THE INTEGRALITY GAP OF THE TRAVELING SALESMAN PROBLEM IS $4/3$ IF THE LP SOLUTION HAS AT MOST $n+6$ NON-ZERO COMPONENTS.

ABSTRACT

In this paper, we address the classical Dantzig–Fulkerson–Johnson formulation of the 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 $4/3$. We prove that, when solving a problem on n nodes, if the optimal SEP solution has at most $n+6$ non-zero components, then the conjecture is true. To establish this result, we consider, for a given integer k, the infinite family $\mathcal{F}_k$ which gathers, among all the vertices of all the SEP polytopes P^n_{SEP} for $n \in \mathbb{N}$, the ones with exactly $n+k$ non-zero components. Then, we introduce a procedure that reduces the description of $\mathcal{F}_k$ to a *finite* set, and we present the *Gap-Bounding algorithm*, which provides provable upper bounds on the integrality gap for entire families $\mathcal{F}_k$. The application of the Gap-Bounding algorithm for $k \leq 6$ yields a computer-aided proof that the conjectured bound holds in this case.

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. If the graph K_n is directed, the problem is referred to as the *asymmetric* TSP; otherwise, it is called the *symmetric* TSP. In this paper, we focus exclusively on the symmetric case: henceforth, we simply use the term TSP to indicate the symmetric TSP. Accordingly, we will use ij and ji interchangeably to denote the same undirected edge connecting nodes i and j. Moreover, we restrict our attention to *metric* cost vectors, i.e., those satisfying the triangle inequality: for all nodes i, j, k, we have $c_{ik} + c_{kj} \geq c_{ij}$.

The TSP is typically formulated as an Integer Linear Program (ILP). Many formulations have been proposed in the literature: one of the most prominent is the Dantzig-Fulkerson-Johnson (DFJ) formulation [1], which introduces subtour elimination constraints to rule out infeasible subcycles. The DFJ formulation is as follows:

$$\begin{aligned}
\text{minimize} \quad & \sum_{e \in E_n} c_e x_e && && (1)\\
\text{subject to:} \quad & \sum_{e \in \delta(v)} x_e = 2 && \forall v \in V_n, && (2)\\
& \sum_{e \in \delta(S)} x_e \geq 2 && \forall S \in \mathcal{S}, && (3)\\
& 0 \leq x_e \leq 1 && \forall e \in E_n, && (4)\\
& x_e \ \text{integer} && \forall e \in E_n, && (5)
\end{aligned}$$

where $\mathcal{S} := \{S \subseteq V_n \mid 3 \leq |S| \leq n-3\}$ and $\delta(S)$ denotes, for any subset $S \subseteq V_n$, the set of edges having one node in S and one not in S; moreover, the brackets denoting singletons are omitted[1]. We call *node-degree constraints* the constraints of the form (2), *subtour-elimination constraints* (3), *edge constraints* (4), and *integrality constraints* (5). From now on, we denote by SEP (Subtour Elimination Problem) the relaxed Linear Program (LP) obtained from this ILP by dropping the integrality constraints (5), known in the literature also as Held-Karp relaxation. Furthermore, we denote by P^n_{SEP} the polytope associated to SEP, i.e. $P^n_{\text{SEP}} := \{\boldsymbol{x} \in \mathbb{R}^{E_n} \mid (2), (3), (4)\}$.

It becomes of interest to analyse the strength of this formulation, that is, the so-called *integrality gap of an instance*, defined as

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

where with $\text{TSP}(\boldsymbol{c})$, $\text{SEP}(\boldsymbol{c})$ we respectively denote the optimal value of (1) – (5) and (1) – (4) for a given cost vector $\boldsymbol{c}$. For a worst-case analysis, the quantity to examine is

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

broadly known as *integrality gap*. Its exact value is currently unknown; in order to determine it, various approaches have been explored. Wolsey [2] proposed an upper bound of $3/2$, which remains the best known to date. Later, [3, 4] exploited exhaustive enumeration of the vertices of P^n_{SEP} to compute the exact value of the integrality gap for TSP instances with $n \leq 12$. Other than that, only results for specific subclasses of instances are available in the literature, with a particular focus on those having *half-integer* SEP solutions, that is, solutions having all the entries in $\{0, 1/2, 1\}$. Schalekamp, Williamson, and van Zuylen [5] conjectured that the maximum integrality gap is attained on a cost vector having a half-integer solution as an optimal solution, and since that, progress has been made in this case. In recent years, promising lines of research have examined subclasses of instances whose optimal SEP solution $\boldsymbol{x}$ is 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 [6] proved that the integrality gap is $4/3$ when the support graph of the SEP solution contains disjoint $1/2$ triangles. Mömke and Svensson [7] showed that the integrality gap is $4/3$ for graph-TSP[2] restricted either to half-integral solutions or to a class of graphs that contains subcubic and claw-free graphs. With a breakthrough result, Karlin, Klein, and Oveis Gharan [8] gave an approximation algorithm leading to an integrality gap smaller than $3/2$ in the half-integer case. This factor has been improved later on

[1] This abuse of notation (e.g., $v = \{v\}$) will be used throughout this work, when the context does not lead to ambiguity.

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

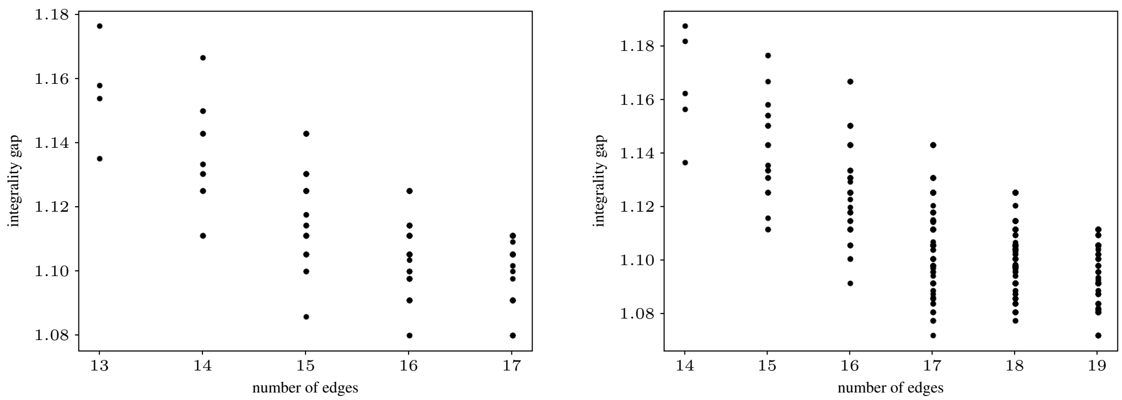

Figure 1: Correlation between the number of edges in the support graph of a vertex and the maximum integrality gap achievable by any cost vector having that vertex as an optimal solution. Each plot collects all the vertices of P^n_{SEP} for a fixed n: $n = 10$ on the left, $n = 11$ on the right.

by [9]. Boyd and Sebő [10] proved that the integrality gap is at most ~ 1.4286 for instances having as SEP solution one of the so-called *Boyd-Carr points* [6]. Very recently [11] 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. [3, 12, 13, 14] show families of TSP instances with high integrality gap, asymptotically tending to $4/3$. Recently [15] proposed an approach to *heuristically* generate instances with a high integrality gap: no instance with an IG greater than $4/3$ was found.

Our contribution. Our main contribution was inspired by an intriguing, computationally verified observation. Using data from [3], we noticed a strong correlation between the number of edges in the support graphs of the vertices of P^n_{SEP} (for $6 \leq n \leq 12$) and the maximum integrality gap achievable by any cost vector having that vertex as an optimal SEP solution. As shown in Figure 1, for $n \leq 12$, 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.

Guided by this observation, we consider, for any integer k, the family $\mathcal{F}_k$ of vertices whose number of edges in the support graph is exactly equal to the number of nodes n plus k. Although the family $\mathcal{F}_k$ is infinite, we identify a finite subset 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 for the integrality gap of all the costs whose SEP solution is a vertex of $\mathcal{F}_k$. By applying the GB algorithm on the ancestors of $\mathcal{F}_k$ with $k \leq 6$, we achieve a computer-aided proof that $\alpha(\boldsymbol{c}) \leq 4/3$ for every metric cost whose optimal SEP solution has at most $n + 6$ non-zero components.

Notice that the approach followed in this paper is fundamentally different from the one used in [3]. In the latter, the authors fix a small dimension n and enumerate all the vertices of P^n_{SEP}: the analysis conducted on the vertices coincides with the exhaustive computational exploration of them. Here, in an "orthogonal" fashion, we fix a small integer k and, for *any* n, we are able to describe all the vertices $\boldsymbol{x} \in P^n_{\text{SEP}}$ that have exactly $n + k$ edges in their support graph: the analysis starts with the exploration of finitely many vertices (the ancestors) but provides results valid for infinite families, which include vertices with an arbitrarily large number of nodes.

Another valuable characteristic of our result is that, being essentially based only on the number of non-zero components, it is straightforward to identify the instances on which it applies. We observe that many relevant TSP instances used in the literature to prove lower bounds for the integrality gap fall in this case, see, e.g., [3, 12, 13, 14][3].

Finally, our result implies that, if solving the SEP formulation yields a fractional solution with at most $n + 6$ non-zero components, then the optimal integer solution is guaranteed to be at most $4/3$ times the value of the SEP relaxation.

Outline. In Section 2, we recall some definitions from the literature that will be extensively used throughout the paper. In Section 3, we introduce the concept of ancestors, together with the procedure to retrieve them. In Section 4, we elaborate the theoretical background that allows us to properly define the GB algorithm, which is designed and

[3]Counting nodes and edges in the constructions, one may see that the instances proposed in [3, 12, 14] belong to $\mathcal{F}_3$ and the ones in [13] belong to $\mathcal{F}_6$.

described in detail in Section 5. In Section 6, we show how the GB algorithm can be used to bound the integrality gap for all vertices with at most $n+6$ non-zero components. Finally, in Section 7, we discuss possible future research directions inspired by our work. The last two sections are devoted to the technical aspects postponed with respect to the main argumentation: Section 8 presents the proofs removed from Section 5; Section 9 reports further details of the GB algorithm and the computational results obtained.

2 Background material

In this section, we introduce several definitions that will be used extensively throughout the manuscript. These are organized into two categories: (i) definitions and results from graph theory, and (ii) definitions and results from the literature on the integrality gap.

2.1 Preliminaries from graph theory

Before giving the list of definitions we use in this work, let us clarify that, to avoid confusion, given a graph $G=(V,E)$, we use the term *node* to refer to the elements of V; the term *vertex* is reserved exclusively for the extreme points of P^n_{SEP}. Furthermore, we distinguish between the terms *cost* and *weight*: the former is used for the objective of TSP or SEP; the latter denotes the components x_{ij} of a feasible solution, as they can be considered weights of the support graph.

Definition 2.1 (Graph isomorphism)**.** Let $G=(V_G,E_G)$ and $H=(V_H,E_H)$ be two undirected graphs with weight functions $\chi_G : E_G \to \mathbb{R}$ and $\chi_H : E_H \to \mathbb{R}$ respectively. We say that G and H are *isomorphic* if there exists a bijection $\phi : V_G \to V_H$ such that for all $i, j \in V,\ ij \in E_G$ if and only if $\phi(i)\phi(j) \in E_H$, and $\chi_G(ij) = \chi_H(\phi(i)\phi(j))$.

Definition 2.2 (Vertex isomorphism)**.** Two vertices $\boldsymbol{x}, \boldsymbol{x}' \in P^n_{\text{SEP}}$ are isomorphic if their support graphs are isomorphic.

We now make a clear distinction between the two following definitions, as it will be crucial throughout this work.

Definition 2.3 (Hamiltonian walk)**.** 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. For a given walk $\boldsymbol{w}$, let $w_{ij} \in \mathbb{N}$ denote the *multiplicity* of ij in $\boldsymbol{w}$, that is, the number of times the walk $\boldsymbol{w}$ uses the edge ij.

Definition 2.4 (Hamiltonian tour)**.** Let G be an undirected graph. A *Hamiltonian tour* is a closed walk in G that visits every vertex exactly once (hence no edge is traversed multiple times). For a given tour $\boldsymbol{t}$, let $t_{ij} \in \{0,1\}$ denote whether $\boldsymbol{t}$ uses or not the edge ij.

With abuse of notation, 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.

Throughout this work, we simplify this terminology and use the terms *walk* and *tour* to denote only Hamiltonian walks and Hamiltonian tours, respectively. If the underlying graph G is not specified, we assume it is the complete one.

