Fractional Decomposition Tree Algorithm: A tool for studying the integrality gap of Integer Programs

Robert Carr* Arash Haddadan† Cynthia A. Phillips‡

August 11, 2020

Abstract

We present a new algorithm, Fractional Decomposition Tree (FDT) for finding a feasible solution for an integer program (IP) where all variables are binary. FDT runs in polynomial time and is guaranteed to find a feasible integer solution provided the integrality gap is bounded. The algorithm gives a construction for Carr and Vempala's theorem that any feasible solution to the IP's linear-programming relaxation, when scaled by the instance integrality gap, dominates a convex combination of feasible solutions. FDT is also a tool for studying the integrality gap of IP formulations. We demonstrate that with experiments studying the integrality gap of two problems: optimally augmenting a tree to a 2-edge-connected graph and finding a minimum-cost 2-edge-connected multi-subgraph (2EC). We also give a simplified algorithm, Dom2IP, that more quickly determines if an instance has an unbounded integrality gap. We show that FDT's speed and approximation quality compare well to that of feasibility pump on moderate-sized instances of the vertex cover problem. For a particular set of hard-to-decompose fractional 2EC solutions, FDT always gave a better integer solution than the best previous approximation algorithm (Christofides).

*University of New Mexico bobcarr@unm.edu. This material is based upon research supported in part by the U.S. Office of Naval Research under award number N00014-18-1-2099.

†University of Virginia ahaddada@virginia.edu. This work was mainly done when this author was a graduate student at Carnegie Mellon University.

‡Sandia National Laboratories caphill@sandia.gov. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

1 Introduction

In this paper we focus on finding feasible solutions to binary Integer Linear Programs (IP). Informally, an integer program is the optimization of a linear objective function subject to linear constraints, where the variables must take integer values. Binary variables represent yes/no decisions. Integer Programming (and more generally Mixed Integer Linear Programming (MILP)) can model many practical optimization problems including scheduling, logistics and resource allocation.

It is NP-hard even to determine if an IP instance has a feasible solution [GJ90]. Given its practical importance, however, there are many commercial (e.g. CPLEX, GUROBI, XPRESS) and free (e.g. CBC) solvers that for specific IP instances can often find solutions that are optimal within a given tolerance. Still we have formulated moderate-sized IP instances that no commercial solver can currently solve. Thus there is value in heuristics to find feasible solutions for general IP instances (see e.g. [HT17]). These heuristics, such as the popular Feasibility pump algorithm [FGL05, FS09], are often effective and fast in practice. However, the heuristics can sometimes fail to find a feasible solution. Moreover, these heuristics do not provide any bounds on the quality of the solution they find.

A major tool for finding feasible solutions for discrete optimization problems expressed as IPs is the *linear-programming (LP) relaxation* for the IP formulation. This is a new problem created by relaxing the integrality constraints for an IP instance, allowing the variables to take continuous (rational) values. Linear programs can be solved in polynomial time. The objective value of the linear-programming relaxation provides a bound (lower bound for a minimization problem and upper bound for a maximization problem) on the optimal solution to the IP instance. The solutions can also provide some useful global structure, even though the fractional values might not be directly meaningful.

LP-based approximation algorithms use LP relaxations to find provably good approximate feasible solutions to IP problems in polynomial time. At the highest level, they involve solving the LP relaxation, using special structure from the problem to find a feasible solution, and proving that the objective value of the solution is no more than C times worse than the bound from the LP relaxation. The approximation factor C can be a constant or depend on the input parameters of the IP, e.g. $O(\log(n))$ where n is the number of variables in the formulation of the IP (the dimension of the problem).

There is an inherent limit to how small C can be for a given IP. The integrality gap for an IP instance is the ratio of the best integer solution to the best solution of the LP relaxation. Any LP-based approximation cannot have an approximation factor C smaller than the integrality gap because there is no feasible solution with an objective value better than a factor of C worse than the optimal solution of the LP relaxation.

If the integrality gap for an IP formulation is large, it is sometimes possible to add families of constraints to the formulation to reduce the integrality gap. These constraints are redundant for the integer problem, but can make some fractional solutions no longer feasible for the LP. These families of constraints (cuts) can have exponential size (number of constraints) as long as we can provide a polynomial-time separation algorithm. A separation algorithm for a family of constraints takes an optimal solution to an LP instance that explicitly enforces only a (potentially empty) subset of the family. It either confirms that all constraints are satisfied or returns a most violated constraint. Thus one can add this constraint and repeat at most a polynomial number of times until all are satisfied.

Reducing the integrality gap of an IP formulation has two advantages. It can lead to better LP-based approximation algorithm bounds as described above. It can also help exact solvers run faster or solve instances it could not before. Exact IP solvers are based on intelligent branch-and-bound strategies. As mentioned above, commercial and open-source MILP solvers can find exact solutions (or near-optimal solutions with a provable bound) to many specific instances of NP-hard combinatorial optimization problems. These solvers use the LP relaxation to get lower bounds (for minimization problems). The search requires exponential time in the worst case. But this search is practically feasible when the solver can prune large amounts of the search space. This happens when the lower bound for a region of the search space (subproblem) is worse than the value of a known feasible solution. This requires a way to find a good heuristic solution and it requires good lower bounds that are as close to the actual optimal value of an IP subproblem as possible.

In this paper, we give a method to find feasible solutions for IPs if the integrality gap is bounded. The method is also a tool for evaluating the integrality gap for a formulation. Researcher can use it to determine whether they should expend effort to find new classes of cuts. They can also use it to help guide theory for finding tighter bounds on the integrality gap for classic problems like the traveling salesman problem.

We now describe IPs and our methods more formally. The set of feasible points for a pure IP (henceforth IP) is the set

$$S(A,b) = \{x \in \mathbb{Z}^n \ : \ Ax \geq b\}, \tag{1}$$

where matrix A of rationals has m constraints on n variables and $b \in \mathbb{R}^m$. If we drop the integrality constraints, we have the linear relaxation of set $S(A,b)$,

$$P(A,b) = \{x \in \mathbb{R}^n \ : \ Ax \geq b\}. \tag{2}$$

Let $I = (A,b)$ denote an instance. Then $S(I)$ and $P(I)$ denote $S(A,b)$ and $P(A,b)$, respectively. Given a linear objective function c, an IP is $\min \{cx : \ x \in S(I)\}$.

Relaxing the integrality constraints gives the polynomial-time-solvable linear-programming relaxation: $\min\{cx : x \in P(I)\}$. The optimal value of this linear program (LP), denoted $z_{\text{LP}}(I, c)$, is a lower bound on the optimal value for the IP, denoted $z_{\text{IP}}(I, c)$.

Many researchers (see [WS11, Vaz01]) have developed polynomial time LP-based approximation algorithms that find solutions for special classes of IPs whose cost are provably at most $C \cdot z_{LP}(I, c)$ for some (possibly constant) function C. If the analysis uses the LP bound to prove the approximation quality, then C is at least the integrality gap.

Definition 1. *The integrality gap $g(I)$ for instance I is:*

$$g(I) = \max_{c \geq 0} \frac{z_{IP}(I, c)}{z_{LP}(I, c)},$$

where $z_{IP}(I, c)$ is the optimal solution to the integer program and $z_{LP}(I, c)$ is the solution to the linear-programming relaxation.

In general the integrality gap is defined similarly for any objective function, but we wish to be explicit about the class of problems we assume in this paper. For example consider the minimum cost 2-EDGE-CONNECTED MULTI-SUBGRAPH PROBLEM (2EC): Given a graph $G = (V, E)$ and $c \in \mathbb{R}^E_{\geq 0}$, 2EC asks for the minimum cost 2-edge-connected subgraph of G, with multi-edges allowed. A graph is 2-edge-connected if there are at least two edge-disjoint paths between every pair of vertices. A linear-programming relaxation for this problem, known as the *subtour-elimination* relaxation, is

$$\min\{cx : \sum_{e \in \delta(U)} x_e \geq 2 \text{ for } \emptyset \subsetneq U \subsetneq V,\ x \in [0, 2]^E\}, \tag{3}$$

where $\delta(U)$ for vertex subset U is the set of edges that cross the cut defined by U. That is, each $e \in \delta(U)$ has one endpoint in U and the other endpoint in $V - U$. In this case the instance-specific integrality gap is the integrality gap of the subtour-elimination relaxation for the 2EC on a graph with n vertices. Alexander et al. [ABE06] showed this instance-specific integrality gap is at most $\frac{7}{6}$ for instances with $n = 10$.

The value of $g(I)$ depends on the constraints in (1). We cannot hope to find solutions for the IP with objective values better than $g(I) \cdot z_{LP}(I, c)$. More generally we can define the integrality gap for a class of instances $\mathcal{I}$ as follows.

$$g(\mathcal{I}) = \max_{c \geq 0, I \in \mathcal{I}} \frac{z_{IP}(I, c)}{z_{LP}(I, c)}. \tag{4}$$

For example, the integrality gap of the subtour-elimination relaxation for the 2EC is at most $\frac{3}{2}$ [Wol80] and at least $\frac{6}{5}$ [ABE06]. Therefore, we cannot hope to obtain an LP-based

$(\frac{6}{5} - \epsilon)$-approximation algorithm for this problem using this LP relaxation to bound the quality of a feasible solution.

Our methods apply theory connecting integrality gaps to sets of feasible solutions. Instances I with $g(I) = 1$ have $P(I) = \text{conv}(S(I))$, the convex hull of the lattice of feasible points. In this case, $P(I)$ is an *integral* polyhedron. The spanning tree polytope of graph G, $\text{ST}(G)$, and the perfect-matching polytope of graph G, $\text{PM}(G)$, have this property ([Edm70, Edm65]). Thus the linear-programming relaxation for minimum-cost spanning tree has basic feasible solutions (vertices) that are integral solutions, i.e. spanning trees. For such problems there is an algorithm to express vector $x \in P(I)$ (a feasible LP solution) as a convex combination of points in $S(I)$ (feasible IP solutions) in polynomial time [GLS93].

