1}, e_h)$ as:

$$C(\boldsymbol{x}_{h-1}, e_h) = \sum_{\substack{\boldsymbol{w}^{h-1} \\ \text{walk on } G_{\boldsymbol{x}_{h-1}}}} \sum_{m=0,1,2} \theta_m \cdot \mathbf{1}_{\mathcal{W}_m^{e_h}}(\boldsymbol{w}^{h-1}) \cdot \mu^{h-1}_{\boldsymbol{w}^{h-1}} \ ,$$

where $\theta_0 := 2,\ \theta_1 := 1,\ \theta_2 := 2$.

We prove the claim adopting a recursive strategy: we show that the constant $C(\boldsymbol{x}_{h-1}, e_h)$ can equivalently be computed by swapping steps h and $h-1$ in the construction, namely $C(\boldsymbol{x}_{h-1}, e_h) = C(\boldsymbol{x}_{h-2}, e_h)$. Repeating the same argument until the step 0 is reached proves the desired statement. For the sake of simplifying the indexing, we only consider the second step $h = 2$; the same argument is easily generalizable for the generic step $h \in \{2, \dots, p\}$.

We start by unfolding the definition of $C(\boldsymbol{x}_1, e_2)$, studying how the walks $\boldsymbol{w}^1$ on $G_{\boldsymbol{x}_1}$ are originated from the walks $\boldsymbol{w}^0$ on $G_{\boldsymbol{x}_0}$:

$$\begin{aligned}
C(\boldsymbol{x}_1, e_2) &= \sum_{\substack{\boldsymbol{w}^1 \\ \text{walk on } G_{\boldsymbol{x}_1}}} \sum_{m=0,1,2} \theta_m \cdot \mathbf{1}_{\mathcal{W}_m^{e_2}}(\boldsymbol{w}^1) \cdot \mu^1_{\boldsymbol{w}^1} \\
&= \sum_{\boldsymbol{w}^0 \in \mathcal{W}_0^{e_1}} \left(\sum_{k=0}^{d_1} \left(\sum_{m=0,1,2} \theta_m \cdot \mathbf{1}_{\mathcal{W}_m^{e_2}}(\mathrm{E}_0^{e_1}(\boldsymbol{w}^0)_k) \cdot \mu^1_{\mathrm{E}_0^{e_1}(\boldsymbol{w}^0)_k} \right) \right) \\
&\quad + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_1^{e_1}} \left(\sum_{m=0,1,2} \theta_m \cdot \mathbf{1}_{\mathcal{W}_m^{e_2}}(\mathrm{E}_1^{e_1}(\boldsymbol{w}^0)) \cdot \mu^1_{\mathrm{E}_1^{e_1}(\boldsymbol{w}^0)} \right) \\
&\quad + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_2^{e_1}} \left(\sum_{m=0,1,2} \theta_m \cdot \mathbf{1}_{\mathcal{W}_m^{e_2}}(\mathrm{E}_2^{e_1}(\boldsymbol{w}^0)) \cdot \mu^1_{\mathrm{E}_2^{e_1}(\boldsymbol{w}^0)} \right) .
\end{aligned}$$

Notice that, when extending a walk on the 1-edge e_1, its multiplicity on e_2 is not affected. Therefore, for all $m = 0, 1, 2$, we have the equivalent conditions:

$$\begin{array}{ccccccc}
\boldsymbol{w}^0 \in \mathcal{W}_m^{e_2} & \Leftrightarrow & \mathrm{E}_0^{e_1}(\boldsymbol{w}^0)_k \in \mathcal{W}_m^{e_2} & \Leftrightarrow & \mathrm{E}_1^{e_1}(\boldsymbol{w}^0) \in \mathcal{W}_m^{e_2} & \Leftrightarrow & \mathrm{E}_2^{e_1}(\boldsymbol{w}^0) \in \mathcal{W}_m^{e_2} \ ; \\
\mathbf{1}_{\mathcal{W}_m^{e_2}}(\boldsymbol{w}^0) & = & \mathbf{1}_{\mathcal{W}_m^{e_2}}(\mathrm{E}_0^{e_1}(\boldsymbol{w}^0)_k) & = & \mathbf{1}_{\mathcal{W}_m^{e_2}}(\mathrm{E}_1^{e_1}(\boldsymbol{w}^0)) & = & \mathbf{1}_{\mathcal{W}_m^{e_2}}(\mathrm{E}_2^{e_1}(\boldsymbol{w}^0)) \ .
\end{array}$$