Remark 2.1. Since we have interest only in walks which are the shortest possible, we assume that every edge ij of a walk $\boldsymbol{w}$ is traversed at most twice (i.e. $w_{ij} \in \{0,1,2\}$): indeed, it is well known (see, e.g. [16]) 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. It is important to notice that, imposing the condition $w_{ij} \in \{0,1,2\}$ on all edges ij, the number of walks on a graph becomes finite.

Definition 2.5 (Metric completion)**.** Let $G=(V,E)$ be a graph with $|V|=n$, and let $\boldsymbol{c} \in \mathbb{R}^E$ be a cost that satisfies the triangle inequalities on E. The *metric completion* of $\boldsymbol{c}$ is the cost $\boldsymbol{c}^*$ on the complete graph K_n defined as

$$c^*_{ij} = \begin{cases} c_{ij} & \text{if } ij \in E, \\ p(i,j) & \text{if } ij \notin E, \end{cases}$$

where $p(i,j)$ is 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 the investigation of the integrality gap, another granularity may be considered: the *integrality gap of a given dimension* n, defined as

$$\alpha_n := \sup_{\boldsymbol{c} \text{ metric on } n \text{ nodes}} \frac{\text{TSP}(\boldsymbol{c})}{\text{SEP}(\boldsymbol{c})}.$$

The study of the integrality gap α can be divided into the study of α_n and recovered as $\alpha = \sup_{n\in\mathbb{N}} \alpha_n$.

Benoit and Boyd, in [3], partition the integrality gap even further, introducing the following concept.

Definition 2.6 (Gap of a vertex)**.** Let $\boldsymbol{x} \in P^n_{\text{SEP}}$ be a vertex. The *Gap of* $\boldsymbol{x}$ is

$$\text{Gap}(\boldsymbol{x}) := \sup \left\{ \frac{\text{TSP}(\boldsymbol{c})}{\text{SEP}(\boldsymbol{c})} \;\mid\; \boldsymbol{c} \text{ metric}\;,\; \boldsymbol{x} \in \arg\min \text{SEP}(\boldsymbol{c}) \right\} .$$

The authors show that

$$\alpha_n = \max_{\boldsymbol{x} \text{ vertex of } P^n_{\text{SEP}}} \text{Gap}(\boldsymbol{x})$$

and they an LP named OPT$(\boldsymbol{x})$ to compute the inverse of Gap$(\boldsymbol{x})$. The interested reader may find all the details in [3].

We build our work upon these concepts, adapting them to suit our purposes. Other definitions and results from [3] are extensively used in this paper and are presented below.

Definition 2.7 (1-edge, 1-path, from [3])**.** 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 2.1 (Theorem 3.4 of [3])**.** *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 [3] and give it a name using the authors' initials.

Definition 2.8 (BB-move)**.** 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 w is the new node added:

$$x'_e = \begin{cases} 0 & \text{if } \ e = ab \\ 1 & \text{if } \ e \in \{aw, wb\} \\ 0 & \text{if } \ e \in \delta(w) \setminus \{aw, wb\} \\ x_e & \text{otherwise} \end{cases}$$

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 2.2 (Theorem 3.2 in [3])**.** *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}.*

In other words, inserting a node in a 1-path is an operation that maps vertices to vertices. The inverse operation, namely removing one internal node from a 1-path, is also a vertex-preserving operation. Interestingly, the authors do not mention anything about how this operation impacts the value of Gap$(\boldsymbol{x})$. In Section 4, we will discuss this point and provide some insights within a relaxed framework.

Finally, the following result from [3] will also be useful for our work: it bounds the number of edges in the support graph of a vertex in relation to the number of nodes n.

Theorem 2.3 (Theorem 3.1 of [3])**.** *Let $\boldsymbol{x} \in P^n_{SEP}$ be a vertex. Then $|E_{\boldsymbol{x}}| \leq 2n - 3$.*

3 Families of vertices with a given number of edges

In this section, we will study the vertices of P^n_{SEP} considering, in their support graph, the relation between the number of edges and the number of nodes. We start by stating the following simple Lemma.

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 2.1) 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

$$|E_{\boldsymbol{x}}| = \frac{1}{2} \sum_{v \in V_n} \deg(v) = \frac{1}{2} \left(\sum_{v \in \tilde{V}} \deg(v) + \sum_{v \notin \tilde{V}} \deg(v) \right) \geq \frac{1}{2} \left(\, 6 \cdot 3 + (n-6) \cdot 2 \, \right) = n + 3.$$

□

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 $\mathcal{F}_k$:

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

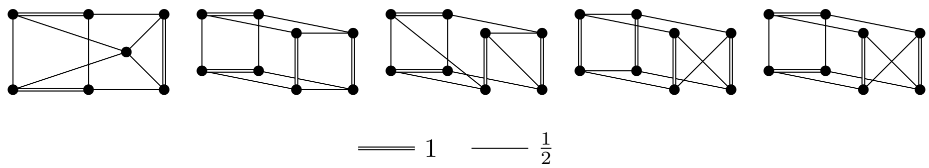

Figure 2: Vertices of $\mathcal{A}_4$, up to isomorphism.

Notice that Lemma 3.1 implies $k \geq 3$, that is $\mathcal{F}_1 = \mathcal{F}_2 = \emptyset$. We also observe that a BB-move add 1 node and 1 edge, therefore, in virtue of Theorem 2.2, the application of a BB-move to a vertex $\boldsymbol{x} \in P^n_{\text{SEP}}$ with $|E_{\boldsymbol{x}}| = n + k$ yields a vertex $\boldsymbol{x'} \in P^{n+1}_{\text{SEP}}$ with $|E_{\boldsymbol{x'}}| = (n + 1) + k$. In other words, 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 $\mathcal{F}_k$ with the shortest possible 1-paths (i.e., 1-paths composed of a single 1-edge).

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 (i.e., no internal node in the 1-paths). 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 $\mathcal{A}_k$. The following lemma helps us in this direction.

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 2.3 gives the inequality $n + k = |E_{\boldsymbol{x}}| \leq 2n - 3$ which leads to $k + 3 \leq n$. The upper bound on n is derived from the fact that the minimum degree of the nodes of $\boldsymbol{x}$ 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, all the elements of $\mathcal{A}_k$ can be recovered from the lists of vertices of P^n_{SEP}, with n up to $2k$ (provided we have them at our disposal). We simply need to scroll through the lists of vertices of P^n_{SEP} for all the $n = k{+}3, \dots, 2k$, and extract the ones with no node of degree 2 and with exactly $n + k$ edges. Gathering all of them, we get $\mathcal{A}_k$.

Since, at the time being, an exhaustive list of fractional vertices (up to isomorphism) is already available for n up to 12 ([3], [4]), it is also possible to completely determine $\mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5, \mathcal{A}_6$ (up to isomorphism) by Lemma 3.2. For instance, in Figure 2 we can see all the ancestors in $\mathcal{A}_4$: all the vertices of $\mathcal{F}_4$ are either of these shapes or with the 1-edges replaced with 1-paths of arbitrary length. As $k = 4$, these ancestors can be extracted from the lists of vertices of P^7_{SEP} and P^8_{SEP}.

4 Redefining the Gap problem

In this section, we exploit the idea proposed in [3] of dividing the study of α_n into subproblems Gap($\boldsymbol{x}$) associated to the vertices $\boldsymbol{x}$ of the polytope P^n_{SEP}. In [3], Benoit and Boyd explain how to craft a linear problem OPT($\boldsymbol{x}$) for each vertex $\boldsymbol{x} \in P^n_{\text{SEP}}$, in order to compute Gap($\boldsymbol{x}$) $= 1/\text{OPT}(\boldsymbol{x})$. The integrality gap α_n is finally recovered as $\alpha_n = \max\{\text{Gap}(\boldsymbol{x}) \mid \boldsymbol{x} \text{ vertex of } P^n_{\text{SEP}}\}$.

In this work, we build upon the same idea, slightly changing the definition of Gap (see Definition 2.6), without losing the main aim of this approach.

Definition 4.1 (Gap$^+$ of a vertex)**.** Let $\boldsymbol{x} \in P^n_{\text{SEP}}$ be a vertex. The *Gap$^+$ of $\boldsymbol{x}$* is

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

This definition is a relaxed version of the previous one, from which we dropped the request that $\boldsymbol{c}$ satisfies $\boldsymbol{c}\boldsymbol{x} = \text{SEP}(\boldsymbol{c})$; thus Gap$^+(\boldsymbol{x}) \geq$ Gap($\boldsymbol{x}$). Even though Gap$^+$ appears to be rougher than Gap on the individual vertices, it pursues the same overall goal [4], that is

$$\alpha_n = \max_{\boldsymbol{x} \text{ vertex of } P^n_{\text{SEP}}} \text{Gap}^+(\boldsymbol{x}) \, .$$

Following the same strategy used for Gap (see [3]), to compute Gap$^+$ we consider the linear problem OPT$^+$.

$$\begin{aligned}
\text{OPT}^+(\boldsymbol{x}) := \ \text{minimize} \quad & \sum_{ij\in E_n} x_{ij}c_{ij} && && (6)\\
\text{subject to:} \quad & c_{ik}+c_{jk}-c_{ij} \ \geq 0 && \forall ij \in E_n,\ k \in V_n \smallsetminus \{i,j\}, && (7)\\
& \sum_{ij\in E_n} t_{ij}c_{ij} \ \geq 1 && \forall \boldsymbol{t} \text{ tour}, && (8)\\
& c_{ij} \geq 0 && \forall ij \in E_n. && (9)
\end{aligned}$$

Constraints (7) force $\boldsymbol{c}$ to be a metric cost; constraints (8) normalize TSP$(\boldsymbol{c}) = 1$; minimizing $\boldsymbol{cx}$ is then equivalent to maximizing $1/\boldsymbol{cx} = \text{TSP}(\boldsymbol{c})/\boldsymbol{cx}$, thus OPT$^+(\boldsymbol{x}) = 1/\text{Gap}^+(\boldsymbol{x})$.

We now state the first immediate result about Gap$^+$, which is the symmetric version of Proposition 4.6.1 of [17]. This lemma guarantees that, for the study of Gap$^+(\boldsymbol{x})$, it is enough to consider only (the metric completion of) metric costs on the support graph $G_{\boldsymbol{x}}$.

Lemma 4.1. *Let $\boldsymbol{x}$ be a vertex of P^n_{SEP} and let $c \in \arg\max Gap^+(\boldsymbol{x})$ (i.e. $\boldsymbol{c}$ is a metric cost that realizes the Gap$^+$ on $\boldsymbol{x}$). Let $\mathbf{c}_{|G_{\boldsymbol{x}}}$ be the restriction of $\mathbf{c}$ on the edges of $G_{\boldsymbol{x}}$, and consider its metric completion $\tilde{\boldsymbol{c}} := (\boldsymbol{c}_{|G_{\boldsymbol{x}}})^*$. Then $\tilde{\boldsymbol{c}} \in \arg\max Gap^+(\boldsymbol{x})$ (i.e. $\tilde{\boldsymbol{c}}$ also is a metric cost that realizes the Gap$^+$ on $\boldsymbol{x}$).*

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

The following lemma is the first result that studies the behavior of Gap$^+$ under the application of the BB-move.

Lemma 4.2. *The BB-move is Gap$^+$-increasing, that is, Gap$^+(\text{BB}(\boldsymbol{x})) \geq$ Gap$^+(\boldsymbol{x})$ for all the vertices $\boldsymbol{x} \in P^n_{SEP}$ for every $n \geq 3$.*

Proof. Let $\boldsymbol{x}_0 \in P^n_{\text{SEP}}$ be a vertex with a 1-edge ab. Let $\boldsymbol{x}_1 \in P^{n+1}_{\text{SEP}}$ be the result of the BB-move applied on ab: we denote w the node added ($V_{n+1} := V_n \cup \{w\}$) and aw, wb the two 1-edges originated by this move. Consider a metric cost $\boldsymbol{c}^0$ which realizes the Gap$^+$ on $\boldsymbol{x}_0$, namely Gap$^+(\boldsymbol{x}_0) = \frac{\text{TSP}(\boldsymbol{c}^0)}{\boldsymbol{c}^0\boldsymbol{x}_0}$. We construct the metric $\boldsymbol{c}^1$ on V_{n+1} adding the node w at distance 0 form b: $c^1_{wb} = 0$, $c^1_{vw} = c^0_{vb}$ for all nodes $v \in V_n \smallsetminus b$ and $c^1_{v_1v_2} = c^0_{v_1v_2}$ for all couples of nodes $v_1, v_2 \in V_n$. Clearly $\boldsymbol{c}^1$ is metric and $\boldsymbol{c}^1\boldsymbol{x}_1 = \boldsymbol{c}^0\boldsymbol{x}_0$.