Proposition 2. *If $g(I) = 1$, then for $x \in P(I)$ there exists a positive integer k and $\theta \in [0,1]^k$, where $\sum_{i=1}^k \theta_i = 1$ and $\tilde{x}^i \in S(I)$ for $i \in [k]$ such that $\sum_{i=1}^k \theta_i \tilde{x}^i \leq x$. Moreover, we can find such a convex combination in polynomial time.*

An equivalent way of describing Proposition 2 is the following Theorem of Carr and Vempala [CV04]. The *dominant* of $P(I)$, which we denote by $\mathcal{D}(P(I))$, is the set of points x' such that there exists a point $x \in P$ with $x' \geq x$ in every component. A polyhedron is of *blocking type* if it is equal to its dominant.

Theorem 3 (Carr, Vempala [CV04])**.** *We have $g(I) \leq C$ if and only if for $x \in P(I)$ there exists $\theta \in [0,1]^k$ where $\sum_{i=1}^k \theta_i = 1$ and $\tilde{x}^i \in \mathcal{D}(S(I))$ for $i \in [k]$ such that $\sum_{i=1}^k \theta_i \tilde{x}^i \leq Cx$.*

Goemans [Goe95] first introduced Theorem 3 for blocking-type polyhedra. While there is an exact algorithm for problems with gap 1, as stated in Proposition 2, Theorem 3 is existential, with no construction. To study integrality gaps, we wish to decompose a suitably scaled linear-programming solution into a convex combination of feasible integer solutions **constructively**. That is, we ask: assuming reasonable complexity assumptions, given a specific problem $\mathcal{I}$ with $1 < g(\mathcal{I}) < \infty$, and $x \in P(I)$ for some $I \in \mathcal{I}$, can we find $\theta \in [0,1]^k$, where $\sum_{i=1}^k \theta_i = 1$ and $\tilde{x}^i \in S(I)$ for $i \in [k]$ such that $\sum_{i=1}^k \theta_i \tilde{x}^i \leq Cx$ in polynomial time? We wish to find the smallest factor C possible.

1.1 Algorithms and Theory Contributions

We give a general approximation framework for solving binary IPs. Consider the set of points described by sets $S(I)$ and $P(I)$ as in (1) and (2), respectively. Assume in addition that $S(I) \in \{0,1\}^n$ and $P(I) \subseteq [0,1]^n$. For a vector $x \in \mathbb{R}^n_{\geq 0}$ such that $x \in P(I)$, let the *support* of x be $\text{supp}(x) = \{i \in [n] : x_i \neq 0\}$. For an integer β let $\{\beta\}^n$ be the vector $y \in \mathbb{R}^n$ with $y_i = \beta$ for $i \in [n]$.

In Section 3 we introduce the *Fractional Decomposition Tree Algorithm* (FDT) which runs in polynomial time algorithm. Given a point $x \in P(I)$ FDT produces a convex combination of feasible points in $S(I)$ that are dominated coordinatewise by a "factor" C times x. If $C = g(I)$, it would be optimal. However we can only guarantee a factor of $g(I)^{|\operatorname{supp}(x)|}$. FDT iteratively solves linear programs that are about the same size as the description of $P(I)$.

Theorem 4. *Assume $1 \le g(I) < \infty$. The Fractional Decomposition Tree (FDT) algorithm, given $x^* \in P(I)$, produces in polynomial time $\lambda \in [0,1]^k$ and $z^1, \ldots, z^k \in S(I)$ such that $k \le |\operatorname{supp}(x^*)|$, $\sum_{i=1}^k \lambda_i z^i \le \min(Cx^*, \{1\}^n)$, and $\sum_{i=1}^k \lambda_i = 1$. Moreover, $C \le g(I)^{|\operatorname{supp}(x^*)|}$.*

In Section 2 we describe a subroutine of the FDT, called the DomToIP algorithm, which finds feasible solutions to any IP with finite gap. This can be of independent interest, especially in proving that a model has unbounded gap.

Theorem 5. *Assume $1 \le g(I) < \infty$. The DomToIP algorithm finds $\hat{x} \in S(I)$ in polynomial time.*

Here is how the FDT algorithm works at a high level for an instance I with LP feasible region $P(I)$: in iteration i the algorithm maintains a convex combination of vectors in $\mathcal{D}(P(I))$ that have a 0 or 1 value for coordinates indexed $0, \ldots, i-1$. Let y be a vector in the convex combination from iteration $i-1$. We solve a linear-programming problem that gives us $\theta_0, \theta_1 \in [0,1]$ and $y^0, y^1 \in \mathcal{D}(P(I))$ such that $y \ge \theta_0 y^0 + \theta_1 y^1$, $\theta_0 + \theta_1 \le g(I)$, $y^0_i = 0$ and $y^1_i = 1$. We then replace y in the convex combination with $\frac{\theta_0}{\theta_0+\theta_1} y^0 + \frac{\theta_1}{\theta_0+\theta_1} y^1$. Repeating this for every vector in the convex combination from the previous iteration yields a new convex combination of points. It is "more" integral because now all vectors in the convex combination are integral in their first $i+1$ elements. If in any iteration there are too many points in the convex combination, we solve a linear-programming problem that "prunes" the convex combination. At the end we have a convex combination of integer solutions $\mathcal{D}(P(I))$. For each such solution z we invoke the DomToIP algorithm to find $z' \in S(I)$ where $z' \le z$.

One can extend the FDT algorithm for binary IPs into covering[1] $\{0,1,2\}$ IPs by losing a factor $2^{|\operatorname{supp}(x)|}$ on top of the loss for FDT. To eradicate this extra factor, we must treat the coordinate i with $x_i = 1$ differently. In Section 4 we focus on the 2-edge-connected multi-subgraph graph problem (2EC) defined above. It's subtour-elimination LP relaxation is given in (3). For input graph G, let $\operatorname{Subtour}(G)$ denote the feasible region of this LP. Let $\operatorname{2EC}(G)$ be the convex hull of incidence vectors[2] of 2-edge-connected multi-subgraphs of

[1] A covering IP has nonnegative constraint matrix, objective coefficients and right-hand side (A, b, c).

[2] The incidence vector x of a 2-edge-connected multi-subgraph has an element for each edge e, with $x_e \in \{0,1,2\}$ indicating the number of times edge e appears in the solution.

graph G. Following the definition in (4) have

$$g(2\text{EC}) = \max_{c \geq 0, G} \frac{\min_{x \in 2\text{EC}(G)} cx}{\min_{x \in \text{Subtour}(G)} cx}. \tag{5}$$

Theorem 6. *Let $G = (V, E)$ and x be an extreme point of* Subtour(G). *The FDT algorithm for 2EC produces $\lambda \in [0, 1]^k$ and 2-edge-connected multi-subgraphs $F_1, \ldots, F_k$ such that $k \leq 2|V| - 1$, $\sum_{i=1}^{k} \lambda_i \chi^{F_i} \leq \min(Cx, \{2\}^n)$, and $\sum_{i=1}^{k} \lambda_i = 1$. Moreover, $C \leq g(2\text{EC})^{|E_x|}$.*

1.2 Experiments.

Although the bounds guaranteed in both Theorems 4 and 6 are large, in Section 5 we show that in practice, the algorithm can work well for network design problems like those described above. We also show how one might use FDT to investigate the integrality gap for such well-studied problems.

1.2.1 Minimum vertex cover problem

In the MINIMUM VERTEX COVER PROBLEM (VC) we are given a graph $G = (V, E)$ and $c \in \mathbb{R}^E_{\geq 0}$. A subset of U of V is a *vertex cover* if for all $e \in E$ at least one endpoint of e is in U. The goal in VC is to find the minimum-cost vertex cover. The linear-programming relaxation for VC is

$$\min\{cx \ : \ x_u + x_v \geq 1 \text{ for all } e = uv \in E, \ x \in [0, 1]^V\}. \tag{6}$$

The integrality gap of this formulation is exactly 2 [WS11]. Austrin, Khot and Safra [AKS11] show that it is UG-hard to approximate VC within any factor strictly better than 2. We compare FDT and the feasbility pump heuristic [FGL05] on the small instances of the PACE[3] 2019 challenge test cases [DFH19]. FDT was 3-5x slower, but still ran in seconds. It always gave a better vertex cover. For some instances the FDT solution's relative gap with respect to the optimal was half that of feasibility pump.

1.2.2 Tree augmentation problem

In the TREE AUGMENTATION PROBLEM (TAP) we are given a graph $G = (V, E)$, a spanning tree T of G and a cost vector $c \in \mathbb{R}^{E \setminus T}_{\geq 0}$. A subset F of $E \setminus T$ is called a *feasible augmentation* if $(V, T \cup F)$ is a 2-edge-connected graph. In TAP we seek the minimum cost feasible

[3]Parameterized Algorithms and Computational Experiments: `https://pacechallenge.org/2019/`

augmentation. The natural linear-programming relaxation for TAP is

$$\min\{cx \ : \ \sum_{\ell \in \text{cov}(e)} x_\ell \geq 1 \text{ for } e \in T, \ x \in [0,1]^{E \setminus T}\}. \tag{7}$$

where $\text{cov}(e)$ for $e \in T$ is the set of edges $\ell \in E \setminus T$ such that e is in the unique cycle of $T \cup \{\ell\}$. Another way to think of $\text{cov}(e)$ is that if we remove edge e from tree T, this partitions the vertices into two connected sets U and $V - U$. Then $\text{cov}(e) = \delta(U)$ is the set of edges $\ell \neq e$ that cross the cut between U and $V - U$, and whose addition would reconnect the tree. We call the LP above the cut LP. The integrality gap of the cut LP is known to be between $\frac{3}{2}$ [CKKK08] and 2 [FJ81]. We create random fractional extreme points of the cut LP and apply FDT to find integral solutions. For our instances, the ratio of the value of the feasible solution to the LP lower bound is always below $\frac{3}{2}$. This provides evidence that the integrality gap for such instances may be less than $\frac{3}{2}$.