With this observation, we can rewrite $C(\boldsymbol{x}_1, e_2)$ as $C(\boldsymbol{x}_0, e_2)$, completing the proof:

$$\begin{aligned} C(\boldsymbol{x}_1, e_2) = & \sum_{\boldsymbol{w}^0 \in \mathcal{W}_0^{e_1}} \left(\sum_{k=0}^{d_1} \Big(\sum_{m=0,1,2} \theta_m \cdot \mathbf{1}_{\mathcal{W}_m^{e_2}}(\boldsymbol{w}^0) \cdot \frac{1}{d_1+1} \mu^0_{\boldsymbol{w}^0} \Big) \right) \\ & + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_1^{e_1}} \left(\sum_{m=0,1,2} \theta_m \cdot \mathbf{1}_{\mathcal{W}_m^{e_2}}(\boldsymbol{w}^0) \cdot \mu^0_{\boldsymbol{w}^0} \right) \\ & + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_2^{e_1}} \left(\sum_{m=0,1,2} \theta_m \cdot \mathbf{1}_{\mathcal{W}_m^{e_2}}(\boldsymbol{w}^0) \cdot \mu^0_{\boldsymbol{w}^0} \right) \\ = & \sum_{\substack{\boldsymbol{w}^0 \\ \text{walk on } G_{\boldsymbol{x}_0}}} \sum_{m=0,1,2} \theta_m \cdot \mathbf{1}_{\mathcal{W}_m^{e_2}}(\boldsymbol{w}^0) \cdot \mu^0_{\boldsymbol{w}^0} \\ = & \; C(\boldsymbol{x}_0, e_2) \; . \end{aligned}$$

□

6 Computational results

This section reports how the execution of the Gap-Bounding algorithm yields the *computer-aided* proof of our main result, Theorem 6.1. Our pipeline starts by generating the set of ancestors $\mathcal{A}_k$ for $k \leq 8$. Subsequently, the Gap-Bounding algorithm is applied on each vertex $\boldsymbol{x} \in \mathcal{A}_k$.

We inform that, for some ancestors, a straightforward application of the GB algorithm returned a value higher than $4/3$. To improve the bounds returned, we exploited Lemma 5.5 and executed the GB algorithm on some selected successors. Even though there is no guarantee that the GB algorithm applied to successors gives strictly better results, we thought it reasonable that more information (more nodes, edges, and dual variables) would lead to a more detailed analysis and a tighter bound. Eventually, this intuition led us to accomplish our objective.

Theorem 6.1 *$Gap(\boldsymbol{x}) \leq \frac{4}{3}$ for all the vertices of $\boldsymbol{x} \in \mathcal{F}_k$ with $k \leq 8$.*

We implemented the GB algorithm in C++, employing the commercial software Gurobi [27] as the LP solver. The next subsections discuss the main implementation details, and the corresponding source code is available here.

6.1 Generation of $\mathcal{A}_k$

To generate the ancestors of $\mathcal{A}_k$ for small $k \leq 8$, we adapt the pipelines used in [7, 25] to our purpose.

For each dimension n in the range $k+3 \leq n \leq 2k$, perform the following steps.

1. Generate the collection $\mathcal{G}_k^n$ of all non-isomorphic graphs with exactly n nodes and $n+k$ edges that are 2-node-connected and of minimum degree at least 3. These are the suitable support graphs for the ancestors in $\mathcal{A}_k$. We generated $\mathcal{G}_k^n$ using the software nauty [28].

2. For each graph $G \in \mathcal{G}_k^n$, compute the set of all non-isomorphic vertices of P_{SEP}^n whose support graph is contained in G. This enumeration was carried out with the software PPL [29]. The output of PPL then requires additional filtering, by verifying that the support graph of the given vertex is precisely G, rather than a proper subgraph, and removing duplicates up to isomorphism. Let $\mathcal{A}^G$ denote the filtered set.

Collecting the vertices obtained with this procedure yields

$$\mathcal{A}_k \;=\; \bigcup_{k+3\le n\le 2k} \;\bigcup_{G\in\mathcal{G}_k^n} \mathcal{A}^G \;.$$

6.2 Computation of $\mathcal{D}\,\mathrm{OPT}^+$

Algorithm 1 computes $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x})$ (8) – (10) as a subroutine (line 2). Solving the dual problem with all the variables included from the outset is computationally impractical. Instead, we employ a standard column-generation approach, starting from a restricted set of variables, and enlarging it iteratively. At each iteration, a pricing problem is solved to determine whether there exists a Hamiltonian walk $\boldsymbol{w}$ whose associated variable $\mu_{\boldsymbol{w}}$ can improve the current solution of $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x})$. If such a walk exists, the corresponding variable is added to the master problem; otherwise, optimality for the full formulation has been certified. Selecting the most promising variable $\mu_{\boldsymbol{w}}$ to insert amounts to solving the separation problem for the current relaxation of the primal (see, e.g., [30]): namely, finding a minimum-cost Hamiltonian walk $\boldsymbol{w}$ in $G_{\boldsymbol{x}}$ under the edge costs $\boldsymbol{c}$ determined by the current primal solution. This can be accomplished by solving the following ILP.