We first show that TSP$(\boldsymbol{c}^0) \geq$ TSP$(\boldsymbol{c}^1)$. Let $\boldsymbol{t}_0$ be a tour on K_n that is optimal for TSP$(\boldsymbol{c}^0)$: 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 $\boldsymbol{t}_0$ but visits w immediately after b, that is, if the sequence of nodes visited by $\boldsymbol{t}_0$ is $b, v_2, \ldots, v_n$, then the sequence of nodes visited by $\boldsymbol{t}_0^+$ is $b, w, v_2, \ldots, v_n$. It is immediate to verify that $\boldsymbol{c}^1\boldsymbol{t}_0^+ = \boldsymbol{c}^0\boldsymbol{t}_0$: $\boldsymbol{t}_0^+$ uses the same edges of $\boldsymbol{t}_0$, except for aw and wb in place of ab, which still involve the same cost $c^1_{aw} + c^1_{wb} = c^0_{ab} + 0$. Therefore TSP$(\boldsymbol{c}^1) \leq \boldsymbol{c}^1\boldsymbol{t}_0^+ = \boldsymbol{c}^0\boldsymbol{t}_0 =$ TSP$(\boldsymbol{c}^0)$.

On the other hand, TSP$(\boldsymbol{c}^1)$ is always larger than TSP$(\boldsymbol{c}^0)$: given an optimal tour $\boldsymbol{t}_1$ for TSP$(\boldsymbol{c}^1)$, we can always cut out w to get a tour $\boldsymbol{t}_1^-$ with $\boldsymbol{c}^0\boldsymbol{t}_1^- \leq \boldsymbol{c}^1\boldsymbol{t}_1$ by triangle inequality, thus TSP$(\boldsymbol{c}^0) \leq \boldsymbol{c}^0\boldsymbol{t}_1^- \leq \boldsymbol{c}^1\boldsymbol{t}_1 =$ TSP$(\boldsymbol{c}^1)$.

Putting both the inequalities together, we have TSP$(\boldsymbol{c}^1) =$ TSP$(\boldsymbol{c}^0)$, which finally proves Gap$^+(\boldsymbol{x}_1) \geq \frac{\text{TSP}(\boldsymbol{c}^1)}{\boldsymbol{c}^1\boldsymbol{x}_1} = \frac{\text{TSP}(\boldsymbol{c}^0)}{\boldsymbol{c}^0\boldsymbol{x}_0} = \text{Gap}^+(\boldsymbol{x}_0)$. □

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 achieve this goal, two main facts are exploited.

[4] This fact was already mentioned in [3], last note of Section 2.

- Any vertex of $\mathcal{F}_k$ can be obtained starting from one ancestor in $\mathcal{A}_k$ and sequentially applying the BB-move finitely-many times. Moreover, the set of ancestors $\mathcal{A}_k$ is finite.
- Knowing a vertex $\boldsymbol{x}_0$, it is possible to give a bound on $\text{Gap}^+(\text{BB}(\boldsymbol{x}_0))$: we can contain the increase in Gap^+ originated by an iterative application of a BB-move, thus bounding $\text{Gap}^+(\boldsymbol{x}')$ for any successor $\boldsymbol{x}'$ of $\boldsymbol{x}_0$.

The former has already been examined in Section 3; the latter is the core of this section.

To give a lower bound on OPT^+ (which is equivalent to giving an upper bound on Gap^+), we follow this simple idea. Assume that we are given a vertex $\boldsymbol{x}_0$ alongside with an optimal solution of $\text{OPT}^+(\boldsymbol{x}_0)$ and an optimal solution of the dual problem $\mathcal{D}\,\text{OPT}^+(\boldsymbol{x}_0)$. When considering a successor $\boldsymbol{x}'$ of $\boldsymbol{x}$, we may try to produce a feasible dual solution of $\mathcal{D}\,\text{OPT}^+(\boldsymbol{x}')$ starting from the optimal dual solution of $\mathcal{D}\,\text{OPT}^+(\boldsymbol{x}_0)$: succeeding in this task will directly result in a lower bound for $\text{OPT}^+(\boldsymbol{x}')$, as guaranteed by the well known weak duality theorem. Clearly, in this procedure, the key point is not just a matter of finding *any* feasible dual solution, but rather of finding a *good-enough* feasible dual solution, so that the bound obtained is meaningful (e.g., the trivial dual solution of all zeros ultimately gives no bound on Gap^+).

5.1 Dual formulations of the OPT^+ problem

The dual problem of OPT^+ (6) – (9) is the following.

$$
\begin{aligned}
\mathcal{D}\,\text{OPT}^+(\boldsymbol{x}) :=\ \text{maximize}\quad & \sum_{\boldsymbol{t}\text{ tour}} \mu_{\boldsymbol{t}} && && (10)\\
\text{subject to:}\quad & \sum_{k\neq i,j} (-\lambda_{ijk} + \lambda_{ikj} + \lambda_{jki}) + \sum_{\boldsymbol{t}\text{ tour}} t_{ij}\mu_{\boldsymbol{t}} \ \le x_{ij} && \forall ij \in E_n, && (11)\\
& \lambda_{ijk} \ \ge 0 && \forall ij \in E_n,\ k \in V_n \setminus \{i,j\}, && (12)\\
& \mu_{\boldsymbol{t}} \ \ge 0 && \forall \boldsymbol{t} \text{ tour on } K_n. && (13)
\end{aligned}
$$

Notice that the abuse of notation $ij \equiv ji \in E$ induces also $\lambda_{ijk} \equiv \lambda_{jik}$; this fact does not involve the third index: $\lambda_{ijk} \not\equiv \lambda_{kji}$ [5].

When considering a vertex $\boldsymbol{x}$ and one of its successors $\boldsymbol{x}'$, the task of producing a feasible solution of $\mathcal{D}\,\text{OPT}^+(\boldsymbol{x}')$ starting from an optimal solution of $\mathcal{D}\,\text{OPT}^+(\boldsymbol{x})$ is anything but trivial. In what follows, we design an equivalent formulation and make use of it instead: $\mathcal{D}\,\text{OPT}^{\text{II}}$. Its main purpose is to get rid of $\boldsymbol{\lambda}$ variables: this is achieved at the price of considering walks on $G_{\boldsymbol{x}}$ (see Definition 2.3 and Remark 2.1) in place of tours.

$$
\begin{aligned}
\mathcal{D}\,\text{OPT}^{\text{II}}(\boldsymbol{x}) :=\ \text{maximize}\quad & \sum_{\boldsymbol{w}\text{ walk on } G_{\boldsymbol{x}}} \mu_{\boldsymbol{w}} && && (14)\\
\text{subject to:}\quad & \sum_{\boldsymbol{w}\text{ walk on } G_{\boldsymbol{x}}} w_{ij}\mu_{\boldsymbol{w}} \ \le x_{ij} && \forall ij \in E_{\boldsymbol{x}}, && (15)\\
& \mu_{\boldsymbol{w}} \ \ge 0 && \forall \boldsymbol{w} \text{ walk on } G_{\boldsymbol{x}}. && (16)
\end{aligned}
$$

The equivalence of the two formulations $\mathcal{D}\,\text{OPT}^+(\boldsymbol{x})$ and $\mathcal{D}\,\text{OPT}^{\text{II}}(\boldsymbol{x})$ (that is, they have the same optimal value) is stated in the following lemma; the proof is rather technical and it is deferred in Section 8.1.

Lemma 5.1. *$\mathcal{D}\,OPT^+(\boldsymbol{x}) = \mathcal{D}\,OPT^{II}(\boldsymbol{x})$ for every vertex $\boldsymbol{x}$ of P^n_{SEP}.*

5.2 Bounding the Gap on successors

Given the equivalence of the formulations $\mathcal{D}\,\text{OPT}^+$ and $\mathcal{D}\,\text{OPT}^{\text{II}}$, the goal of this section is to derive a feasible solution of $\mathcal{D}\,\text{OPT}^{\text{II}}(\boldsymbol{x}')$ starting from an optimal solution of $\mathcal{D}\,\text{OPT}^{\text{II}}(\boldsymbol{x}_0)$, where $\boldsymbol{x}'$ is a generic successor of $\boldsymbol{x}_0$. This will ultimately give a bound on $\text{Gap}^+(\boldsymbol{x}')$.

Let $\boldsymbol{x}_0$ be a vertex of P^n_{SEP} and let $\boldsymbol{\mu}^0 \in \arg\max \mathcal{D}\,\text{OPT}^{\text{II}}(\boldsymbol{x}_0)$. We consider a 1-edge ab of $\boldsymbol{x}_0$ and apply d consecutive BB-moves (see Definition 2.8), inserting nodes $a_1, \ldots, a_d$ and obtaining, by Theorem 2.2, the vertices $\boldsymbol{x}_k := \text{BB}(\boldsymbol{x}_{k-1},\ a_{k-1}b)$ for all $k = 1, \ldots, d$ (where $a_0 := a$). We aim to design a feasible solution for $\mathcal{D}\,\text{OPT}^{\text{II}}(\boldsymbol{x}_d)$.

[5] With the symbol $\equiv$, we do not mean that two distinct variables have the same value; rather, we mean that the two notations coincide: they denote the same variable.

To obtain an assignment $\boldsymbol{\mu}^d$ for $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x}_d)$ we have to construct walks $\boldsymbol{w}^d$ on $G_{\boldsymbol{x}_d}$ starting from the walks $\boldsymbol{w}^0$ on $G_{\boldsymbol{x}_0}$. In this perspective, we divide the latter into three families, according to how many times the edge ab is traversed: we define $\mathcal{W}^{ab}_m := \{\boldsymbol{w} \text{ walk} \mid w_{ab} = m\}$ for $m = 0, 1, 2$.

- If $\boldsymbol{w}^0 \in \mathcal{W}^{ab}_0$, we construct the following $d+1$ walks on $G_{\boldsymbol{x}_d}$. For each $k = 0, \dots, d$, the walk $\boldsymbol{w}^d_k$ retraces $\boldsymbol{w}^0$ but, when it arrives at a, it deviates to pass through $a_1, a_2, \dots, a_k$ (the nodes inserted via the BB-moves) and back, and when it arrives at b, it deviates to $a_d, a_{d-1}, \dots, a_{k+1}$ and back.
  We set $\mu^d_{\boldsymbol{w}^d_k} := \frac{1}{d+1}\mu^0_{\boldsymbol{w}^0}$ for all $k = 0, \dots, d$.
- If $\boldsymbol{w}^0 \in \mathcal{W}^{ab}_1$, we construct the walk $\boldsymbol{w}^d$ on $G_{\boldsymbol{x}_d}$ as the walk that retraces $\boldsymbol{w}^0$ but replaces the passage on ab by passing through $aa_1, a_1a_2, \dots, a_{d-1}a_d, a_db$.
  We set $\mu^d_{\boldsymbol{w}^d} := \mu^0_{\boldsymbol{w}^0}$.
- If $\boldsymbol{w}^0 \in \mathcal{W}^{ab}_2$, we construct the walk $\boldsymbol{w}^d$ on $G_{\boldsymbol{x}_d}$ as the walk that retraces $\boldsymbol{w}^0$ but replaces the double passage on ab by passing twice through $aa_1, a_1a_2, \dots, a_{d-1}a_d, a_db$.
  We set $\mu^d_{\boldsymbol{w}^d} := \mu^0_{\boldsymbol{w}^0}$.

For all the walks $\boldsymbol{w}^d$ on $G_{\boldsymbol{x}_d}$ not obtained this way, we set $\mu^d_{\boldsymbol{w}^d} := 0$. Figure 3 illustrates the new assignment $\boldsymbol{\mu}^d$ for $d = 2$.

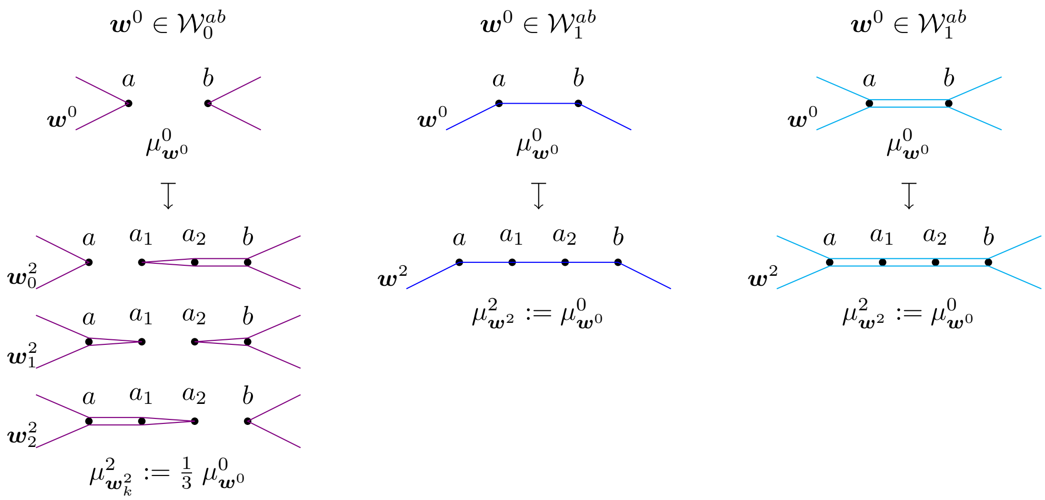

Figure 3: Construction of $\boldsymbol{\mu}^2$ variables for $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x}_2)$ starting from $\boldsymbol{\mu}^0$ variables for $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x}_0)$.