1.2.3 2-edge-connected multi-subgraph problem

Known polyhedral structure makes it easier to study integrality gaps for such problems. We use the idea of fundamental extreme point [CR98, BC11, CV04] to create the "hardest" LP solutions to decompose.

There are fairly good bounds for the integrality gap for the Traveling Salesman Problem (TSP, see [ABCC06]) or 2EC. Benoit and Boyd [BB08] used a quadratic program to show the integrality gap of the subtour-elimination relaxation for the TSP, $g(\text{TSP})$, is at most $\frac{20}{17}$ for graphs with at most 10 vertices. Alexander et al. [ABE06] used the same ideas to provide an upper bound of $\frac{7}{6}$ for $g(2\text{EC})$ on graphs with at most 10 vertices.

Consider a graph $G = (V, E)$. A *Carr-Vempala point* $x \in \mathbb{R}^E$ is a fractional point in $\text{Subtour}(G)$ where the edges e with $0 < x_e < 1$ form a single cycle in G and the vertices on the cycle are connected via vertex-disjoint paths of edges e with $x_e = 1$ (see Figure 2).

Carr and Vempala [CV04] showed that $g(2\text{EC})$ is achieved for instances where the optimal solution to $\min_{x \in \text{Subtour}(G)} cx$ is a Carr-Vempala point. Multiple groups of researchers have conjectured that $g(2\text{EC}) \leq \frac{6}{5}$ (see [ABE06, BL17, HN18], but the only known upper bound on $g(2\text{EC})$ stands at $\frac{3}{2}$ [Wol80]. We show that the integrality gap is at most $\frac{6}{5}$ for Carr-Vempala points with at most 12 vertices on the cycle formed by the fractional edges, thereby confirming the conjecture for these instances. Note that the number of vertices in these instances can be arbitrarily high since the paths of edges with x-value 1 can be arbitrarily long.

1.3 Contribution summary

In summary, this papers's contributions are:

- We give an algorithm to express any feasible point for the LP relaxation of a binary IP as a convex combination of feasible solutions, provided the IP integrality gap is bounded. This is the Fractional Decomposition Tree (FDT) algorithm.
- We give a simple algorithm to give a feasible solution for any binary IP with bounded integrality gap. If this algorithm fails, then the integrality gap is infinite.
- We experimentally show that FDT can give good approximate solutions to small instances of vertex-cover and tree augmentation problems.
- We show that the integrality gap for 2EC (minimum 2-edge-connected sub-multigraph) is at most $\frac{6}{5}$ for Carr-Vempala points with at most 12 vertices on the cycle formed by the fractional edges, matching the lower bound for integrality gap for 2EC for these instances.
- We demonstrate (with tree augmentation and 2EC) how FDT exeriments can support theoretical work on integrality gaps for specific problems.

2 Finding a Feasible Solution

In this section we give the algorithm for DomToIP and prove its performance (Theorem 5).

Consider an IP instance $I = (A, b)$. Define sets $S(I)$ and $P(I)$ as in (1) and (2), respectively. Assume $S(I) \subseteq \{0,1\}^n$ and $P(I) \subseteq [0,1]^n$. For simplicity in the notation we assume an instance I and denote $P(I), S(I)$, and $g(I)$ by P, S, and g for this section and the next section. Also, for both sections let x^* be the optimal solution to the LP formulation and assume $t = |\operatorname{supp}(x^*)|$. Reorder variables as necessary to put all the nonzero variables into the first t indices. So we can assume $x^*_i = 0$ for $i = t+1, \ldots, n$.

In this section we prove Theorem 5. In fact, we prove a stronger result.

Lemma 7. *Given an integral vector in the dominant of an LP relaxation ($\tilde{x} \in \mathcal{D}(P)$ and $\tilde{x} \in \{0,1\}^n$) for an IP instance that has integrality gap $g < \infty$, there is an algorithm (the DomToIP algorithm) that finds $\bar{x} \in S$ in polynomial time such that $\bar{x} \leq \tilde{x}$.*

Lemma 7 implies Theorem 5, since it is easy to obtain an integer point in $\mathcal{D}(P)$: numerically rounding up x^* (or any fractional point in P) gives us a point in $\mathcal{D}(P)$. Hence, we can assume in the proof below that $\tilde{x}_i = 0$ for $i = t+1, \ldots, n$.

2.1 Proof of Lemma 7: The DomToIP Algorithm

We introduce an algorithm that builds a solution from the input integral point $\tilde{x}$. It finalizes a binary value for each variable iteratively, starting from the first coordinate and ending at the t-th coordinate. In iteration $\ell \in \{0, \dots, t-1\}$, if $\tilde{x}_{\ell+1} = 1$, the algorithm reduces the $\ell + 1$st coordinate of the solution to 0 if there is an LP-feasible point x' that respects the decisions made so far, and has $x'_\ell = 0$. Intuitively, we can fix this coordinate to 0 in the partial solution and still set the remaining coordinates to find the solution promised in Lemma 7. . More specifically, in iteration $\ell \in \{0, \dots, t-1\}$ we produce $x^{(\ell)} \in \mathcal{D}(P)$ such that $x_i^{(\ell)} \in \{0, 1\}$ for $i = 1, \dots, \ell$. These ℓ components will not change for the remainder of the algorithm.

We can set $x^{(0)} = \tilde{x}$. We now show how to find $x^{(\ell+1)}$ from $x^{(\ell)}$. Consider the following linear program. The variables of this linear program are the $z \in \mathbb{R}^n$.

$$\begin{aligned}
\text{FixNextVarLP}(x^{(\ell)}) \qquad \min \quad & z_{\ell+1} && (8)\\
\text{s.t.} \quad & Az \geq b && (9)\\
& z_j = x_j^{(\ell)} \quad j = 1, \dots, \ell && (10)\\
& z_j \leq x_j^{(\ell)} \quad j = \ell+1, \dots, n && (11)\\
& z \geq 0 && (12)
\end{aligned}$$

If the optimal value to FixNextVarLP($x^{(\ell)}$) is 0, then let $x_{\ell+1}^{(\ell+1)} = 0$. Otherwise if the optimal value is strictly positive let $x_{\ell+1}^{(\ell+1)} = 1$. Let $x_j^{(\ell+1)} = x_j^{(\ell)}$ for $j \in [n] \setminus \{\ell + 1\}$. The DomToIP algorithm initializes with $x^{(0)} = \tilde{x}$ and iteratively calls this procedure in order to obtain $x^{(t)}$ (See Algorithm 1).

Algorithm 1: The DomToIP algorithm

Input: $\tilde{x} \in \mathcal{D}(P)$, $\tilde{x} \in \{0, 1\}^n$
Output: $x^{(t)} \in S$, $x^{(t)} \leq \tilde{x}$

```
x^(0) ← x̃
for ℓ = 0 to t − 1 do
    x^(ℓ+1) ← x^(ℓ)
    η ← optimal value of FixNextVarLP(x^(ℓ))
    if η = 0 then
        x_{ℓ+1}^(ℓ+1) ← 0
    else
        x_{ℓ+1}^(ℓ+1) ← 1
    end
end
```

We prove that indeed $x^{(t)} \in S$. First, we need to show that in any iteration $\ell = 0, \ldots, t-1$ of DomToIP that FixNextVarLP($x^{(\ell)}$) is feasible. We show something stronger. For $\ell = 0, \ldots, t-1$ let

$$\text{LP}^{(\ell)} = \{z \in P \ : \ z \leq x^{(\ell)} \text{ and } z_j = x_j^{(\ell)} \text{ for } j \in [\ell]\}, \text{ and}$$
$$\text{IP}^{(\ell)} = \{z \in \text{LP}^{(\ell)} \ : \ z \in \{0,1\}^n\}.$$

If $\text{LP}^{(\ell)}$ is a non-empty set then FixNextVarLP($x^{(\ell)}$) is feasible. We show by induction on ℓ that $\text{LP}^{(\ell)}$ and $\text{IP}^{(\ell)}$ are not empty sets for $\ell = 0, \ldots, t-1$. First, $\text{LP}^{(0)}$ is feasible since by definition $x^{(0)} \in \mathcal{D}(P)$, meaning there exists $z \in P$ such that $z \leq x^{(0)}$. By Theorem 3, there exists $\tilde{z}^i \in S$ and $\theta_i \geq 0$ for $i \in [k]$ such that $\sum_{i=1}^k \theta_i = 1$ and $\sum_{i=1}^k \theta_i \tilde{z}^i \leq gz$. (This assumes a finite integrality gap g). Hence, $\sum_{i=1}^k \theta_i \tilde{z}^i \leq gz \leq gx^{(0)}$. So if $x_j^{(0)} = 0$, then $\sum_{i=1}^k \theta_i \tilde{z}_j^i = 0$, which implies that $\tilde{z}_j^i = 0$ for all $i \in [k]$ and $j \in [n]$ where $x_j^{(0)} = 0$. Hence, $\tilde{z}^i \leq x^{(0)}$ for $i \in [k]$. Therefore $\tilde{z}^i \in \text{IP}^{(0)}$ for $i \in [k]$, which implies $\text{IP}^{(0)} \neq \emptyset$.

Now assume $\text{IP}^{(\ell)}$ is non-empty for some $\ell \in [t-2]$. Since $\text{IP}^{(\ell)} \subseteq \text{LP}^{(\ell)}$ we have $\text{LP}^{(\ell)} \neq \emptyset$ and hence the FixNextVarLP($x^{(\ell)}$) has an optimal solution z^*.