$$\begin{aligned}
\text{minimize} \quad & \sum_{ij\in E_{\boldsymbol{x}}} c_{ij} w_{ij} & & \\
\text{subject to:} \quad & \sum_{ij\in\delta(v)} w_{ij} \;=\; 2\; d_v & & \forall v \in V_n, \\
& \sum_{ij\in\delta(S)} w_{ij} \;\ge\; 2 & & \forall S \in \mathcal{S}, \\
& w_{ij} \;\le\; 2 & & \forall ij \in E_{\boldsymbol{x}}, \\
& d_v \;\ge\; 1 & & \forall v \in V_n. \\
& w_{ij} \;\text{ integer} & & \forall ij \in E_{\boldsymbol{x}}, \\
& d_v \;\text{ integer} & & \forall v \in V_n.
\end{aligned}$$

Upon termination, the optimal $\boldsymbol{\mu}$ values in the model are used to compute the estimates $C(\boldsymbol{x}, e)$ required at line 4 of Algorithm 1.

6.3 Details on the obtained upper bounds

To prove Theorem 6.1, we executed the GB algorithm on all the ancestors of $\mathcal{A}_k$ with $k \le 8$. As anticipated at the beginning of the section, these initial runs have not always returned the desired 4/3 bound. Consequently, to accomplish our objective, the GB algorithm was also applied to certain successors of these "unsatisfying" ancestors, in

accordance with Lemma 5.5. Since there is some flexibility in choosing the next successor to explore, we adopted the following heuristic: given a vertex $\boldsymbol{x}$ with $\mathrm{GB}(\boldsymbol{x}) > 4/3$, we considered the successor $\boldsymbol{x}' = \textsc{bb}(\boldsymbol{x}, e)$ obtained by the application of a BB-move on the 1-edge e of $\boldsymbol{x}$ with the largest constants $C(\boldsymbol{x}, e)$ (as computed in Algorithm 1, line 4).

Notice that, when repeatedly applying the GB algorithm on subsequent successors, the procedure of solving $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x}')$ (subroutine of Algorithm 1, line 2) may benefit from the previous solution of $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x})$, already at our disposal: instead of restarting the column generation technique from zero, we may take the optimal walks of $\boldsymbol{x}$, transform them into walks on $\boldsymbol{x}'$ as described at the beginning of Section 5.1, and initialize the model with the associated variables.

Table 1 reports, for $k \leq 8$: the number of ancestors in $\mathcal{A}_k$ (up to isomorphism); the upper bound on the Gap obtained for the vertices in $\mathcal{F}_k$ (namely, the maximum of all the values returned by the GB algorithm applied on the ancestors in $\mathcal{A}_k$ or on the selected successors); the number of additional iterations of the GB algorithm needed on the ancestors to achieve the final bound; and the runtime. **Tuls:** Aggiungi specifiche delle macchine

Table 1 Computational results of the application of the Gap-Bounding algorithm. An asterisk (*) indicates a value updated with respect to [26]. The update stems from a revision of the vertex lists of P^{11}_{SEP} and P^{12}_{SEP}: earlier computations relied on the lists accompanying [8], which were later proven to be incomplete in [25].

k	$\lvert\mathcal{A}_k\rvert$	upper bound on Gap for $\mathcal{F}_k$	extra GB iterations Avg	extra GB iterations Max	runtime (s) Avg	runtime (s) Max	runtime (s) Total
3	1	4/3	0.00	0	0.08	0.08	0.08
4	5	4/3	0.40	2	0.10	0.12	0.52
5	44	4/3	1.02	7	0.20	0.50	8.58
6	716*	4/3	0.86	10	0.28	2.06	197.91
7	19386	4/3	0.57	17	0.37	10.36	7100.31
8	701749	4/3	0.39	36	0.51	485.37	358494.23

We remark that the values given in the table are upper bounds: in the perspective of proving the conjecture, we were satisfied with 4/3, but in principle the actual Gap on a certain family may be even lower.