Direct explicit computations (reported in Section 8.2) show that: (i) $\boldsymbol{\mu}^d$ attains the same objective value as $\boldsymbol{\mu}^0$; (ii) $\boldsymbol{\mu}^d$ satisfies all the constraints (15), except for the ones associated to the edges of the 1-path $aa_1 \dots a_db$, where, anyway, the value of the constraint does not exceed a fixed constant $C(\boldsymbol{x}_0, ab)$. The factor $C(\boldsymbol{x}_0, ab)$ depends only on the starting vertex $\boldsymbol{x}_0$ and the edge ab chosen for the application of the BB-moves; it is obtained as

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

We can thus rescale $\boldsymbol{\mu}^d$ by the factor $C(\boldsymbol{x}_0, ab)$ to get the feasible solution $\boldsymbol{\mu}^* := \frac{1}{C(\boldsymbol{x}_0,ab)}\boldsymbol{\mu}^d$; the objective value of $\boldsymbol{\mu}^*$ gets rescaled accordingly and becomes $\frac{1}{C(\boldsymbol{x}_0,ab)}\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x}_0)$. Therefore $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x}_d) \geq \frac{1}{C(\boldsymbol{x}_0,ab)}\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x}_0)$, $\mathcal{D}\,\mathrm{OPT}^{+}(\boldsymbol{x}_d) \geq \frac{1}{C(\boldsymbol{x}_0,ab)}\mathcal{D}\,\mathrm{OPT}^{+}(\boldsymbol{x}_0)$ by Lemma 5.1, $\mathrm{OPT}^{+}(\boldsymbol{x}_d) \geq \frac{1}{C(\boldsymbol{x}_0,ab)}\mathrm{OPT}^{+}(\boldsymbol{x}_0)$ by the strong duality theorem, and finally, by definition of OPT^{+}, $\mathrm{Gap}^{+}(\boldsymbol{x}_d) \leq C(\boldsymbol{x}_0, ab) \cdot \mathrm{Gap}^{+}(\boldsymbol{x}_0)$.

Notice that we expect $C(\boldsymbol{x}_0, ab)$ to be greater than or equal to 1, otherwise we would obtain $\mathrm{Gap}^{+}(\boldsymbol{x}_d) < \mathrm{Gap}^{+}(\boldsymbol{x}_0)$, contradicting Lemma 4.2.

It is crucial to highlight that $C(\boldsymbol{x}_0, ab)$ does not depend on d, hence the bound found is valid for any vertex which is obtained expanding the 1-edge ab of $\boldsymbol{x}_0$ to a 1-path of arbitrary length.

Our purpose now is to generalize the procedure to get a bound for all the successors of $\boldsymbol{x}_0$, obtained by expanding not just one but all its 1-edges. Let $e_1, \ldots, e_p$ be the 1-edges of $\boldsymbol{x}_0$ and consider the successor obtained expanding them to 1-paths with $d_1, \ldots, d_p$ internal nodes. We expand the 1-paths one by one, creating the vertices $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^p$. Notice that here we are using superscript indices to distinguish these iterations from those of the previous argument: Previously, we added nodes one by one to expand a 1-edge to a 1-path; now we are expanding 1-paths one by one to transform $\boldsymbol{x}_0$ into its successor $\boldsymbol{x}'$. Each iteration of this latter argument encapsulates a full expansion of a 1-edge to a 1-path. To be consistent with this notation, we may also use the superscript for the starting vertex $\boldsymbol{x}^0 := \boldsymbol{x}_0$.

The assignments of dual variables $\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^p$ are thus created sequentially starting from $\boldsymbol{\mu}^0$, repeating the same constructions previously described. The normalization of the assignment $\boldsymbol{\mu}^h$ is not performed at each iteration h; rather, the dual variables $\boldsymbol{\mu}^p$ are adjusted only at the very end of the procedure, to give a final $\boldsymbol{\mu}^*$ guaranteed to be feasible.

The constants $C(\boldsymbol{x}^{h-1}, e_h) := 2\sum_{\boldsymbol{w}^{h-1}\in\mathcal{W}_0^{e_h}} \mu^{h-1}_{\boldsymbol{w}^{h-1}} + \sum_{\boldsymbol{w}^{h-1}\in\mathcal{W}_1^{e_h}} \mu^{h-1}_{\boldsymbol{w}^{h-1}} + 2\sum_{\boldsymbol{w}^{h-1}\in\mathcal{W}_2^{e_h}} \mu^{h-1}_{\boldsymbol{w}^{h-1}}$ are computed at each iteration h, and serve as upper bounds on the value that the constraint (15) attains, for the variables $\boldsymbol{\mu}^h$, on the 1-path that expands e_h. To normalize the last assignment $\boldsymbol{\mu}^p$ of the procedure, we retrieve the maximum factor $C^*(\boldsymbol{x}^0) := \max_{h=1,\ldots,p}\{C(\boldsymbol{x}^{h-1}, e_h)\}$ and define $\boldsymbol{\mu}^* := \frac{1}{C^*(\boldsymbol{x}^0)}\boldsymbol{\mu}^p$, which is feasible. Again, since the objective value of the assignments has remained unvaried until the normalization, $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x}^p) \geq \frac{1}{C^*(\boldsymbol{x}^0)}\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x}^0)$, $\mathcal{D}\,\mathrm{OPT}^{+}(\boldsymbol{x}^p) \geq \frac{1}{C^*(\boldsymbol{x}^0)}\mathcal{D}\,\mathrm{OPT}^{+}(\boldsymbol{x}_0)$, $\mathrm{OPT}^{+}(\boldsymbol{x}^p) \geq \frac{1}{C^*(\boldsymbol{x}^0)}\mathrm{OPT}^{+}(\boldsymbol{x}_0)$, and $\mathrm{Gap}^+(\boldsymbol{x}^p) \leq C^*(\boldsymbol{x}^0)\cdot \mathrm{Gap}^+(\boldsymbol{x}_0)$.

There is one last detail of this procedure that needs to be discussed: the dependence of $C(\boldsymbol{x}^{h-1}, e_h)$ on $\boldsymbol{x}^{h-1}$. If our final bound relies on specific steps of the constructions, we fail the goal of finding a bound that holds for *all* the successors of $\boldsymbol{x}^0$. In fact, the dependence of $C^*(\boldsymbol{x}^0)$ from the choice of $d_1, \ldots, d_p$ is only apparent: although the constructions performed at each iteration do depend on the lengths of the 1-paths, the factors $C(\boldsymbol{x}^{h-1}, e_h)$ do not. We would get the same constants even when applying no BB-move at all, namely $d_1 = \cdots = d_p = 0$: this means that, regardless of the order of application of BB-moves during the construction, we may compute $C(\boldsymbol{x}^{h-1}, e_h)$ as if it was calculated at the very first iteration:

$$C(\boldsymbol{x}^{h-1}, e_h) \;=\; C(\boldsymbol{x}^0, e_h) \;=\; 2\sum_{\boldsymbol{w}^0\in\mathcal{W}_0^{e_h}} \mu^0_{\boldsymbol{w}^0} \;+\; \sum_{\boldsymbol{w}^0\in\mathcal{W}_1^{e_h}} \mu^0_{\boldsymbol{w}^0} \;+\; 2\sum_{\boldsymbol{w}^0\in\mathcal{W}_2^{e_h}} \mu^0_{\boldsymbol{w}^0}\,.$$

The explicit computation that proves this fact is presented in Section 8.2.

We are finally able to design the Gap-Bounding (GB) algorithm as Algorithm 1; its main purpose is clarified in the next theorem.

Algorithm 1 Gap-Bounding algorithm

1: INPUT: $\boldsymbol{x} \in P^n_{\text{SEP}}$ vertex
2: Solve $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x})$ and retrieve an optimal assignment of variables $\{\mu_{\boldsymbol{w}}\}_{\boldsymbol{w}\text{ walk on }\boldsymbol{x}}$.
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: $C^*(\boldsymbol{x}) \;\leftarrow\; \max_e\{C(\boldsymbol{x}, e)\}$
7: $\mathrm{Gap}^+(\boldsymbol{x}) \;\leftarrow\; {}^1/_{\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x})}$
8: Return: $C^*(\boldsymbol{x})\cdot\mathrm{Gap}^+(\boldsymbol{x})$

Theorem 5.2. *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 whole section is the proof that $\mathrm{Gap}^+(\boldsymbol{x}') \leq C^*(\boldsymbol{x})\cdot\mathrm{Gap}^+(\boldsymbol{x}) = \mathrm{GB}(\boldsymbol{x})$ for any successor $\boldsymbol{x}'$ of $\boldsymbol{x}$, and we already saw, by definition of Gap^+, that $\mathrm{Gap}^+(\boldsymbol{x}') \geq \mathrm{Gap}(\boldsymbol{x}')$. □

Notice that both the GB algorithm and Theorem 5.2 apply to all generic $\boldsymbol{x}$ vertices, without requiring that they be ancestors.

We conclude this section with the following lemma, which will be of great help in improving computational results.

Lemma 5.3. *Let $\boldsymbol{x} \in P^n_{SEP}$ be a vertex and let $\boldsymbol{x}_0$ be its ancestor. Then $GB(\boldsymbol{x})$ gives a bound for the gap of all the successors $\boldsymbol{x}'$ of $\boldsymbol{x}_0$ (even when $\boldsymbol{x}'$ is not a successor of $\boldsymbol{x}$): $Gap(\boldsymbol{x}') \leq GB(\boldsymbol{x})$.*

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 the successor $\tilde{\boldsymbol{x}}$ of $\boldsymbol{x}_0$ whose 1-paths have $\max(d_1, d'_1), \ldots, \max(d_p, d'_p)$ internal nodes. Then $\tilde{\boldsymbol{x}}$ is a successor of $\boldsymbol{x}'$, hence $\text{Gap}^+(\boldsymbol{x}') \leq \text{Gap}^+(\tilde{\boldsymbol{x}})$ for Lemma 4.2, and $\tilde{\boldsymbol{x}}$ is a successor of $\boldsymbol{x}$ as well, hence $\text{Gap}^+(\tilde{\boldsymbol{x}}) \leq \text{GB}(\boldsymbol{x})$ for the previous theorem. This proves the lemma. ☐

6 Computational results

Our pipeline starts by retrieving the lists of ancestors $\mathcal{A}_k$ for $k = 3, 4, 5, 6$. As finding new ways for generating them directly is out of the scope of this work, we extract them from the lists of vertices already available from [3, 4][6], as explained in Section 3.

For each vertex $\boldsymbol{x} \in \mathcal{A}_k$, the Gap-Bounding algorithm may be applied. We report that, for some ancestors, a straightforward application of the GB algorithm returned a value higher than $4/3$. To improve the algorithm and make it return tighter bounds, we exploit Lemma 5.3 and design an enhanced version, called for simplicity GBe (enhanced). The GBe algorithm, on input a vertex $\boldsymbol{x}$, iteratively applies GB on the successors of $\boldsymbol{x}$ until a desired bound is found, as described in Algorithm 2.