We consider two cases. In the first case, we have $z_{\ell+1}^* = 0$. In this case we set $x_{\ell+1}^{(\ell+1)} = 0$. Since $z^* \leq x^{(\ell+1)}$, we have $z^* \in \text{LP}^{(\ell+1)}$. Also, $z^* \in P$. By Theorem 3 there exists $\tilde{z}^i \in S$ and $\theta_i \geq 0$ for $i \in [k]$ such that $\sum_{i=1}^k \theta_i = 1$ and $\sum_{i=1}^k \theta_i \tilde{z}^i \leq gz^*$. We have $\sum_{i=1}^k \theta_i \tilde{z}^i \leq gz^* \leq gx^{(\ell+1)}$. So for $j \in [n]$ where $x_j^{(\ell+1)} = 0$, we have $z_j^i = 0$ for $i \in [k]$. This implies $\tilde{z}^i \leq x^{(\ell+1)}$ for $i = 1, \ldots, k$. Hence, there exists $z \in S$ such that $z \leq x^{(\ell+1)}$. We claim that $z \in \text{IP}^{(\ell+1)}$. If $z \notin \text{IP}^{(\ell+1)}$ we must have $1 \leq j \leq \ell$ such that $z_j < x_j^{(\ell+1)}$, and thus $z_j = 0$ and $x_j^{(\ell+1)} = 1$. Without loss of generality assume j is the minimum index (between 1 and ℓ) satisfying $z_j < x_j^{(\ell+1)}$. Consider iteration j of the DomToIP algorithm. We have $z \leq x^{(\ell+1)} \leq x^{(j)}$. We also have $x_j^{(j)} = 1$ which implies when we solved FixNextVarLP($x^{(j-1)}$) the optimal value was strictly larger than zero. However, z is a feasible solution to FixNextVarLP($x^{(j-1)}$) and gives an objective value of 0. This is a contradiction, so $z \in \text{IP}^{(\ell+1)}$.

Now for the second case, assume $z_{\ell+1}^* > 0$. We set $x_{\ell+1}^{(\ell+1)} = 1$. For each point $z \in \text{LP}^{(\ell)}$ we have $z_{\ell+1} > 0$, so for each $z \in \text{IP}^{(\ell)}$ we have $z_{\ell+1} > 0$, i.e. $z_{\ell+1} = 1$. This means that $z \in \text{IP}^{(\ell+1)}$, and $\text{IP}^{(\ell+1)} \neq \emptyset$.

Now consider $x^{(t)}$. Let z be the optimal solution to $\text{LP}^{(t-1)}$. If $x_t^{(t)} = 0$, we have $x^{(t)} = z$, which implies that $x^{(t)} \in P$, and since $x^{(t)} \in \{0,1\}^n$ we have $x^{(t)} \in S$. If $x_t^{(t)} = 1$, it must be the case that $z_t > 0$. By the argument above there is a point $z' \in \text{IP}^{(t-1)}$. We show that $x^{(t)} = z'$. For $j \in [t-1]$ we have $z'_j = x_j^{(t-1)} = x_j^{(t)}$. We just need to show that $z'_t = 1$. Assume $z'_t = 0$ for contradiction. Then $z' \in \text{LP}^{(t-1)}$ has objective value 0 for

FixNextVarLP($x^{(t-1)}$). This is a contradiction to z being the optimal solution to LP$^{(t-1)}$. This concludes the proof of Lemma 7.

3 FDT on Binary IPs

Recall that x^* was the optimal solution to minimizing a cost function cx over set P, which provides a lower bound on $\min_{(x,y)\in S(I)} cx$. In this section, we prove Theorem 4 by describing the Fractional Decomposition Tree (FDT) algorithm. We also remark that if $g(I) = 1$, then the algorithm gives an exact decomposition of any feasible solution.

The FDT algorithm grows a tree similar to the classic branch-and-bound search tree for integer programs. Each node represents a partially integral vector $\bar{x}$ in $\mathcal{D}(P)$ together with a multiplier $\bar{\lambda}$. The solutions contained in the nodes of the tree become progressively more integral at each level. In each level of the tree, the algorithm maintain a conic combination of points with the properties mentioned above. Leaves of the FDT tree contain integral solutions that dominate a point in P. In Lemma 7 we saw how to turn these into points in S.

Branching on a node. We begin with the following lemmas that show how the FDT algorithm branches on a variable.

Lemma 8. *Given $x' \in \mathcal{D}(P)$ and $\ell \in [n]$ where $x'_\ell < 1$, in polynomial time we can find vectors $\hat{x}^0, \hat{x}^1$ and scalars $\gamma_0, \gamma_1 \in [0,1]$ such that: (i) $\gamma_0 + \gamma_1 \geq 1/g$, (ii) $\hat{x}^0$ and $\hat{x}^1$ are in P, (iii) $\hat{x}^0_\ell = 0$ and $\hat{x}^1_\ell = 1$, and (iv) $\gamma_0\hat{x}^0 + \gamma_1\hat{x}^1 \leq x'$.*

Proof. Consider the following linear program which we denote by LPC(ℓ, x'). The variables of LPC(ℓ, x') are λ_0, λ_1, x^0 and x^1.

$$\begin{aligned}
\text{LPC}(\ell, x') \qquad \max \quad & \lambda_0 + \lambda_1 && && (13)\\
\text{s.t.} \quad & Ax^j \geq b\lambda_j && \text{for } j = 0, 1 && (14)\\
& 0 \leq x^j \leq \lambda_j && \text{for } j = 0, 1 && (15)\\
& x^0_\ell = 0,\ x^1_\ell = \lambda_1 && && (16)\\
& x^0 + x^1 \leq x' && && (17)\\
& \lambda_0, \lambda_1 \geq 0 && && (18)
\end{aligned}$$

Let y^0, y^1, λ^*_0, and λ^*_1 be an optimal solution to the LP above. Set $\gamma_0 = \lambda^*_0$ and $\gamma_1 = \lambda^*_1$. Let $\hat{x}^0 = y^0/\lambda^*_0$, $\hat{x}^1 = y^1/\lambda^*_1$. This choice satisfies (ii), (iii), (iv). To show that (i) is also satisfied we prove the following claim.

Claim 1. *We have $\gamma_0 + \gamma_1 \geq 1/g$.*

Proof. We show that there is a feasible solution that achieves the objective value of $\frac{1}{g}$. By Theorem 3 there exists $\theta \in [0,1]^k$, with $\sum_{i=1}^k \theta_i = 1$ and $\tilde{x}^i \in S$ for $i \in [k]$ such that $\sum_{i=1}^k \theta_i \tilde{x}^i \leq g x'$. So

$$x' \geq \sum_{i=1}^{k} \frac{\theta_i}{g} \tilde{x}^i = \sum_{i \in [k]: \bar{x}^i_\ell = 0} \frac{\theta_i}{g} \tilde{x}^i + \sum_{i \in [k]: \bar{x}^i_\ell = 1} \frac{\theta_i}{g} \tilde{x}^i. \tag{19}$$

For $j = 0, 1$, let $x^j = \sum_{i \in [k]: \bar{x}^i_\ell = j} \frac{\theta_i}{g} \tilde{x}^i$. Also let $\lambda_0 = \sum_{i \in [k]: \bar{x}^i_\ell = 0} \frac{\theta_i}{g}$ and $\lambda_1 = \sum_{i \in [k]: \bar{x}^i_\ell = 1} \frac{\theta_i}{g}$. Note that $\lambda_0 + \lambda_1 = 1/g$. Constraint (17) is satisfied by Inequality (19). Also, for $j = 0, 1$ we have

$$Ax^j = \sum_{i \in [k], \bar{x}^i_\ell = j} \frac{\theta_i}{g} A\tilde{x}^i \geq b \sum_{i \in [k], \bar{x}^i_\ell = j} \frac{\theta_i}{g} = b\lambda_j. \tag{20}$$

Hence, Constraints (14) holds. Constraint (16) also holds since x^0_ℓ is obviously 0 and $x^1_\ell = \sum_{i \in [k]: \bar{x}^i_\ell = 1} \frac{\theta_i}{g} = \lambda_1$. The rest of the constraints trivially hold. ◊

This concludes the proof of Lemma 8. □

We now show if x' in the statement of Lemma 8 is partially integral, we can find solutions with more integral components.

Lemma 9. *Given $x' \in \mathcal{D}(P)$ where $x'_1, \ldots, x'_{\ell-1} \in \{0,1\}$ and $x'_\ell < 1$ for some $\ell \geq 1$ we can find in polynomial time vectors $\hat{x}^0, \hat{x}^1$ and scalars $\gamma_0, \gamma_1 \in [0,1]$ such that: (i) $1/g \leq \gamma_0 + \gamma_1 \leq 1$, (ii) $\hat{x}^0$ and $\hat{x}^1$ are in $\mathcal{D}(P)$, (iii) $\hat{x}^0_\ell = 0$ and $\hat{x}^1_\ell = 1$, (iv) $\gamma_0 \hat{x}^0 + \gamma_1 \hat{x}^1 \leq x'$,(v) $\hat{x}^i_j \in \{0,1\}$ for $i = 0, 1$ and $j \in [\ell - 1]$.*

Proof. By Lemma 8 we can find $\bar{x}^0$, $\bar{x}^1$, γ_0 and γ_1 that satisfy (i), (ii), (iii), and (iv). We define $\hat{x}^0$ and $\hat{x}^1$ as follows. For $i = 0, 1$, for $j \in [\ell - 1]$, let $\hat{x}^i_j = \lceil \bar{x}^i_j \rceil$, for $j = \ell, \ldots, t$ let $\hat{x}^i_j = \bar{x}^i_j$.