7 Conclusion

This paper presents the main result of proving the 4/3-conjecture for vertices of P^n_{SEP} with at most $n+8$ edges in their support graphs. A natural continuation of this research consists in enhancing the computational pipeline to enlarge the set of ancestors on which to apply the GB algorithm, in order to collect bounds for families $\mathcal{F}_k$ with $k > 8$.

Besides this, the new *methodology* introduced is itself of independent interest, as it opens a new range of possible advancements in the field. Investigating novel transformations between vertices and analyzing their impact on the integrality gap

(akin to the analysis we presented for the BB-move) may lead to interesting results on the integrality gap for diverse infinite families of vertices.

To conclude, we also mention a last research direction, motivated by an intriguing, computationally verified observation. Using data from [7], for instances with few nodes, we noticed a correlation between the number of edges in the support graphs of the vertices of P^n_{SEP} (with $6 \leq n \leq 12$) and their integrality gap. As shown in Figure 6, for a fixed $n \leq 12$, the integrality gap is higher on vertices with few non-zero components and appears to decrease as the number of edges in the support graph increases. Although this correlation is currently supported only for small values of n, it presents a fascinating avenue for further investigation. For example, the donut instances of [16] have an integrality gap tending to $4/3$, and they do not belong to the families of instances considered in our result; nevertheless, for each donut instance, the proposed lower bound on its integrality gap is strictly smaller than the integrality gap of a vertex of $\mathcal{F}_3$ with the same number of nodes. Should one succeed in proving that, for every n, the maximum integrality gap on the polytope P^n_{SEP} is always attained on the vertices with the smallest number of edges, such a result together with the contribution of this work would be the two key ingredients to definitely resolve the $4/3$-conjecture.

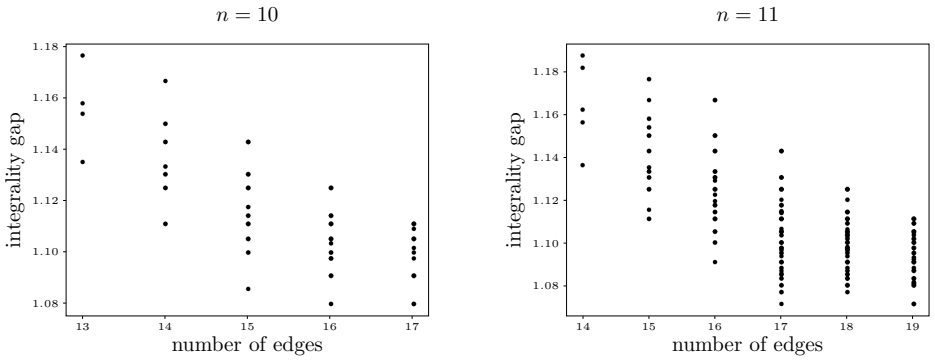

Fig. 6 Correlation between the number of edges in the support graph of a vertex and their integrality gap. Each plot collects all the vertices of P^n_{SEP} for a fixed n: $n = 10$ on the left, $n = 11$ on the right. The observed trend is the same for all small $n \leq 12$.

Acknowledgments

This work was supported by the Swiss National Science Foundation project n. 200021_212929/1 "Computational methods for integrality gaps analysis".

References

[1] Dantzig, G., Fulkerson, R., Johnson, S.: Solution of a large-scale traveling-salesman problem. Journal of the operations research society of America **2**(4), 393–410 (1954)

[2] Wolsey, L.A.: Heuristic analysis, linear programming and branch and bound. In: Rayward-Smith, V.J. (ed.) Combinatorial Optimization II, pp. 121–134. Springer, Berlin, Heidelberg (1980). https://doi.org/10.1007/BFb0120913

[3] Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. Operations Research Forum **3** (1976)

[4] Karlin, A.R., Klein, N., Oveis Gharan, S.: A (slightly) improved approximation algorithm for metric TSP. In: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing. STOC 2021, pp. 32–45. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3406325.3451009

[5] Karlin, A.R., Klein, N., Oveis Gharan, S.: A (slightly) improved bound on the integrality gap of the subtour LP for TSP. In: 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS), pp. 832–843. IEEE Computer Society, Denver, CO, USA (2022). https://doi.org/10.1109/FOCS54457.2022.00084

[6] Goemans, M.X.: Worst-case comparison of valid inequalities for the TSP. Mathematical Programming **69**(1), 335–349 (1995)

[7] Benoit, G., Boyd, S.: Finding the Exact Integrality Gap for Small Traveling Salesman Problems. Mathematics of Operations Research **33**, 921–931 (2008) https://doi.org/10.1287/moor.1080.0337