Algorithm 2 Gap-Bounding algorithm, enhanced

1: INPUT: $\boldsymbol{x} \in P^n_{\text{SEP}}$ vertex, α desired bound, I maximum number of iterations
2: $i \leftarrow 0$
3: $\beta \leftarrow \text{GB}(\boldsymbol{x})$
4: **while** $\beta > \alpha$ and $i \leq I$ **do**
5: $\quad i \leftarrow i + 1$
6: $\quad \boldsymbol{x} \leftarrow \text{BB}(\boldsymbol{x}, e)$ (where e is a 1-edge of $\boldsymbol{x}$)
7: $\quad \beta \leftarrow \min\{\beta, \text{GB}(\boldsymbol{x})\}$
8: **end while**
9: Return: β

Theorem 6.1. *For every α desired bound and every I maximum number of iterations., the Gap-Bounding algorithm enhanced, on input a vertex $\boldsymbol{x} \in P^n_{SEP}$, returns a value $GBe(\boldsymbol{x}, \alpha, I)$ which is an upper bound for the gap of all the successors $\boldsymbol{x}'$ of $\boldsymbol{x}$: $Gap(\boldsymbol{x}') \leq GBe(\boldsymbol{x}, \alpha, I)$. Moreover, the bound returned by GBe is tighter or equal to the one returned by GB: $GBe(\boldsymbol{x}, \alpha, I) \leq GB(\boldsymbol{x})$.*

Proof. The theorem simply follows from the observation that the GB algorithm is called at least once along the execution of Algorithm 2 (line 3), and applying Theorem 5.2 and Lemma 5.3. The enhancement $\text{GBe}(\boldsymbol{x}, \alpha, I) \leq \text{GB}(\boldsymbol{x})$ is achieved simply because, in the while-loop (lines 4-8), the returned value β can only decrease (line 7). ☐

Notice that there is no theoretical guarantee that the GB algorithm applied to successors gives better results; indeed, it may happen that it does not. The idea of iterating the applications of BB-moves simply arose from the fact that we thought it reasonable that more "information" (more nodes, more edges, more dual variables) would lead to a more detailed analysis and a tighter bound. Eventually, the application of the GBe algorithm led us to accomplish our objective.

Theorem 6.2. *$Gap(\boldsymbol{x}) \leq \frac{4}{3}$ for all the vertices of $\boldsymbol{x} \in \mathcal{F}_k$ with $k = 3, 4, 5, 6$.*

Proof. The proof is the computational verification that the GBe algorithm applied on the ancestors of $\mathcal{A}_k$ (considered up to isomorphism), with desired bound $4/3$ and a finite number of maximum iterations (in our implementation, 10 was enough), always returns a value smaller than or equal to $4/3$. ☐

The GB algorithm has been implemented in Python, employing the use of the commercial software Gurobi [18] to solve the LPs. Our code is available at `https://github.com/anonym-doubleblind/IG_for_TSP_with_few_edges`. The computation is relatively lightweight and can be easily performed on a standard laptop. More details are available in Section 9.

[6] `https://www.site.uottawa.ca/~sylvia/subtourvertices/index.htm`, last visited 04.06.2025.

7 Conclusion

In this paper, beyond the main result of proving that the integrality gap for the metric symmetric Traveling Salesman Problem is $4/3$ when the support graph of the solution contains at most $n+6$ edges, we introduce a *methodology* that serves two key purposes. Firstly, it significantly narrows down the number of configurations that need to be checked to achieve an exhaustive analysis. Secondly, it enables the derivation of upper bounds on the integrality gap of infinite families of vertices, without relying on approximation algorithms.

A natural next step in this line of research is to construct the set $\mathcal{A}_7$ by exhaustive enumeration. Although this computation may be demanding, we believe that further combinatorial or geometric properties can be identified to reduce the search space to the point where an exhaustive vertex search is accessible by dedicated software (e.g., Polymake [19]).

Another future direction consists in analyzing the GB-algorithm to see whether it is possible to extract from it a constructive approximation algorithm (with approximation factor better than $3/2 - \varepsilon$ [20]) to apply on those instances whose LP solution has few non-zero components.

Finally, it may be interesting to design and investigate new transformations between vertices (similar in spirit to the BB-move), to reduce the surplus of edges over nodes in the support graph without decreasing the integrality gap. If such transformations existed and their repeated application always took us to the vertices of $\mathcal{F}_k$ with $k \leq 6$, the $4/3$-conjecture would be resolved.

8 Auxiliary proofs

In this section, we discuss in detail all the proofs that have been removed from the main argument in Section 5.

8.1 Proof of Lemma 5.1

In Section 5.1, we introduced two different LPs, $\mathcal{D}\,\mathrm{OPT}^+$ (10) – (13) and $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}$ (14) – (16), and we claimed that they have the same objective value (Lemma 5.1); this whole section is devoted to proving this equivalence.

We begin by giving a non-formal interpretation of the equivalence, as it may guide the understanding of the subsequent arguments. As it appears in the definition of the objective function (10), we shall think that the "important" variables of $\mathcal{D}\,\mathrm{OPT}^+$ are the $\boldsymbol{\mu}$, while the $\boldsymbol{\lambda}$ are just "auxiliary". An optimal dual assignment is an assignment of positive weights $\boldsymbol{\mu}$ to the tours, such that, for every edge ij, the sum of the weights of all the tours passing through ij gives exactly x_{ij} (because of complementary slackness). If we only considered the (positive) weights $\boldsymbol{\mu}$, we would obtain the undesired necessary condition that no tour shall use edges outside the support graph (where $x_{ij} = 0$); $\boldsymbol{\lambda}$ variables are meant to "correct" this restriction. In the constraints (11), the variable λ_{ijk} is subtracted from the edge ij and added on ik, jk, as if it is "erasing" the passage of a tour on ij and deviating it to pass through ik and jk instead. In the flavor of this interpretation, the new formulation $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}$ has $\boldsymbol{\mu}$ variables for walks which stay on the support graph (and may traverse the same edge multiple times); $\boldsymbol{\lambda}$ variables are no longer needed, as their corrections are already taken into account along walks.

To prove Lemma 5.1, we need an intermediate formulation, $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}$, which is an extended version of $\mathcal{D}\,\mathrm{OPT}^+$ where we consider walks on the complete graph K_n in place of tours.

$$\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x}) := \text{ maximize } \sum_{\boldsymbol{w}\text{ walk}} \mu_{\boldsymbol{w}} \tag{17}$$

$$\text{subject to: } \sum_{k\neq i,j} (-\lambda_{ijk} + \lambda_{ikj} + \lambda_{jki}) + \sum_{\boldsymbol{w}\text{ walk}} w_{ij}\mu_{\boldsymbol{w}} \leq x_{ij} \quad \forall ij \in E_n, \tag{18}$$

$$\lambda_{ijk} \geq 0 \qquad \forall ij \in E_n,\ k \in V_n \setminus \{i,j\}, \tag{19}$$

$$\mu_{\boldsymbol{w}} \geq 0 \qquad \forall \boldsymbol{w} \text{ walk on } K_n. \tag{20}$$

Remark 8.1. Consider the $\boldsymbol{\mu}$ variables in the three LPs. In $\mathcal{D}\,\mathrm{OPT}^+(\boldsymbol{x})$, they are indexed by tours on K_n; in $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x})$, they are indexed by walks on K_n; in $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x})$, they are indexed by walks on $G_{\boldsymbol{x}}$. In the subsequent proofs, we see how to pass from one formulation to the other. Tours are already special cases of walks on K_n; when we need to "transform" them into walks on $G_{\boldsymbol{x}}$, we substitute edges ij not in $G_{\boldsymbol{x}}$ with paths connecting i and j in $G_{\boldsymbol{x}}$. This is always possible since $G_{\boldsymbol{x}}$ is connected for the subtour-elimination constraints (3). Vice-versa, given a walk on $G_{\boldsymbol{x}}$, we may obtain a tour listing the nodes visited along the walk (having fixed an order and a starting point) and cutting out repeated visits to the same nodes.

Before proving the equivalence of these three formulations (that is, they give the same optimal value), let us address an "exception" we may encounter along the proofs. In the following arguments, we often construct new walks modifying pre-existing ones, adding or removing edges. In principle, it is not excluded that a new walk $\boldsymbol{w}'$ obtained this way may pass through an edge ij more than twice. The problem arises because, as stated in Remark 2.1, we are not considering such walks in our formulations. Whenever this situation occurs, we shall then replace $\boldsymbol{w}'$ with another walk $\boldsymbol{w}''$ obtained removing two copies of ij; in addition, if a variable $\mu_{\boldsymbol{w}'}$ is assigned to $\boldsymbol{w}'$ in the $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}$ problem, we shall instead add the same amount to the variable $\mu_{\boldsymbol{w}''}$ and not consider $\mu_{\boldsymbol{w}'}$ at all. It is crucial to notice that this change preserves the same objective value as the "erroneous" construction. It preserves the feasibility of the assignment as well, since the value of the constraints (15) stays the same on all the edges but ij, on which it eventually decreases (since we dropped two copies of ij in $\boldsymbol{w}''$).

We are now ready to prove the two main lemmas of this section.

Lemma 8.1. *$\mathcal{D}\,OPT^{I}(\boldsymbol{x}) = \mathcal{D}\,OPT^{+}(\boldsymbol{x})$ for every vertex $\boldsymbol{x}$ of P^n_{SEP}.*

Proof. Since the variables of $\mathcal{D}\,\mathrm{OPT}^{+}(\boldsymbol{x})$ are a subset of the variables of $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x})$ ($\mathcal{D}\,\mathrm{OPT}^{+}(\boldsymbol{x})$ is obtained by dropping from $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x})$ all $\mu_{\boldsymbol{w}}$ variables associated to walks $\boldsymbol{w}$ which are not tours), we have $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x}) \geq \mathcal{D}\,\mathrm{OPT}^{+}(\boldsymbol{x})$.

To prove that $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x}) \leq \mathcal{D}\,\mathrm{OPT}^{+}(\boldsymbol{x})$, we show that for any given optimal solution for $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x})$ we can compute a feasible solution for $\mathcal{D}\,\mathrm{OPT}^{+}(\boldsymbol{x})$ of the same value.

Let $(\boldsymbol{\lambda}^0, \boldsymbol{\mu}^0)$ be an optimal solution for $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x})$. Let $\boldsymbol{w}^0$ be a walk on K_n. If $\boldsymbol{w}^0$ is also a tour, then we can leave the variable $\mu_{\boldsymbol{w}^0}$ as it is; otherwise, if $\boldsymbol{w}^0$ is not a tour, in the aim of designing an assignment for $\mathcal{D}\,\mathrm{OPT}^{+}(\boldsymbol{x})$, we have to set $\mu_{\boldsymbol{w}^0}$ to 0. To do so without reducing the optimal value, we also need to adjust a couple of other variables. Let c be a node that is encountered more than once along $\boldsymbol{w}^0$ (it exists since $\boldsymbol{w}^0$ is not a tour), and let a and b be the nodes encountered immediately before and after c (the second time it is traversed). Consider the walk $\boldsymbol{w}^1$ that retraces the steps of $\boldsymbol{w}^0$ but walks through the shortcut a, b instead of a, c, b, as illustrated in Figure 4. We design a new assignment of variables $(\boldsymbol{\lambda}^1, \boldsymbol{\mu}^1)$, introducing, for simplicity, the constant $\delta^0 := \mu^0_{\boldsymbol{w}^0}$:

$$
\begin{aligned}
&\lambda^1_{abc} := \lambda^0_{abc} + \delta^0, && && \mu^1_{\boldsymbol{w}^0} := \mu^0_{\boldsymbol{w}^0} - \delta^0 = 0, &&\\
&\lambda^1_{ijk} := \lambda^0_{ijk} && \forall ijk \neq abc, && \mu^1_{\boldsymbol{w}^1} := \mu^0_{\boldsymbol{w}^1} + \delta^0, &&\\
& && && \mu^1_{\boldsymbol{w}} := \mu^0_{\boldsymbol{w}} && \forall \boldsymbol{w} \neq \boldsymbol{w}^0, \boldsymbol{w}^1.
\end{aligned}
$$

It is immediate to check that the objective value does not change:

$$
\sum_{\boldsymbol{w}} \mu^1_{\boldsymbol{w}} \;=\; \Big(\sum_{\boldsymbol{w} \neq \boldsymbol{w}^0, \boldsymbol{w}^1} \mu^1_{\boldsymbol{w}} \Big) + \mu^1_{\boldsymbol{w}^0} + \mu^1_{\boldsymbol{w}^1} \;=\; \Big(\sum_{\boldsymbol{w} \neq \boldsymbol{w}^0, \boldsymbol{w}^1} \mu^0_{\boldsymbol{w}} \Big) + 0 + (\mu^0_{\boldsymbol{w}^1} + \mu^0_{\boldsymbol{w}^0}) \;=\; \sum_{\boldsymbol{w}} \mu^0_{\boldsymbol{w}} \,.
$$

We now show that $(\boldsymbol{\lambda}^1, \boldsymbol{\mu}^1)$ is feasible for $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}$, that is, all the constraints (18) are satisfied. In fact, the values of the constraints attained by $(\boldsymbol{\lambda}^1, \boldsymbol{\mu}^1)$ is exactly the same as the ones attained by $(\boldsymbol{\lambda}^0, \boldsymbol{\mu}^0)$. We check this fact for the constraint associated with the edge ab; the equalities holds simply by definition of $(\boldsymbol{\lambda}^1, \boldsymbol{\mu}^1)$ and considering that the multiplicities of the walk $\boldsymbol{w}^1$ are given as $w^1_{ab} = w^0_{ab} + 1$, $w^1_{ac} = w^0_{ac} - 1$, $w^1_{cb} = w^0_{cb} - 1$.