We now show that $\hat{x}^0$, $\hat{x}^1$, γ_0, and γ_1 satisfy all the conditions. Note that conditions (i), (ii), (iii), and (v) are trivially satisfied. Thus we only need to show (iv) holds. We need to show that $\gamma_0 \hat{x}^0_j + \gamma_1 \hat{x}^1_j \leq g x'_j$. If $j = \ell, \ldots, t$, then this clearly holds. Hence, assume $j \leq \ell - 1$. By the property of x' we have $x'_j \in \{0,1\}$. If $x'_j = 0$, then by Constraint (17) we have $\bar{x}^0_j = \bar{x}^1_j = 0$. Therefore, $\hat{x}^i_j = 0$ for $i = 0, 1$, so (iv) holds. Otherwise if $x'_j = 1$, then we have $\gamma_0 \hat{x}^0_j + \gamma_1 \hat{x}^1_j \leq \gamma_0 + \gamma_1 \leq 1 \leq x'_j$. Therefore (v) holds. □

Growing and Pruning FDT tree. The FDT algorithm maintains nodes L_i in iteration i of the algorithm. The nodes in L_i correspond to the nodes in level L_i of the FDT tree. The points in the leaves of the FDT tree, L_t, are points in $\mathcal{D}(P)$ and are integral for all integer variables.

Lemma 10. *There is a polynomial time algorithm that produces sets $L_0, \ldots, L_t$ of pairs of $x \in \mathcal{D}(P)$ together with multipliers λ with the following properties for $i = 0, \ldots, t$: (a) If $x \in L_i$, then $x_j \in \{0, 1\}$ for $j \in [i]$, i.e. the first i coordinates of a solution in level i are integral, (b) $\sum_{[x,\lambda]\in L_i} \lambda \geq \frac{1}{g^i}$, (c) $\sum_{[x,\lambda]\in L_i} \lambda x \leq x^*$, (d) $|L_i| \leq t$.*

Proof. We prove this lemma using induction but one can clearly see how to turn this proof into a polynomial time algorithm. Let L_0 be the set that contains a single node (*root of the FDT tree*) with x^* and multiplier 1. All the requirements in the lemma are satisfied for this choice.

Suppose by induction that we have constructed sets $L_0, \ldots, L_i$. Let the solutions in L_i be x^j for $j \in [k]$ and λ_j be their multipliers, respectively. For each $j \in [k]$ if $x^j_{i+1} = 1$ we add the pair (x^j, λ_j) to L'. Otherwise, applying Lemma 9 (setting $x' = x^j$ and $\ell = i + 1$) we can find x^{j0}, x^{j1}, λ^0_j and λ^1_j with the properties (i) to (v) in Lemma 9. Add the pairs $(x^{j0}, \lambda_j\lambda^0_j)$ and $(x^{j1}, \lambda_j\lambda^1_j)$ to L'. It is easy to check that set L' is a suitable candidate for L_{i+1}, i.e. set L' satisfies (a), (b) and (c). However we can only ensure that $|L'| \leq 2k \leq 2t$, and might have $|L'| > t$. We call the following linear program Pruning(L'). Let $L' = \{[x^1, \gamma_1], \ldots, [x^{|L'|}, \gamma_{|L'|}]\}$. The variables of Pruning(L') are scalar variables θ_j for each node j in L'.

$$\text{Pruning}(L') \qquad \{\max \sum_{j=1}^{|L'|} \theta_j \ : \ \sum_{j=1}^{|L'|} \theta_j x^j_i \leq x^*_i \text{ for } i \in [t],\ \theta \geq 0\} \tag{21}$$

Setting $\theta = \gamma$ gives a feasible solution to Pruning(L'). Let θ^* be the optimal vertex solution to this LP. Since the problem is in $\mathbb{R}^{|L'|}$, θ^* has to satisfy $|L'|$ linearly independent constraints at equality. However, there are only t constraints of type $\sum_{j=1}^{|L'|} \theta_j x^j_i \leq x^*_i$. Therefore, there are at most t coordinates of θ^*_j that are non-zero. Set L_{i+1} which consists of x^j for $j = 1, \ldots, |L'|$ and their corresponding multipliers θ^*_j satisfy the properties in the statement of the lemma. We can discard the nodes in L_{i+1} that have $\theta^*_j = 0$, so $|L_{i+1}| \leq t$. Also, since θ^* is optimal and γ is feasible for Pruning(L'), we have $\sum_{j=1}^{|L'|} \theta^*_j \geq \sum_{j=1}^{|L'|} \gamma_j \geq \frac{1}{g^{i+1}}$. □

From leaves of FDT to feasible solutions. For the leaves of the FDT tree, L_t, we have that every solution x in L_t has $x \in \{0, 1\}^n$ and $x \in \mathcal{D}(P)$. By applying Lemma 7 we can obtain a point $x' \in S$ such that $x' \leq x$. This concludes the description of the FDT

algorithm and proves Theorem 4. See Algorithm 2 for a summary of the FDT algorithm.

Algorithm 2: Fractional Decomposition Tree Algorithm

Input: $P = \{x \in \mathbb{R}^n : Ax \geq b\}$ and $S = \{x \in P : x \in \{0,1\}^n\}$ such that $g = \max_{c \in \mathbb{R}^n_+} \frac{\min_{x \in S} cx}{\min_{x \in P} cx}$ is finite, $x^* \in P$

Output: $z^i \in S$ and $\lambda_i \geq 0$ for $i \in [k]$ such that $\sum_{i=1}^k \lambda_i = 1$, and $\sum_{i=1}^k \lambda_i z^i \leq g^t x^*$

$L^0 \leftarrow [x^*, 1]$
for $i = 1$ ***to*** t **do**
  $L' \leftarrow \emptyset$
  for $[x, \lambda] \in L^i$ **do**
    Apply Lemma 9 to obtain $[\hat{x}^0, \gamma_0]$ and $[\hat{x}^1, \gamma_1]$
    $L' \leftarrow L' \cup \{[\hat{x}^0, \lambda \cdot \gamma_0]\} \cup \{[\hat{x}^1, \lambda \cdot \gamma_1]\}$
  end
  Apply Lemma 10 to prune L' to obtain L^{i+1}.
end
for $[x, \lambda] \in L^t$ **do**
  Apply Algorithm 1 to x to obtain $z \in S$
  $F \leftarrow F \cup \{[z, \lambda]\}$
end
return F

There are $O(n^2)$ nodes in the FDT tree. A faster way to achieve feasible solutions with good quality for an IP with bounded integrality gap is an algorithm that takes a random dive into the FDT tree, hence only visiting $O(n)$ nodes.

Algorithm 3: Dive FDT Algorithm

Input: $P = \{x \in \mathbb{R}^n : Ax \geq b\}$ and $S = \{x \in P : x \in \{0,1\}^n\}$ such that $g = \max_{c \in \mathbb{R}^n_+} \frac{\min_{x \in S} cx}{\min_{x \in P} cx}$ is finite, $x^* \in P$

Output: $z \in S$

$y = x^*$
for $i = 1$ ***to*** t **do**
  Apply Lemma 9 to obtain $[\hat{x}^0, \gamma_0]$ and $[\hat{x}^1, \gamma_1]$
  $i \sim \text{Bernoulli}(\frac{\gamma_0}{\gamma_0 + \gamma_1})$
  $y \to \hat{x}^i$
end
Apply Algorithm 1 to y to obtain $z \in S$
return z

4 FDT for 2EC

In Section 3 our focus was on binary IPs. In this section, in an attempt to extend FDT to {0,1,2} problems we introduce an FDT algorithm for a 2-edge-connected multi-subgraph problem. Given a graph $G = (V, E)$ a multi-subset of edges F of G is a 2-edge-connected multi-subgraph of G if for each set $\emptyset \subset U \subset V$, the number of edges in F that have one endpoint in U and one not in U is at least 2. In 2EC, we are given non-negative costs on the edges of G and the goal is to find the minimum cost 2-edge-connected multi-subgraph of G. We want to prove Theorem 6.

Theorem 6. *Let $G = (V, E)$ and x be an extreme point of* Subtour(G). *The FDT algorithm for 2EC produces $\lambda \in [0, 1]^k$ and 2-edge-connected multi-subgraphs $F_1, \ldots, F_k$ such that $k \leq 2|V| - 1$, $\sum_{i=1}^{k} \lambda_i \chi^{F_i} \leq \min(Cx, \{2\}^n)$, and $\sum_{i=1}^{k} \lambda_i = 1$. Moreover, $C \leq g(\text{2EC})^{|E_x|}$.*

We do not know the exact value for $g(\text{2EC})$, but we know $\frac{6}{5} \leq g(\text{2EC}) \leq \frac{3}{2}$ [ABE06, Wol80]. The FDT algorithm for 2EC is very similar to the one for binary IPs, but there are some differences as well. A natural thing to do is to have three branches for each node of the FDT tree, however, the branches that are equivalent to setting a variable to 1, might need further decomposition. That is the main difficulty when dealing with $\{0, 1, 2\}$-IPs.

First, we need a branching lemma. Observe that the following branching lemma is essentially a translation of Lemma 8 for $\{0, 1, 2\}$ problems except for one additional clause.