[8] Boyd, S., Elliott-Magwood, P.: Structure of the Extreme Points of the Subtour Elimination Polytope of the STSP. In: RIMS Kokyuroku Bessatsu, pp. 33–47 (2010). https://api.semanticscholar.org/CorpusID:15182527

[9] Schalekamp, F., Williamson, D.P., Zuylen, A.: 2-matchings, the traveling salesman problem, and the subtour LP: A proof of the Boyd-Carr conjecture. Mathematics of Operations Research **39**(2), 403–417 (2014)

[10] Karlin, A.R., Klein, N., Oveis Gharan, S.: An improved approximation algorithm for TSP in the half integral case. In: Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing. STOC 2020, pp. 28–39. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3357713.3384273

[11] Gupta, A., Lee, E., Li, J., Mucha, M., Newman, H., Sarkar, S.: Matroid-based TSP rounding for half-integral solutions. Mathematical Programming **206**(1), 541–576 (2024)

[12] Klein, N., Taziki, M.: Dual Charging for Half-Integral TSP. arXiv preprint arXiv:2507.17999 (2025)

[13] Boyd, S., Carr, R.: Finding low cost TSP and 2-matching solutions using certain

half-integer subtour vertices. Discrete Optimization **8**(4), 525–539 (2011) https://doi.org/10.1016/j.disopt.2011.05.002

[14] Boyd, S., Sitters, R., Ster, S., Stougie, L.: TSP on cubic and subcubic graphs. In: Integer Programming and Combinatoral Optimization, pp. 65–77. Springer, Berlin, Heidelberg (2011)

[15] Mömke, T., Svensson, O.: Removing and Adding Edges for the Traveling Salesman Problem. J. ACM **63**(1), 2–1228 (2016) https://doi.org/10.1145/2739008

[16] Boyd, S., Sebő, A.: The salesman's improved tours for fundamental classes. Mathematical Programming **186**(1), 289–307 (2021)

[17] Jin, B., Klein, N., Williamson, D.P.: A 4/3-approximation algorithm for half-integral cycle cut instances of the TSP. Mathematical Programming **209** (2025) https://doi.org/10.1007/s10107-025-02193-5

[18] Hougardy, S.: On the integrality ratio of the subtour LP for Euclidean TSP. Operations Research Letters **42**(8), 495–499 (2014)

[19] Hougardy, S., Zhong, X.: Hard to solve instances of the Euclidean Traveling Salesman Problem. Mathematical Programming Computation **13**, 51–74 (2021)

[20] Zhong, X.: Lower bounds on the integrality ratio of the subtour LP for the traveling salesman problem. Discrete Applied Mathematics **365**, 109–129 (2025)

[21] Vercesi, E., Gualandi, S., Mastrolilli, M., Gambardella, L.M.: On the generation of metric TSP instances with a large integrality gap by branch-and-cut. Mathematical Programming Computation **15**(2), 389–416 (2023)

[22] Villa, T., Vercesi, E., Barta, J., Mastrolilli, M.: The integrality gap of the traveling salesman problem is 4/3 if the LP solution has at most n+6 non-zero components. In: Dey, S.S., Di Summa, M., Salvagnin, D. (eds.) Integer Programming and Combinatorial Optimization, pp. 128–143. Springer, Cham (2026)

[23] Cornuejols, G., Fonlupt, J., Naddef, D.: The traveling salesman problem on a graph and some related integer polyhedra. Mathematical Programming **33**, 1–27 (1985)

[24] Boyd, S., Pulleyblank, W.R.: Optimizing over the subtour polytope of the travelling salesman problem. Mathematical Programming **49**, 163–187 (1990)

[25] Cook, W., Hougardy, S., Petrich, M.: Extending Exact Integrality Gap Computations for the Metric TSP (2026). https://arxiv.org/abs/2603.12995

[26] Villa, T., Vercesi, E., Barta, J., Mastrolilli, M.: The Integrality Gap of the Traveling Salesman Problem is 4/3 if the LP Solution Has at Most n+6 Non-zero Components. arXiv preprint (2025). https://arxiv.org/abs/2507.07003

[27] Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2024). https://www.gurobi.com

[28] McKay, B.D., Piperno, A.: Practical graph isomorphism, {II}. Journal of Symbolic Computation **60**(0), 94–112 (2014) https://doi.org/10.1016/j.jsc.2013.09.003

[29] Parma Polyhedra Library, B.: The Parma Polyhedra Library. https://www.bugseng.com/ppl/. Accessed: 2026-08-14

[30] Conforti, M., Cornuéjols, G., Zambelli, G.: Integer Programming Models. Springer, Cham (2014)