$$
\begin{aligned}
&\sum_{k \neq a,b} (-\lambda^1_{abk} + \lambda^1_{akb} + \lambda^1_{bka}) \;+\; \sum_{\boldsymbol{w}} w_{ab} \mu^1_{\boldsymbol{w}} = \\
&\quad = \left(\sum_{k \neq a,b,c} (-\lambda^1_{abk} + \lambda^1_{akb} + \lambda^1_{bka}) \right) + (-\lambda^1_{abc} + \lambda^1_{acb} + \lambda^1_{cba}) \;+\; \left(\sum_{\boldsymbol{w} \neq \boldsymbol{w}^0, \boldsymbol{w}^1} w_{ab} \mu^1_{\boldsymbol{w}} \right) + w^0_{ab} \mu^1_{\boldsymbol{w}^0} + w^1_{ab} \mu^1_{\boldsymbol{w}^1} \\
&\quad = \left(\sum_{k \neq a,b,c} (-\lambda^0_{abk} + \lambda^0_{akb} + \lambda^0_{bka}) \right) + (-(\lambda^0_{abc} + \delta^0) + \lambda^0_{acb} + \lambda^0_{cba}) \;+\; \left(\sum_{\boldsymbol{w} \neq \boldsymbol{w}^0, \boldsymbol{w}^1} w_{ab} \mu^0_{\boldsymbol{w}} \right) + 0 + w^1_{ab} (\mu^0_{\boldsymbol{w}^1} + \delta^0) \\
&\quad = \left(\sum_{k \neq a,b} (-\lambda^0_{abk} + \lambda^0_{akb} + \lambda^0_{bka}) \right) \;-\; \delta^0 \;+\; \left(\sum_{\boldsymbol{w} \neq \boldsymbol{w}^0, \boldsymbol{w}^1} w_{ab} \mu^0_{\boldsymbol{w}} \right) + w^1_{ab} \mu^0_{\boldsymbol{w}^1} + (w^0_{ab} + 1) \delta^0 \\
&\quad = \left(\sum_{k \neq a,b} (-\lambda^0_{abk} + \lambda^0_{akb} + \lambda^0_{bka}) \right) \;-\; \delta^0 \;+\; \left(\sum_{\boldsymbol{w} \neq \boldsymbol{w}^0, \boldsymbol{w}^1} w_{ab} \mu^0_{\boldsymbol{w}} \right) + w^1_{ab} \mu^0_{\boldsymbol{w}^1} + w^0_{ab} \mu^0_{\boldsymbol{w}^1} + \delta^0 \\
&\quad = \left(\sum_{k \neq a,b} (-\lambda^0_{abk} + \lambda^0_{akb} + \lambda^0_{bka}) \right) \;+\; \left(\sum_{\boldsymbol{w}} w_{ab} \mu^0_{\boldsymbol{w}} \right)
\end{aligned}
$$

For ac, bc and all other edges ij, the computation is similar: contributes brought by the new $\boldsymbol{\lambda}^1$ and $\boldsymbol{\mu}^1$ (i.e. $\pm\delta^0$) cancel out and give the same value of the constraint computed for $(\boldsymbol{\lambda}^0, \boldsymbol{\mu}^0)$, as illustrated in Figure 4.

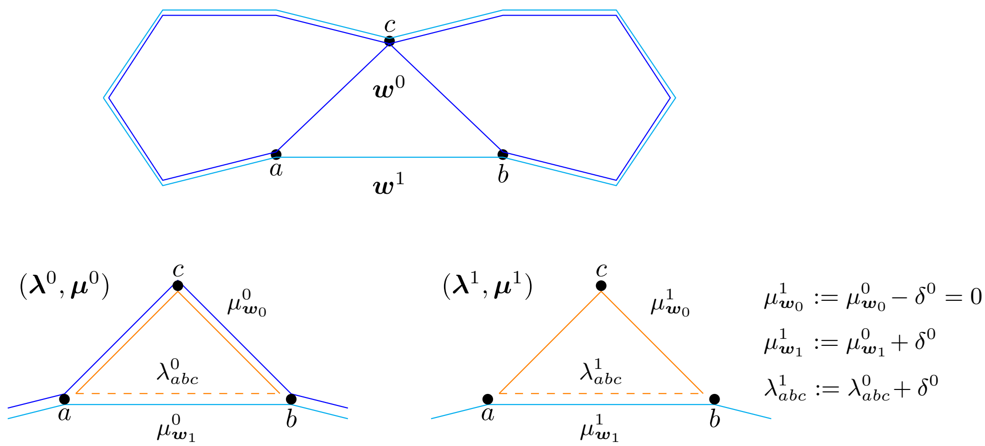

Figure 4: Graphical representation of the proof of Lemma 8.1. Above: the walks $\boldsymbol{w}^0$ and $\boldsymbol{w}^1$: the latter is obtained from the former by cutting c out of the path acb. Below: a detail of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ variables, represented as triangles and walks with solid or dashed edges according to whether the variable is added or subtracted in the corresponding constraint. For both the assignments $(\boldsymbol{\lambda}^0, \boldsymbol{\mu}^0)$ and $(\boldsymbol{\lambda}^1, \boldsymbol{\mu}^1)$, the constraint (18) attains the same value on every edge.

We may apply the same argument on $\boldsymbol{w}^1$ and iterate to get the walks $\boldsymbol{w}^0, \boldsymbol{w}^1, \boldsymbol{w}^2, \ldots, \boldsymbol{w}^s$, until every node is encountered just once along the walk. At the last step, $\boldsymbol{w}^s$ is a tour and all the $\boldsymbol{\mu}$ variables associated to $\boldsymbol{w}^0, \boldsymbol{w}^1, \boldsymbol{w}^2, \ldots, \boldsymbol{w}^{s-1}$ are 0.

Applying this whole procedure for all the walks that are not tours will give a solution $(\boldsymbol{\mu}^*, \boldsymbol{\lambda}^*)$ for $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x})$ which attains the optimal value, and with $\mu^*_{\boldsymbol{w}} = 0$ if $\boldsymbol{w}$ is not a tour; such a solution is in turn a feasible solution for $\mathcal{D}\,\mathrm{OPT}^{+}(\boldsymbol{x})$. This completes the proof. □

Lemma 8.2. *$\mathcal{D}\,OPT^{I}(\boldsymbol{x}) = \mathcal{D}\,OPT^{II}(\boldsymbol{x})$ for every vertex $\boldsymbol{x}$ of P^n_{SEP}.*

Proof. Since the variables of $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x})$ are a subset of the variables of $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x})$ ($\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x})$ is obtained by dropping from $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x})$ all λ_{ijk} variables and all $\mu_{\boldsymbol{w}}$ variables associated to walks $\boldsymbol{w}$ which do not stay on the support graph $G_{\boldsymbol{x}}$), we have $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x}) \geq \mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x})$.

We now need to prove that $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x}) \leq \mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x})$. To achieve this goal, we make use of the following claim, deferring its proof to the end of the main argumentation.

Claim 8.1.1. For any $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ feasible solution for $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x})$ with $\boldsymbol{\lambda} \neq \mathbf{0}$, there exists another feasible solution $(\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}})$ for $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x})$ such that $\sum_{\boldsymbol{w}} \bar{\mu}_{\boldsymbol{w}} = \sum_{\boldsymbol{w}} \mu_{\boldsymbol{w}}$ and $\sum_{ijk} \bar{\lambda}_{ijk} < \sum_{ijk} \lambda_{ijk}$.

Let $(\boldsymbol{\lambda}^0, \boldsymbol{\mu}^0)$ be an optimal solution for $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x})$. Let $P := \{(\boldsymbol{\lambda}, \boldsymbol{\mu}) \mid (\boldsymbol{\lambda}, \boldsymbol{\mu})$ optimal solution for $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x})$, $\sum_{ijk} \lambda_{ijk} \leq \sum_{ijk} \lambda^0_{ijk} \} = \{(\boldsymbol{\lambda}, \boldsymbol{\mu}) \mid (\boldsymbol{\lambda}, \boldsymbol{\mu})$ feasible solution for $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x})$, $\sum_{\boldsymbol{w}} \mu_{\boldsymbol{w}} = \sum_{\boldsymbol{w}} \mu^0_{\boldsymbol{w}}$, $\sum_{ijk} \lambda_{ijk} \leq \sum_{ijk} \lambda^0_{ijk} \}$. P is a bounded polyhedron; hence it is compact. Consider the function $f : P \longrightarrow \mathbb{R}$ which maps $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ into $f(\boldsymbol{\lambda}, \boldsymbol{\mu}) := \sum_{ijk} \lambda_{ijk}$. Since f is a continuous function on the compact polyhedron P, by the Weierstrass theorem, there exists at least one point $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) \in P$ where f attains its minimum value, i.e., $f(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = \min_{(\boldsymbol{\lambda}, \boldsymbol{\mu}) \in P} f(\boldsymbol{\lambda}, \boldsymbol{\mu})$. Claim 8.1.1 guarantees that this minimum is realized with $\boldsymbol{\lambda}^* = \mathbf{0}$. Moreover, for any edge ij not in $E_{\boldsymbol{x}}$, $x_{ij} = 0$ and the constraint (18) becomes $\sum_w w_{ij} \mu^*_{\boldsymbol{w}} \leq 0$, thus $\mu^*_{\boldsymbol{w}} = 0$ for every walk $\boldsymbol{w}$ with $w_{ij} > 0$. This ultimately means that $\lambda^*_{ijk} = 0$ for all ijk and $\mu^*_{\boldsymbol{w}} = 0$ for all the walks $\boldsymbol{w}$ which do not stay on the support graph $G_{\boldsymbol{x}}$; that is, this optimal solution of $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}$ is in turn a feasible solution of $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}$ with the same objective value. □

Proof of Claim 8.1.1. We will divide the argument into two cases.

First case: exists an edge ab such that exists a node c with $\lambda_{abc} > 0$ and exists some walk $\boldsymbol{w}$ with $w_{ab}\mu_{\boldsymbol{w}} > 0$ (the walk traverses ab at least once and the associated variable $\mu_{\boldsymbol{w}}$ is not zero). We aim to "adjust" the assignment by a small quantity δ that leaves λ_{abc} and $\mu_{\boldsymbol{w}}$ grater or equal to 0: $\delta := \min(\lambda_{abc},\ \mu_{\boldsymbol{w}})$. Let $\bar{\boldsymbol{w}}$ be the walk that retraces the steps of $\boldsymbol{w}$ but drops once the edge ab and walks through a, c, b instead. We design a new assignment of variables:

$$\bar{\lambda}_{abc} := \lambda_{abc} - \delta \qquad \bar{\mu}_{\boldsymbol{w}} := \mu_{\boldsymbol{w}} - \delta$$
$$\bar{\lambda}_{ijk} := \lambda_{ijk} \quad \forall ijk \neq abc \qquad \bar{\mu}_{\bar{\boldsymbol{w}}} := \mu_{\bar{\boldsymbol{w}}} + \delta$$
$$\bar{\mu}_{\boldsymbol{w}'} := \mu_{\boldsymbol{w}'} \quad \forall \boldsymbol{w}' \neq \boldsymbol{w}, \bar{\boldsymbol{w}}$$

It is immediate to check that the objective value stays the same and the sum of the $\boldsymbol{\lambda}$ variables decreases:

$$\sum_{\boldsymbol{w}'} \bar{\mu}_{\boldsymbol{w}'} = \Big(\sum_{\boldsymbol{w}' \neq \boldsymbol{w}, \bar{\boldsymbol{w}}} \bar{\mu}_{\boldsymbol{w}'} \Big) + \bar{\mu}_{\boldsymbol{w}} + \bar{\mu}_{\bar{\boldsymbol{w}}} = \Big(\sum_{\boldsymbol{w}' \neq \boldsymbol{w}, \bar{\boldsymbol{w}}} \mu_{\boldsymbol{w}'} \Big) + (\mu_{\boldsymbol{w}} - \delta) + (\mu_{\bar{\boldsymbol{w}}} + \delta) = \sum_{\boldsymbol{w}'} \mu_{\boldsymbol{w}'}$$
$$\sum_{ijk} \bar{\lambda}_{ijk} = \Big(\sum_{ijk \neq abc} \bar{\lambda}_{ijk} \Big) + \bar{\lambda}_{abc} = \Big(\sum_{ijk \neq abc} \lambda_{ijk} \Big) + (\lambda_{abc} - \delta) = \sum_{ijk} \lambda_{ijk} - \delta$$

Showing that the new assignment $(\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{w}})$ is feasible for $\mathcal{D}\,\mathrm{OPT}^{\mathrm{I}}(\boldsymbol{x})$, as in the proof of Lemma 8.1, is just a matter of expand the new variables and check that the new weights and multiplicities balance out to give the same value as $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ for all the constraints.