Lemma 11. *Given $x \in$ Subtour(G), and $e \in E$ we can find in polynomial time vectors x^0, x^1 and x^2 and scalars γ_0, γ_1, and γ_2 such that: (i) $\gamma_0 + \gamma_1 + \gamma_2 \geq 1/g(\text{2EC})$, (ii) x^0, x^1, and x^2 are in* Subtour(G), *(iii) $x^0_e = 0$, $x^1_e = 1$, and $x^2_e = 2$, (iv) $\gamma_0 x^0 + \gamma_1 x^1 + \gamma_2 x^2 \leq x$, (v) for $f \in E$ with $x_f \geq 1$, we have $x^j_f \geq 1$ for $j = 0, 1, 2$.*

Proof. Consider the following LP with variables λ_j and x^j for $j = 0, 1, 2$.

$$\max \quad \sum_{j=0,1,2} \lambda_j \tag{22}$$

$$\text{s.t.} \quad x^j(\delta(U)) \geq 2\lambda_j \qquad \text{for } \emptyset \subset U \subset V, \text{ and } j = 0, 1, 2 \tag{23}$$

$$0 \leq x^j \leq 2\lambda_j \qquad \text{for } j = 0, 1, 2 \tag{24}$$

$$x^j_e = j \cdot \lambda_j \qquad \text{for } j = 0, 1, 2 \tag{25}$$

$$x^j_f \geq \lambda_j \qquad \text{for } f \in E \text{ where } x_f \geq 1, \text{ and } j = 0, 1, 2 \tag{26}$$

$$x^0 + x^1 + x^2 \leq x \tag{27}$$

$$\lambda_0, \lambda_1, \lambda_2 \geq 0 \tag{28}$$

Let x^j, γ_j for $j = 0, 1, 2$ be an optimal solution solution to the LP above. Let $\hat{x}^j = x^j/\gamma_j$ for $j = 0, 1, 2$ where $\gamma_j > 0$. If $\gamma_j = 0$, let $\hat{x}^j = 0$. Observe that (ii), (iii), (iv), and (v) are

satisfied with this choice. We can also show that $\gamma_0 + \gamma_1 + \gamma_2 \geq 1/g(2\text{EC})$, which means that (i) is also satisfied. The proof is similar to the proof of the claim in Lemma 8, but we need to replace each $f \in E$ with $x_f \geq 1$ with a suitably long path to ensure that Constraint (26) is also satisfied.

Claim 2. *We have* $\gamma_0 + \gamma_1 + \gamma_2 \geq \frac{1}{g(2\text{EC})}$.

Proof. Suppose for contradiction $\sum_{j=0,1,2} \gamma_j = \frac{1}{g(2\text{EC})} - \epsilon$ for some $\epsilon > 0$. Construct graph G' by removing edge f with $x_f \geq 1$ and replacing it with a path P_f of length $\lceil \frac{2}{\epsilon} \rceil$. Define $x'_h = x_h$ for each edge h such that $x_h < 1$. For each $h \in P_f$ let $x'_h = x_f$ for all f with $x_f \geq 1$. It is easy to check that $x' \in \text{Subtour}(G')$. By Theorem 3 there exists $\theta \in [0,1]^k$, with $\sum_{i=1}^k \theta_i = 1$ and 2-edge-connected multi-subgraphs F'_i of G' for $i = 1, \ldots, k$ such that $\sum_{i=1}^k \theta_i \chi^{F'_i} \leq g(2\text{EC}) x'$.

Each F'_i contains at least one copy of every edge in any path P_f, except for at most one edge in the path. We will obtain 2-edge-connected multi-subgraphs $F_1, \ldots, F_k$ of G using $F'_1, \ldots, F'_k$, respectively. To obtain F_i first remove all P_f paths from F'_i. Suppose there is an edge h in P_f such that $\chi_h^{F'_i} = 0$, this means that for any edge $p \in P_f$ such that $p \neq h$, $\chi_p^{F'_i} = 2$. In this case, let $\chi_f^{F_i} = 2$, i.e. add two copies of f to F_i. If there are at least one edge $h \in P_f$ with $\chi_h^{F'_i} = 1$, let $\chi_f^{F_i} = 1$, i.e. add one copy of f to F_i. If for all edges $h \in P_f$, we have $\chi_h^{F'_i} = 2$, then let $\chi_f^{F_i} = 2$. For $f \in E$ with $x_f < 1$ we have

$$\sum_{i=1}^k \theta_i \chi_f^{F_i} = \sum_{i=1}^k \theta_i \chi_f^{F'_i} \leq g(2\text{EC}) x'_f = g(2\text{EC}) x_f. \tag{29}$$

In addition for $f \in E$ with $x_f \geq 1$ we have $\chi_f^{F_i} \leq \frac{\sum_{h \in P_f} \chi_h^{F'_i}}{\lceil \frac{2}{\epsilon} \rceil - 1}$ by construction.

$$\begin{aligned}
\sum_{i=1}^k \theta_i \chi_f^{F_i} &\leq \sum_{i=1}^k \theta_i \frac{\sum_{h \in P_f} \chi_h^{F'_i}}{\lceil \frac{2}{\epsilon} \rceil - 1} \\
&= \frac{\sum_{h \in P_f} \sum_{i=1}^k \theta_i \chi_h^{F'_i}}{\lceil \frac{2}{\epsilon} \rceil - 1} \\
&\leq \frac{\sum_{h \in P_f} g(2\text{EC}) x'_h}{\lceil \frac{2}{\epsilon} \rceil - 1} \\
&= \frac{\sum_{h \in P_f} g(2\text{EC}) x_f}{\lceil \frac{2}{\epsilon} \rceil - 1} \\
&= \frac{\lceil \frac{2}{\epsilon} \rceil}{\lceil \frac{2}{\epsilon} \rceil - 1} g(2\text{EC}) x_f.
\end{aligned}$$

Therefore, since $\frac{\lceil \frac{2}{\epsilon} \rceil}{\lceil \frac{2}{\epsilon} \rceil - 1} \geq 1$, we have

$$x \geq \sum_{i \in [k]: \chi_e^{F_i} = 0} \frac{\theta_i(\lceil \frac{2}{\epsilon} \rceil - 1)}{g(2\mathrm{EC})\lceil \frac{2}{\epsilon} \rceil} \chi^{F_i} + \sum_{i \in [k]: \chi_e^{F_i} = 1} \frac{\theta_i(\lceil \frac{2}{\epsilon} \rceil - 1)}{g(2\mathrm{EC})\lceil \frac{2}{\epsilon} \rceil} \chi^{F_i} + \sum_{i \in [k]: \chi_e^{F_i} = 2} \frac{\theta_i(\lceil \frac{2}{\epsilon} \rceil - 1)}{g(2\mathrm{EC})\lceil \frac{2}{\epsilon} \rceil} \chi^{F_i}. \quad (30)$$

Let $x^j = \sum_{i \in [k]: \chi_e^{F_i} = j} \frac{\theta_i(\lceil \frac{2}{\epsilon} \rceil - 1)}{g(2\mathrm{EC})\lceil \frac{2}{\epsilon} \rceil} \chi^{F_i}$ and $\theta_j = \sum_{i \in [k]: \chi_e^{F_i} = j} \frac{\theta_i(\lceil \frac{2}{\epsilon} \rceil - 1)}{g(2\mathrm{EC})\lceil \frac{2}{\epsilon} \rceil}$ for $j = 0, 1, 2$. It is easy to check that x^j , θ_j for $j = 0, 1, 2$ is a feasible solution to the LP above. We have $\sum_{j=0,1,2} \theta_j = \frac{\lceil \frac{2}{\epsilon} \rceil - 1}{g(2\mathrm{EC})\lceil \frac{2}{\epsilon} \rceil}$. By assumption, we have $\frac{\lceil \frac{2}{\epsilon} \rceil - 1}{g(2\mathrm{EC})\lceil \frac{2}{\epsilon} \rceil} \leq \frac{1}{g(2\mathrm{EC})} - \epsilon$, which is a contradiction. ♢

This concludes the proof. □

In contrast to FDT for binary IPs where we round up the fractional variables that are already branched on at each level, in FDT for 2EC we keep all coordinates as they are and perform a rounding procedure at the end. Formally, let L_i for $i = 1, \ldots, |\operatorname{supp}(x^*)|$ be collections of pairs of feasible points in Subtour(G) together with their multipliers. Let $t = |\operatorname{supp}(x^*)|$ and assume without loss of generality that $\operatorname{supp}(x^*) = \{e_1, \ldots, e_t\}$.

Lemma 12. *The FDT algorithm for 2EC in polynomial time produces sets $L_0, \ldots, L_t$ of pairs $x \in 2\mathrm{EC}(G)$ together with multipliers λ with the following properties for $i \in [t]$: (a) If $x \in L_i$, then $x_{e_j} = 0$ or $x_{e_j} \geq 1$ for $j = 1, \ldots, i$, (b) $\sum_{(x,\lambda) \in L_i} \lambda \geq \frac{1}{g(2\mathrm{EC})^i}$, (c) $\sum_{(x,\lambda) \in L_i} \lambda x \leq x^*$, (d) $|L_i| \leq t$.*

The proof is similar to Lemma 10, but we need to use property (v) in Lemma 11 to prove that (a) also holds.

Proof. We proceed by induction on i. Define $L_0 = \{(x^*, 1)\}$. It is easy to check all the properties are satisfied. Now, suppose by induction we have L_{i-1} for some $i = 1, \ldots, t$ that satisfies all the properties. For each solution x^ℓ in L_{i-1} apply Lemma 11 on x^ℓ and e_i to obtain $x^{\ell j}$ and $\lambda_{\ell j}$ for $j = 0, 1, 2$. Let L' be the collection that contains $(x^{\ell j}, \lambda_\ell \cdot \lambda_{\ell j})$ for $j = 0, 1, 2$, when applied to all (x^ℓ, λ_ℓ) in L_{i-1}. Similar to the proof in Lemma 10 one can check that L_i satisfies properties (b), (c). We now verify property (a). Consider a solution x^ℓ in L_{i-1}. For $e \in \{e_1, \ldots, e_{i-1}\}$ if $x^\ell_e = 0$, then by property (iv) in Lemma 11 we have $x^{\ell j} = 0$ for $j = 0, 1, 2$. Otherwise by induction we have $x^\ell_e \geq 1$ in which case property (v) in Lemma 11 ensures that $x^{\ell j}_e \geq 1$ for $j = 0, 1, 2$. Also, $x^{\ell j}_{e_i} = j$, so $x^{\ell j}_{e_i} = 0$ or $x^{\ell j}_{e_i} \geq 1$ for $j = 0, 1, 2$.

Finally, if $|L'| \leq t$ we let $L_i = L'$, otherwise apply Pruning(L') to obtain L_i. □

Consider the solutions x in L_t. For each variable e we have $x_e = 0$ or $x_e \geq 1$.

Lemma 13. *Let x be a solution in L_t. Then $\lfloor x \rfloor \in \text{Subtour}(G)$.*

Proof. Suppose not. Then there is a set of vertices $\emptyset \subset U \subset V$ such that $\sum_{e \in \delta(U)} \lfloor x_e \rfloor < 2$. Since $x \in \text{Subtour}(G)$ we have $\sum_{e \in \delta(U)} x_e \geq 2$. Therefore, there is an edge $f \in \delta(U)$ such that x_f is fractional. By property (a) in Lemma 12, we have $1 < x_f < 2$. Therefore, there is another edge h in $\delta(U)$ such that $x_h > 0$, which implies that $x_h \geq 1$. But in this case $\sum_{e \in \delta(U)} \lfloor x_e \rfloor \geq \lfloor x_f \rfloor + \lfloor x_h \rfloor \geq 2$. This is a contradiction. □

The FDT algorithm for 2EC iteratively applies Lemmas 11 and 12 to variables $x_1, \ldots, x_t$ to obtain leaf point solutions L_t. Finally, we just need to apply Lemma 13 to obtain the 2-edge-connected multi-subgraphs from every solution in L_t. Since x is an extreme point we have $t \leq 2|V| - 1$ [BP90]. By Lemma 12 we have

$$\sum_{(x,\lambda) \in L_t} \frac{\lambda}{\sum_{(x,\lambda) \in L_t} \lambda} \lfloor x \rfloor \leq \frac{1}{\sum_{(x,\lambda) \in L_t} \lambda} \sum_{(x,\lambda) \in L_t} \lambda x \leq g^t_{\text{2EC}} x^*.$$

5 Computational Experiments with FDT

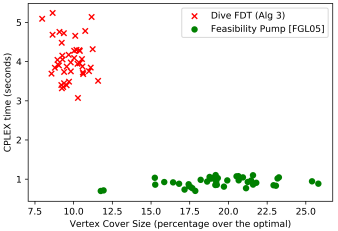
We ran FDT on three network design problems: VC, TAP and 2EC.

We implemented the experiments for VC and TAP in Python running on a linux workstation (Ubuntu 18.04.3) with 8 cores of Intel(R) Core(TM) i7-8565U CPU 1.80GHz processors and 1Mb of cache. We used the CPLEX 12.9.0.0 solver to solve the pyomo LP models. We ran the experiments for 2EC on a Windows machine, coded in AMPL with CPLEX as the solver.

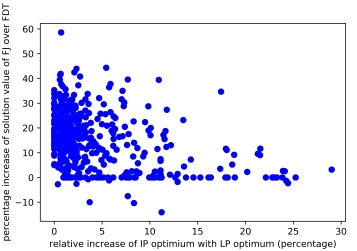
FDT on VC instances from (PACE 2019) [DFH19]. We compared Dive FDT (Algorithm 3) with feasbility pump [FGL05] in terms of running time and the quality of solution provided by each algorithm. We used the small (200 vertex) test cases from the PACE 2019 vertex-cover challenge. The results are presented in Figure 1a.

FDT on randomly generated instances of TAP. Recall that in the tree augmentation problem (TAP) we are given a tree $T = (V, E)$, a set of non-tree links L between vertices in V and costs $c \in \mathbb{R}^L_{\geq 0}$. A feasible augmentation is $L' \subseteq L$ such that $T + L'$ is 2-edge-connected. In TAP we wish to find the minimum-cost feasible augmentation. The integrality gap of the cut LP for TAP (given in (7)) is

$$g(\text{TAP}) = \max_{c \in \mathbb{R}^L_{\geq 0}} \frac{\min_{x \in \text{TAP}(T,L)} cx}{\min_{x \in \text{CUT}(T,L)} cx},$$



(a) Dive FDT vs feasbility pump on the instances of PACE 2019 [DFH19] with 200 vertices.



(b) TAP on our random instances: FDT run on LP optimal vs the 2-approximation for TAP[FJ81].

Figure 1: Computational experiments with the FDT algorithm

where $\mathrm{TAP}(T, L)$ is the feasible set for the IP and $\mathrm{CUT}(T, L)$ is the feasible set for the cut LP (relaxation). We know $\frac{3}{2} \leq g(\mathrm{TAP}) \leq 2$ [FJ81, CKKK08]. The IP $\min_{x \in \mathrm{TAP}(T,L)} cx$ is binary.

As input for our experiments, we considered full binary trees with 3 to 7 levels with a link for each pairs of leaves. We set the link costs uniformly at random. We summarize these test instances in Table 1. We ran binary FDT on each test instance and chose the solution with minimum cost. We compare the FDT solutions to those from the circulation-based 2-approximation algorithm of Frederickson and JáJá [FJ81] in Figure 1b. The simple FJ heuristic was faster than FDT, from about 8x to about 60x on our examples. However, FDT still ran in less than a minute on the bigger problems and less than 10 seconds on the others. FDT (in the worst case) solves $\Omega(m^2)$ LPs, where m is the number of edges, and FJ solves one LP. So the difference in running time, which depends on instance size, is not surprising. FDT runs in polynomial time and usually gives much better solutions than FJ.

number of edges in T	number of links in L	number of instances (T, L)
6	6	100
14	28	100
30	120	100
62	496	100
126	2016	100

Table 1: Summary of the randomly generated instances of TAP.

For all 500 instances in our experiments, running FDT on the LP optimal (fractional

extreme point) of the cut LP gave a feasible solution with value at most a factor $\frac{3}{2}$ larger than the LP lower bound. Such a feasible solution gives an upper bound on the integrality gap $g(\text{TAP})$ of that specific instance of at most $\frac{3}{2}$. In fact, the integrality gap upper bound derived this way was equal to $\frac{3}{2}$ for only one instance. For 480 instances, the integrality gap upper bound was $\frac{4}{3}$, for 16 instances it was $\frac{6}{5}$, for 2 instances it was $\frac{8}{7}$, for 1 instance it was $\frac{10}{9}$.

Computational comparison between Christofides' algorithm and FDT for 2EC on Carr-Vempala points. As described in Section 1.2.3, when the LP optimum for (3) is a Carr-Vempala point (see Figure 2), the integrality gap of that instance is equal to that of 2EC. So in that sense, these are the hardest fractional points to round to a feasible solution. We now compare the quality of FDT's solution from a Carr-Vempala point to that of Christofides, the best known approximation algorithm. In the classic graph-based Christofides algorithm for TSP, find a minimum spanning tree of the graph. Then, since TSP runs on a complete graph, add a perfect matching on the vertices that have odd degree in the spanning tree to make the (multi)-subgraph 2-edge-connected. If we stop that algorithm without shortcutting, we have a feasible solution for Metric-2EC (given a complete graph, find the minimum-cost 2-edge-connected subgraph, with no multiedges). For 2EC, instead of a matching, we add an O-join, where O is the set of odd-degree vertices in the spanning tree [4]. The best-of-many Christofides (BOMC) algorithm [AKS15] works better than the classic Christofides algorithm in practice [GW17]. The BOMC algorithm samples many spanning trees from the LP relaxation (with edges chosen proportionally to the LP value), augments each spanning tree with the matching, and takes the best solution.

The polyhedral analysis of Christofides shows that if $x \in \text{Subtour}(G)$, then $\frac{3}{2}x$ dominates a convex combination of connected Eulerian (hence 2-edge-connected) multi-subgraphs of G [Wol80, SW90]. This bound holds for any point $x \in \text{Subtour}(G)$ for any graph G, including any Carr-Vempala point. We show how this bound can be improved for specific instances by doing a slightly improved analysis of the algorithm.

Let x be a Carr-Vempala point (from (3)) defined on a graph $G = (V, E)$. If $x \in \text{Subtour}(G)$, then $\frac{|V|-1}{|V|}x$ can be written as a convex combination of spanning trees of G (See Proposition 2 in [Vyg12] for instance). More explicitly,

$$\frac{|V|-1}{|V|} \cdot x = \sum_{i=1}^{k} \lambda_i \chi^{T_i}, \tag{31}$$

[4] For graph $G = (V, E)$ and $O \subseteq V$ with $|O|$ even, an O-join of G is a subgraph of G that has odd degree on the vertices in O and even degree on vertices in $V \setminus O$. Graphs always have an even number of odd vertices since every edge has two endpoints, giving an even number of endpoints.

where T_i is spanning tree of G, χ^{T_i} is its incidence vector, $\sum_{i=1}^{k} \lambda_i = 1$, and $\lambda_i \geq 0$ for $i \in [k]$. The spanning-tree polytope is integral so there are polynomial-time algorithms to construct this decomposition (See Proposition 2 and surrounding discussion).

Let O_i be the set of odd-degree vertices of spanning tree T_i. Adding an O_i-join to tree T_i gives a solution to 2EC. We solve the following LP that allows us to find edges that simultaneously augment all the spanning trees to solutions to 2EC.

$$\min\{\alpha : \sum_{i=1}^{k} \lambda_i y^i = \alpha \cdot x,\ y^i \in \mathcal{D}(O_i\text{- JOIN}(G)) \text{ for } i \in [k]\}. \tag{32}$$

The variables in the above LP are $y^i \in \mathbb{R}^E_{\geq 0}$ for $i \in [k]$, where E is the number of edges in the graph. The parameters λ_i are those we computed in (31). For each $i \in [k]$, we require y^i to be in the dominant of O_i- JOIN(G). We enforce this by adding a set of constraints for each i from [Sch03, p. 490]. These constraints force the inclusion of at least one edge from the cut associated with each set of vertices that have an even number of elements from O_i. The y^i need not be integer vectors. Because the O_i-join polytope is integral, we can replace y^i with a convex combination of O_i-joins of G (proposition 2). Setting $y^i = \frac{x}{2}$ and $\alpha = \frac{1}{2}$ gives a feasible solution to this LP, yielding an integrality gap of at most $\frac{3}{2}$ for the subtour-elimination relaxation for the TSP [Wol80]. More generally, this method gives a $(\frac{|V|-1}{|V|} + \alpha)$-approximation for the specific instance x.