Second case: for every edges ij, all the variables λ_{ijk} are 0 or all the walks $\boldsymbol{w}$ have $w_{ij}\mu_{\boldsymbol{w}} = 0$. Then, for every edge ij, or $\sum_{\boldsymbol{w}} w_{ij}\mu_{\boldsymbol{w}} \leq \sum_{\boldsymbol{w}} w_{ij}\mu_{\boldsymbol{w}} + \sum_{k \neq i,j}(-0 + \lambda_{ikj} + \lambda_{jki}) = \sum_{\boldsymbol{w}} w_{ij}\mu_{\boldsymbol{w}} + \sum_{k \neq i,j}(-\lambda_{ijk} + \lambda_{ikj} + \lambda_{jki}) \leq x_{ij}$, or $\sum_{\boldsymbol{w}} w_{ij}\mu_{\boldsymbol{w}} = 0 \leq x_{ij}$. This proves that we can remove all the $\boldsymbol{\lambda}$ variables (i.e., set them to 0) to obtain the new assignment $(\boldsymbol{0}, \boldsymbol{\mu})$, which is feasible and preserves all the $\boldsymbol{\mu}$ variables, thus preserving the objective value.

Notice that this completes the proof since the two cases analysed are the logical negation of each other. Writing the two clauses with formal predicate logic, it is immediate to check that $\neg$ (First case) $\Leftrightarrow$ (Second case), as we have:

$$\text{First case:} \quad \exists ab \in E_n : (\ \exists c \in V_n : \lambda_{abc} > 0 \ \wedge\ \exists \boldsymbol{w} \text{ walk on } K_n : w_{ab}\mu_{\boldsymbol{w}} > 0\ )\,,$$
$$\text{Second case:} \quad \forall ij \in E_n : (\ \forall k \in V_n : \lambda_{ijk} = 0 \ \vee\ \forall \boldsymbol{w} \text{ walk on } K_n : w_{ij}\mu_{\boldsymbol{w}} = 0\ )\,.$$

□

Finally, Lemma 5.1 trivially follows, concatenating the equalities of Lemma 8.1 and Lemma 8.2.

8.2 Proof of the soundness of the GB algorithm

In Section 5.2 we introduced the GB algorithm and proved its soundness relying upon claims that we only stated. In this section we provide proofs of all these technical statements, ultimately completing the demonstration of the validity of Algorithm 1.

Recall how we designed the GB algorithm. Starting from a vertex $\boldsymbol{x}_0$ alongside with an optimal solution $\boldsymbol{\mu}^0$ for the dual problem $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x}_0)$, we considered the successor $\boldsymbol{x}_d$ obtained expanding a 1-edge ab of $\boldsymbol{x}_0$ to a 1-path with d internal nodes and we constructed an assignment $\boldsymbol{\mu}^d$ for $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x}_d)$. To do so, for every walk $\boldsymbol{w}^0$ on $G_{\boldsymbol{x}_0}$, depending on its multiplicity on the edge ab, we considered different walks on $G_{\boldsymbol{x}_d}$: for $\boldsymbol{w}^0 \in \mathcal{W}_0^{ab}$, we considered $d{+}1$ walks $\boldsymbol{w}_k^d$ on $G_{\boldsymbol{x}_d}$ and set $\mu^d_{\boldsymbol{w}_k^d} := \frac{1}{d+1}\mu^0_{\boldsymbol{w}^0}$ for all $k = 0, \ldots, d$; for $\boldsymbol{w}^0 \in \mathcal{W}_1^{ab}$ or $\boldsymbol{w}^0 \in \mathcal{W}_2^{ab}$, we considered one walk $\boldsymbol{w}^d$ on $G_{\boldsymbol{x}_d}$ and set $\mu^d_{\boldsymbol{w}^d} := \mu^0_{\boldsymbol{w}^0}$ (see Section 5.2). We need to prove that the new assignment $\boldsymbol{\mu}^d$ satisfies the properties specified in the following claim.

Claim 8.2.1. (i) $\boldsymbol{\mu}^d$ attains the same objective value (in $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x}_d)$) as $\boldsymbol{\mu}^0$ (in $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x}_0)$); (ii) $\boldsymbol{\mu}^d$ satisfies all the constraints (15), except for the ones associated to the edges of the 1-path $aa_1 \ldots a_d b$, where, anyway, the value of the constraint does not exceed a fixed constant $C(\boldsymbol{x}_0, ab)$.

Proof. The proof of the first fact (i) is the following straightforward computation:

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

To prove the second fact (ii), we consider the edges of $E_{\boldsymbol{x}_d}$ and study the value of the associated constraints (15).

For edges ij of $E_{\boldsymbol{x}^d}$ not in the 1-path $aa_1 \dots b$, the multiplicity of the new walks $\boldsymbol{w}^d$ are the same of the ones they were originated from, i.e. $w^d_{ij} = w^0_{ij}$, and the constraint is still satisfied:

$$\begin{aligned}
\sum_{\boldsymbol{w}^d \text{ walk on } G_{\boldsymbol{x}_d}} w^d_{ij}\, \mu^d_{\boldsymbol{w}^d} &= \sum_{\boldsymbol{w}^0 \in \mathcal{W}^{ab}_0} \Big(\sum_{k=0}^{d} (w^d_k)_{ij}\, \mu^d_{\boldsymbol{w}^d_k}\Big) + \sum_{\boldsymbol{w}^0 \in \mathcal{W}^{ab}_1} w^d_{ij}\, \mu^d_{\boldsymbol{w}^d} + \sum_{\boldsymbol{w}^0 \in \mathcal{W}^{ab}_2} w^d_{ij}\, \mu^d_{\boldsymbol{w}^d} \\
&= \sum_{\boldsymbol{w}^0 \in \mathcal{W}^{ab}_0} \Big((d{+}1)\; (w^0_{ij}\, \frac{1}{d{+}1} \mu^0_{\boldsymbol{w}^0})\Big) + \sum_{\boldsymbol{w}^0 \in \mathcal{W}^{ab}_1} w^0_{ij}\, \mu^0_{\boldsymbol{w}^0} + \sum_{\boldsymbol{w}^0 \in \mathcal{W}^{ab}_2} w^0_{ij}\, \mu^0_{\boldsymbol{w}^0} \\
&= \sum_{\boldsymbol{w}^0 \text{ walk on } G_{\boldsymbol{x}_0}} w^0_{ij}\, \mu^0_{\boldsymbol{w}^0} \;\le\; (x_0)_{ij} \;=\; (x_d)_{ij} \;.
\end{aligned}$$

It only remains to study the edges of the 1-path and bound from above the value of the constraints associated with them. Consider, for instance, aa_1 (analogous computations apply to the other edges):

$$\begin{aligned}
\sum_{\boldsymbol{w}^d \text{ walk on } G_{\boldsymbol{x}_d}} w^d_{aa_1}\, \mu^d_{\boldsymbol{w}^d} \;&=\; \sum_{\boldsymbol{w}^0 \in \mathcal{W}^{ab}_0} \Big(\Big(\sum_{k \neq 0} (w^d_k)_{aa_1}\, \mu^d_{\boldsymbol{w}^d_k}\Big) + (w^d_0)_{aa_1}\, \mu^d_{\boldsymbol{w}^d_0}\Big) + \sum_{\boldsymbol{w}^0 \in \mathcal{W}^{ab}_1} w^d_{aa_1}\, \mu^d_{\boldsymbol{w}^d} + \sum_{\boldsymbol{w}^0 \in \mathcal{W}^{ab}_2} w^d_{aa_1}\, \mu^d_{\boldsymbol{w}^d} \\
&=\; \sum_{\boldsymbol{w}^0 \in \mathcal{W}^{ab}_0} \Big(2\, \frac{d}{d{+}1} + 0\Big) \mu^0_{\boldsymbol{w}^0} + \sum_{\boldsymbol{w}^0 \in \mathcal{W}^{ab}_1} 1\, \mu^0_{\boldsymbol{w}^0} + \sum_{\boldsymbol{w}^0 \in \mathcal{W}^{ab}_2} 2\, \mu^0_{\boldsymbol{w}^0} \\
&\le\; 2 \sum_{\boldsymbol{w}^0 \in \mathcal{W}^{ab}_0} \mu^0_{\boldsymbol{w}^0} \;+\; \sum_{\boldsymbol{w}^0 \in \mathcal{W}^{ab}_1} \mu^0_{\boldsymbol{w}^0} \;+\; 2 \sum_{\boldsymbol{w}^0 \in \mathcal{W}^{ab}_2} \mu^0_{\boldsymbol{w}^0} \;.
\end{aligned}$$

The value of the constraint is bounded from above by a quantity that does not depend on d. $\square$

We denote by $C(\boldsymbol{x}_0, ab)$ the right-hand side of the last inequality of the previous proof, which serves as upper bound on the value that the variables $\boldsymbol{\mu}^d$ attain for the constraints (15) associated to the 1-edges of the 1-path $aa_1 \dots a_d b$:

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

It is then possible to rescale $\boldsymbol{\mu}^d$ and get a feasible assignment for $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x}^d)$: $\boldsymbol{\mu}^* := \frac{1}{C(\boldsymbol{x}_0, ab)} \boldsymbol{\mu}^d$.

Notice that $C(\boldsymbol{x}_0, ab)$ depends only on $\boldsymbol{x}_0$ and the chosen 1-edge ab to expand. It does not depend on d: we got rid of this dependency in the last inequality, when bounding $\frac{d}{d+1}$ by 1.

Proceeding with the argumentation of Section 5.2, we expanded all the 1-edges $e_1, \dots, e_p$ of $\boldsymbol{x}^0 (:= \boldsymbol{x}_0)$ to 1-paths of arbitrary length, creating sequentially the vertices $\boldsymbol{x}^1, \dots, \boldsymbol{x}^p$ and the corresponding assignments $\boldsymbol{\mu}^1, \dots, \boldsymbol{\mu}^p$. Also the factors $C(\boldsymbol{x}^{h-1}, e_h)$ were built sequentially at each step $h = 1, \dots, p$ of the construction:

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

To finally prove the soundness of the GB algorithm, it only remains to show that these constants $C(\boldsymbol{x}^{h-1}, e_h)$ do not depend on the order in which the 1-edges are expanded: they can be computed directly from the starting vertex $\boldsymbol{x}^0$.

Claim 8.2.2. $C(\boldsymbol{x}^{h-1}, e_h) = C(\boldsymbol{x}^0, e_h)$.

Before digging into the proof of the claim, we introduce a notation that prevents ambiguities. For $m = 0, 1, 2$, we use $\mathrm{E}^{e_h}_m(\boldsymbol{w}^{h-1})$ (eventually with subscript k when $m = 0$) to denote the new walks obtained when extending $\boldsymbol{w}^{h-1}$ ("E" for "extend") to pass through all the new nodes added in the expansion of e_h to a 1-path with d_h internal nodes (see Section 5.2). According to whether $\boldsymbol{w}^{h-1} \in \mathcal{W}^{e_h}_0$, $\boldsymbol{w}^{h-1} \in \mathcal{W}^{e_h}_1$, or $\boldsymbol{w}^{h-1} \in \mathcal{W}^{e_h}_2$, the new assignment of $\boldsymbol{\mu}$ variables is defined as $\mu^h_{\mathrm{E}^{e_h}_0(\boldsymbol{w}^{h-1})_k} := \frac{1}{d_h+1} \mu^{h-1}_{\boldsymbol{w}^{h-1}}$ for all $k = 0, \dots, d_h$, $\mu^h_{\mathrm{E}^{e_h}_1(\boldsymbol{w}^{h-1})} := \mu^{h-1}_{\boldsymbol{w}^{h-1}}$, or $\mu^h_{\mathrm{E}^{e_h}_2(\boldsymbol{w}^{h-1})} := \mu^{h-1}_{\boldsymbol{w}^{h-1}}$.

We also introduce an operator that mimics an *if* statement: we use $\langle condition \rangle$, which has value 1 if $condition$ is true and 0 otherwise. $C(\boldsymbol{x}^{h-1}, e_h)$ can thus be rewritten as:

$$C(\boldsymbol{x}^{h-1}, e_h) = \sum_{\boldsymbol{w} \text{ walk on } G_{\boldsymbol{x}^{h-1}}} \Big(2\, \langle \boldsymbol{w}^{h-1} \in \mathcal{W}^{e_h}_0 \rangle \;+\; \langle \boldsymbol{w}^{h-1} \in \mathcal{W}^{e_h}_1 \rangle \;+\; 2\, \langle \boldsymbol{w}^{h-1} \in \mathcal{W}^{e_h}_2 \rangle \Big) \mu^{h-1}_{\boldsymbol{w}} \;.$$

Proof of Claim 8.2.2. We prove the claim using a recursive strategy: we show that the constant $C(\boldsymbol{x}^{h-1}, e_h)$ can be computed at step $h-1$ instead of step h, namely $C(\boldsymbol{x}^{h-1}, e_h) = C(\boldsymbol{x}^{h-2}, e_h)$. Applying the same argument until the step 0 is reached proves the desired statement.

For the sake of simplifying the notation, we only consider the second step $h = 2$; the same argument may be generalized for the generic step $h \in \{2, \dots, p\}$.

We start by observing that extending a walk on the 1-edge e_1 does not affect the multiplicity on e_2. Therefore, for all $m = 0, 1, 2$, we have the equivalent conditions:

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

The following computation shows how to 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}^1 \text{ walk on } G_{\boldsymbol{x}^1}} \Big(2\,\langle \boldsymbol{w}^1 \in \mathcal{W}_0^{e_2}\rangle \,+\, \langle \boldsymbol{w}^1 \in \mathcal{W}_1^{e_2}\rangle \,+\, 2\,\langle \boldsymbol{w}^1 \in \mathcal{W}_2^{e_2}\rangle\Big)\mu^1_{\boldsymbol{w}^1} \\ = & \sum_{\boldsymbol{w}^0 \in \mathcal{W}_0^{e_1}} \Bigg(\sum_{k=0}^{d_1} \Big(2\,\langle \mathrm{E}_0^{e_1}(\boldsymbol{w}^0)_k \in \mathcal{W}_0^{e_2}\rangle + \langle \mathrm{E}_0^{e_1}(\boldsymbol{w}^0)_k \in \mathcal{W}_1^{e_2}\rangle + 2\,\langle \mathrm{E}_0^{e_1}(\boldsymbol{w}^0)_k \in \mathcal{W}_2^{e_2}\rangle \Big)\mu^1_{\mathrm{E}_0^{e_1}(\boldsymbol{w}^0)_k}\Bigg) \\ & + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_1^{e_1}} \Big(2\,\langle \mathrm{E}_1^{e_1}(\boldsymbol{w}^0) \in \mathcal{W}_0^{e_2}\rangle \,+\, \langle \mathrm{E}_1^{e_1}(\boldsymbol{w}^0) \in \mathcal{W}_1^{e_2}\rangle \,+\, 2\,\langle \mathrm{E}_1^{e_1}(\boldsymbol{w}^0) \in \mathcal{W}_2^{e_2}\rangle\Big)\mu^1_{\mathrm{E}_1^{e_1}(\boldsymbol{w}^0)} \\ & + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_2^{e_1}} \Big(2\,\langle \mathrm{E}_2^{e_1}(\boldsymbol{w}^0) \in \mathcal{W}_0^{e_2}\rangle \,+\, \langle \mathrm{E}_2^{e_1}(\boldsymbol{w}^0) \in \mathcal{W}_1^{e_2}\rangle \,+\, 2\,\langle \mathrm{E}_2^{e_1}(\boldsymbol{w}^0) \in \mathcal{W}_2^{e_2}\rangle\Big)\mu^1_{\mathrm{E}_2^{e_1}(\boldsymbol{w}^0)} \\ = & \sum_{\boldsymbol{w}^0 \in \mathcal{W}_0^{e_1}} \Bigg(\sum_{k=0}^{d_1} \Big(2\,\langle \boldsymbol{w}^0 \in \mathcal{W}_0^{e_2}\rangle \,+\, \langle \boldsymbol{w}^0 \in \mathcal{W}_1^{e_2}\rangle \,+\, 2\,\langle \boldsymbol{w}^0 \in \mathcal{W}_2^{e_2}\rangle \Big)\frac{1}{d_1+1}\mu^0_{\boldsymbol{w}^0}\Bigg) \\ & + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_1^{e_1}} \Big(2\,\langle \boldsymbol{w}^0 \in \mathcal{W}_0^{e_2}\rangle \,+\, \langle \boldsymbol{w}^0 \in \mathcal{W}_1^{e_2}\rangle \,+\, 2\,\langle \boldsymbol{w}^0 \in \mathcal{W}_2^{e_2}\rangle\Big)\mu^0_{\boldsymbol{w}^0} \\ & + \sum_{\boldsymbol{w}^0 \in \mathcal{W}_2^{e_1}} \Big(2\,\langle \boldsymbol{w}^0 \in \mathcal{W}_0^{e_2}\rangle \,+\, \langle \boldsymbol{w}^0 \in \mathcal{W}_1^{e_2}\rangle \,+\, 2\,\langle \boldsymbol{w}^0 \in \mathcal{W}_2^{e_2}\rangle\Big)\mu^0_{\boldsymbol{w}^0} \\ = & \sum_{\boldsymbol{w}^0 \text{ walk on } G_{\boldsymbol{x}^0}} \Big(2\,\langle \boldsymbol{w}^0 \in \mathcal{W}_0^{e_2}\rangle \,+\, \langle \boldsymbol{w}^0 \in \mathcal{W}_1^{e_2}\rangle \,+\, 2\,\langle \boldsymbol{w}^0 \in \mathcal{W}_2^{e_2}\rangle\Big)\mu^0_{\boldsymbol{w}^0} \\ = & \; C(\boldsymbol{x}^0, e_2) \end{aligned}$$

□

9 Implementation details for the GB algorithm

This section enriches the content of Section 6, discussing implementation details of the GB algorithm.

9.1 Details on the computation of $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}$

First, we briefly delve into the procedure of solving $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x})$ (14) – (16), used as a subroutine of Algorithm 1.

It is not computationally efficient to solve the dual problem $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x})$ with all variables included from the outset. Instead, we adopt a well-known *row generation* approach to the primal: constraints are added incrementally, one at a time, until it can be proven that all remaining constraints are implied by those already included. This technique is analogous to the separation of subtour elimination constraints in the standard approach to solving the TSP (See, e.g., [21]). The LP to solve becomes $\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x})$, the primal of $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x})$: variables (respectively constraints) of the former correspond to constraints (respectively variables) of the latter.

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

The constraints associated with the walks on $G_{\boldsymbol{x}}$ are added to the model sequentially, choosing the "most restrictive", i.e., the ones corresponding to the walks of minimal cost. These may be found by solving the following ILP, which is the version of the graph-TSP with costs on the edges.

$$
\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{ij \in E_{\boldsymbol{x}}} c_{ij} w_{ij} & \\
\text{subject to:} & \displaystyle\sum_{ij \in \delta(v)} w_{ij} \ = 2\, d_v & \forall v \in V_n, \\
& \displaystyle\sum_{ij \in \delta(S)} w_{ij} \ \geq 2 & \forall S \in \mathcal{S}, \\
& 0 \leq \ w_{ij} \ \leq 2 & \forall ij \in E_{\boldsymbol{x}}, \\
& w_{ij} \ \text{ integer} & \forall ij \in E_{\boldsymbol{x}}, \\
& d_v \ \text{ integer} & \forall v \in V_n.
\end{array}
$$

Once the solution is proven to be optimal, we formulate $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}$ using only those variables $\mu_{\boldsymbol{w}}$ corresponding to tight constraints of $\mathrm{OPT}^{\mathrm{II}}$. In virtue of the theorem of complementary slackness (see, e.g., [22]), all remaining dual variables can be safely set to zero. At this point, the objective values of the primal and dual formulations coincide, and we recover the non-zero $\mu_{\boldsymbol{w}}$ values required for the estimates $C(\boldsymbol{x}, e)$.

9.2 Details on the computation of the GBe algorithm

In Algorithm 2, we have a certain degree of freedom in applying the BB-move (line 6). Given a vertex $\boldsymbol{x}$ with $\mathrm{GB}(\boldsymbol{x}) > 4/3$, we found it reasonable to consider the successor $\boldsymbol{x}' = \mathrm{BB}(\boldsymbol{x}, e)$ 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). Moreover, the procedure of solving $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x}')$ (subroutine of Algorithm 1, line 7) may benefit from the previous solution of $\mathcal{D}\,\mathrm{OPT}^{\mathrm{II}}(\boldsymbol{x})$, already at our disposal: instead of restarting the row 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 5.2, and initialize the model with the associated constraints.

9.3 Details on the obtained upper bounds

For each $k = 3, 4, 5, 6$, by running Algorithm 2 on all the ancestors of $\mathcal{A}_k$ (up to isomorphism), we were able to obtain the upper bound of $4/3$ for the Gap of all the vertices of $\mathcal{F}_k$. Table 1 reports the number of ancestors in $\mathcal{A}_k$ (up to isomorphism), the upper bound found for the Gap on vertices in $\mathcal{F}_k$ (namely, the maximum of all the values returned by the GBe algorithm applied on the ancestors in $\mathcal{A}_k$), and the maximum number of additional iterations of the GB algorithm needed in the execution of GBe.

k	$\lvert\mathcal{A}_k\rvert$	upper bound on Gap	max additional iterations of GB in GBe
3	1	$4/3$	0
4	5	$4/3$	2
5	44	$4/3$	5
6	715	$4/3$	10

Table 1: Computational results of the application of the Gap-Bounding algorithm.

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.

References

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

[2] Laurence A. Wolsey. Heuristic analysis, linear programming and branch and bound. In V. J. Rayward-Smith, editor, *Combinatorial Optimization II*, pages 121–134. Springer Berlin Heidelberg, Berlin, Heidelberg, 1980.

[3] Geneviève Benoit and Sylvia Boyd. Finding the Exact Integrality Gap for Small Traveling Salesman Problems. *Mathematics of Operations Research*, 33:921–931, 11 2008.

[4] Sylvia Boyd and Paul Elliott-Magwood. Structure of the Extreme Points of the Subtour Elimination Polytope of the STSP. In *RIMS Kokyuroku Bessatsu*, pages 33–47, 12 2010.

[5] Frans Schalekamp, David P Williamson, and Anke van Zuylen. 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.

[6] Sylvia Boyd and Robert Carr. Finding low cost TSP and 2-matching solutions using certain half-integer subtour vertices. *Discrete Optimization*, 8(4):525–539, 2011.

[7] Tobias Mömke and Ola Svensson. Removing and Adding Edges for the Traveling Salesman Problem. *J. ACM*, 63(1):2:1–2:28, February 2016.

[8] Anna R Karlin, Nathan Klein, and Shayan Oveis Gharan. An improved approximation algorithm for TSP in the half integral case. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 28–39, 2020.

[9] Anupam Gupta, Euiwoong Lee, Jason Li, Marcin Mucha, Heather Newman, and Sherry Sarkar. Matroid-based TSP rounding for half-integral solutions. *Mathematical Programming*, 206(1):541–576, 2024.

[10] Sylvia Boyd and András Sebő. The salesman's improved tours for fundamental classes. *Mathematical Programming*, 186(1):289–307, 2021.

[11] Billy Jin, Nathan Klein, and David P. Williamson. A 4/3-approximation algorithm for half-integral cycle cut instances of the TSP. *Mathematical Programming*, February 2025.

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

[13] Stefan Hougardy and Xianghui Zhong. Hard to solve instances of the Euclidean Traveling Salesman Problem. *Mathematical Programming Computation*, 13:51–74, 2021.

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

[15] Eleonora Vercesi, Stefano Gualandi, Monaldo Mastrolilli, and Luca Maria Gambardella. On the generation of metric TSP instances with a large integrality gap by branch-and-cut. *Mathematical Programming Computation*, 15(2):389–416, 2023.

[16] Gerard Cornuejols, Jean Fonlupt, and Denis Naddef. The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical Programming*, 33:1–27, 09 1985.

[17] Paul Elliott-Magwood. *The Integrality Gap of the Asymmetric Traveling Salesman Problem*. PhD thesis, University of Ottawa, 2008.

[18] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.

[19] Ewgenij Gawrilow and Michael Joswig. polymake: a framework for analyzing convex polytopes. In *Polytopes—Combinatorics and Computation (Oberwolfach, 1997)*, volume 29 of *DMV Seminar*, pages 43–73. Birkhäuser, Basel, 2000.

[20] Anna R Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation algorithm for metric TSP. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 32–45, 2021.

[21] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer programming models*. Springer, 2014.

[22] David Gale, Harold W. Kuhn, and Albert W. Tucker. Linear Programming and the Theory of Games. In Tjalling C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 317–329. Wiley, New York, NY, 1951.