Figure 3 shows FDT's solutions on all Carr-Vempala points that have 10 vertices on the cycle formed by fractional edges. We show for these points the apporoximation factor provided by FDT is always better than those from the polyhedral version of Christofides' algorithm. In Figure 3 the horizontal axis of the plot is indexed with the 60 Carr-Vempala points that we considered. For each Carr-Vempala point x, there are two data points. The value of the first data point depicted by a circle on the vertical axis is $\frac{|V|-1}{|V|} + \alpha$ where α is the optimal solution to (32). The value of the second data point depicted by a cross on the vertical axis is C where C is obtained from applying Theorem 6 to x. In other words, Figure 3 is comparing the instance-specific upper bound on integrality gap certified by Christofides' algorithm to the approximation factor of the FDT algorithm for 2EC.

FDT for 2EC on Carr-Vempala points. We ran FDT for 2EC on 963 fractional extreme points of Subtour(G). We enumerated all (fractional) Carr-Vempala points with 10 and 12 vertices. Table 2 shows that again FDT found solutions better than the integrality-gap lower bound for most instances.

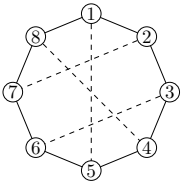


Figure 2: A Carr-Vempala point with 8 vertices on its cycle. Solid lines are edges with value strictly between 0 and 1. Dashed edges represent paths where each edge on the path has $x_e = 1$. There can be an arbitrary number of degree-2 vertices on the dashed paths.

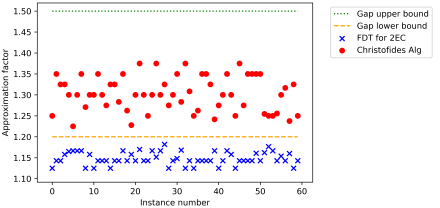


Figure 3: Polyhedral version of Christofides' algorithm vs FDT on all Carr-Vempala points that have 10 vertices on the single cycle formed by fractional edges.

	$C \in [1.08, 1.11]$	$C \in (1.11, 1.14]$	$C \in (1.14, 1.17]$	$C \in (1.17, 1.2]$
2EC	79	201	605	78

Table 2: FDT for 2EC implemented applied to all Carr-Vempala with 10 or 12 vertices. A Carr-Vempala point with k vertices has $\frac{3k}{2}$ edges. Thus, the upper bound provided by Theorem 6 is $g(2\text{EC})^{3k/2}$. The lower bound on $g(2\text{EC})$ is $\frac{6}{5}$.

6 Concluding Remarks

The results in Sections 2, 3 and 4 hold if the intial integer program has some auxilliary continuous variables[5]. That is, when we have

$$S(A, b) = \{x \in \mathbb{Z}^n \times \mathbb{R}^p : Ax \geq b\},$$

[5]Here by auxilliary variables we mean a variable that does not participate in the objective function

where the last p variables are continous, $P(A,b) = \{x \in \mathbb{R}^{n+p} : Ax \geq b\}$ and define

$$g(I) = \max_{c \in \mathbb{R}^n_+} \frac{\min_{x \in S(A,b)} \sum_{i=1}^n c_i x_i}{\min_{x \in P(A,b)} \sum_{i=1}^n c_i x_i},$$

our main results work. In fact our implementation of the subtour-elimination relaxation is based on an extended formulation with auxiliary variables (see [CKL$^+$07]). We removed this extension to make the presentation simpler.

Our experiments in Section 5 give a proof of concept. They show that its plausible FDT will have practical benefit as an IP heuristic for problems with appropriate structure. FDT performance will likely improve in a future more high-performance version coded in C or C++, perhaps able to take advantage of the low-level parallelism available on all modern platforms (even laptops). This will allow tests on larger instances. We leave as future work a more comprehensive set of experiments to determine on which kinds of problems FDT is likely to outperform other general heuristics. Any consistent structure of such instances could lead to proofs of better approximation bounds.

Fractional Decomposition Tree is a tool to experimentally evaluate the known bounds (and conjectured bounds) on the integrality gap of combinatorial optimization IP formulations. We hope that applying this tool to a wide range of problems can be a tool to guide conjectures on the upper bounds of integrality gap or at least narrow down the instances for which a closer look is required for the study of integrality gap.

References

[ABCC06] David L. Applegate, Robert E. Bixby, Vašek Chvatál, and William J. Cook. *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, 2006.

[ABE06] Anthony Alexander, Sylvia Boyd, and Paul Elliott-Magwood. On the integrality gap of the 2-edge connected subgraph problem. Technical report, TR-2006-04, SITE, University of Ottawa, 2006.

[AKS11] Per Austrin, Subhash Khot, and Muli Safra. Inapproximability of vertex cover and independent set in bounded degree graphs. *Theory of Computing*, 7(3):27–43, 2011.

[AKS15] Hyung-Chan An, Robert Kleinberg, and David B. Shmoys. Improving christofides' algorithm for the s-t path TSP. *J. ACM*, 62(5), November 2015.

[BB08] Geneviève Benoit and Sylvia Boyd. Finding the exact integrality gap for small Traveling Salesman Problems. *Mathematics of Operations Research*, 33(4):921–931, 2008.

[BC11] 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.

[BL17] Sylvia Boyd and Philippe Legault. Toward a 6/5 bound for the minimum cost 2-edge connected spanning subgraph. *SIAM J. Discrete Math.*, 31:632–644, 2017.

[BP90] S Boyd and W. R. Pulleyblank. Optimizing over the subtour polytope of the travelling salesman problem. *Mathematical Programming*, 49:163–187, 1990.

[CCZ14] Michele Conforti, Gerard Cornuejols, and Giacomo Zambelli. *Integer Programming*. Springer Publishing Company, Incorporated, 2014.

[CKKK08] J. Cheriyan, H. Karloff, R. Khandekar, and J. Könemann. On the integrality ratio for tree augmentation. *Operations Research Letters*, 36(4):399 – 401, 2008.

[CKL+07] Robert D. Carr, Goran Konjevod, Greg Little, Venkatesh Natarajan, and Ojas Parekh. Compacting cuts: A new linear formulation for minimum cut. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 43–52, USA, 2007. Society for Industrial and Applied Mathematics.

[CR98] Robert Carr and R. Ravi. A new bound for the 2-edge connected subgraph problem. In *IPCO*, 1998.

[CV04] Robert Carr and Santosh Vempala. On the Held-Karp relaxation for the asymmetric and symmetric traveling salesman problems. *Mathematical Programming*, 100(3):569–587, Jul 2004.

[DFH19] M. Ayaz Dzulfikar, Johannes K. Fichte, and Markus Hecher. The PACE 2019 Parameterized Algorithms and Computational Experiments Challenge: The Fourth Iteration (Invited Paper). In Bart M. P. Jansen and Jan Arne Telle, editors, *14th International Symposium on Parameterized and Exact Computation (IPEC 2019)*, volume 148 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:23, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Edm65] Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards B*, 69:125–130, 1965.

[Edm70] Jack Edmonds. *Submodular Functions, Matroids, and Certain Polyhedra*, pages 11–26. Springer Berlin Heidelberg, Berlin, Heidelberg, 1970.

[FGL05] Matteo Fischetti, Fred Glover, and Andrea Lodi. The feasibility pump. *Mathematical Programming*, 104(1):91–104, Sep 2005.

[FJ81] Greg N. Frederickson and Joseph JáJá. Approximation algorithms for several graph augmentation problems. *SIAM Journal on Computing*, 10(2):270–283, 1981.

[FS09] Matteo Fischetti and Domenico Salvagnin. Feasibility pump 2.0. *Mathematical Programming Computation*, 1(2):201–222, Oct 2009.

[GJ90] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1990.

[GLS93] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2. Second corrected edition edition, 1993.

[Goe95] Michel X. Goemans. Worst-case comparison of valid inequalities for the TSP. *Mathematical Programming*, 69(1):335–349, Jul 1995.

[GW17] Kyle Genova and David P. Williamson. An experimental evaluation of the best-of-many Christofides' algorithm for the traveling salesman problem. *Algorithmica*, 78:1109–1130, 2017.

[HN18] Arash Haddadan and Alantha Newman. Polynomial-time algorithms for 2-edge-connected subgraphs on fundamental classes by top-down coloring. *CoRR*, abs/1811.09906, 2018.

[HT17] Saïd Hanafi and Raca Todosijević. Mathematical programming based heuristics for the 0-1 mip: a survey. *Journal of Heuristics*, 23(4):165–206, Aug 2017.

[Sch03] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.

[SW90] David B. Shmoys and David P. Williamson. Analyzing the held-karp tsp bound: a monotonicity property with application. *Information Processing Letters*, 35(6):281 – 285, 1990.

[Vaz01] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag, Berlin, Heidelberg, 2001.

[Vyg12] Jens Vygen. New approximation algorithms for TSP. *Optima*, 90:1–12, 2012.

[Wol80] Laurence A. Wolsey. *Heuristic analysis, linear programming and branch and bound*, pages 121–134. Springer Berlin Heidelberg, Berlin, Heidelberg, 1980.

[WS11] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, USA, 1st edition, 2011.