Practical Frank–Wolfe Method with Decision Diagrams for Computing Wardrop Equilibrium of Combinatorial Congestion Games

Kengo Nakamura, Shinsaku Sakaue, Norihito Yasuda
NTT Communication Science Laboratories
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0237, Japan
{kengo.nakamura.dx, shinsaku.sakaue.va, norihito.yasuda.hn}@hco.ntt.co.jp

Abstract

Computation of equilibria for congestion games has been an important research subject. In many realistic scenarios, each strategy of congestion games is given by a combination of elements that satisfies certain constraints; such games are called combinatorial congestion games. For example, given a road network with some toll roads, each strategy of routing games is a path (a combination of edges) whose total toll satisfies a certain budget constraint. Generally, given a ground set of n elements, the set of all such strategies, called the strategy set, can be large exponentially in n, and it often has complicated structures; these issues make equilibrium computation very hard. In this paper, we propose a practical algorithm for such hard equilibrium computation problems. We use data structures, called zero-suppressed binary decision diagrams (ZDDs), to compactly represent strategy sets, and we develop a Frank–Wolfe-style iterative equilibrium computation algorithm whose per-iteration complexity is linear in the size of the ZDD representation. We prove that an ϵ-approximate Wardrop equilibrium can be computed in $O(\mathrm{poly}(n)/\epsilon)$ iterations, and we improve the result to $O(\mathrm{poly}(n)\log\epsilon^{-1})$ for some special cases. Experiments confirm the practical utility of our method.

1 Introduction

Congestion games form an important subclass of non-cooperative games since they can model various resource allocation scenarios. Motivated by a wide variety of applications, computation of equilibria for congestion games, which enable us to examine structures of equilibria (e.g., the price of anarchy), has been an attracting research subject. In many practical congestion games, each strategy forms a subset of a finite ground set, $[n] \coloneqq \{1, \dots, n\}$, and the set of all strategies, or the strategy set, can be large exponentially in n. We call such games combinatorial congestion games. A typical example is selfish routing (Roughgarden 2005): Given a graph with edge set $[n]$ and origin-destination pair (s, t), each player chooses an s–t path selfishly to go from s to t as fast as possible. In this case, the strategy set is the collection of all s–t paths, whose size is generally exponential in n. For the standard selfish routing, efficient equilibrium computation methods have been developed (Fabrikant, Papadimitriou, and Talwar 2004; Thai 2017). These methods handle the huge strategy sets by utilizing efficient polynomial-time algorithms for the min-cost flow or shortest path problems. In more general settings, however, strategy sets can have more complicated structures, for which such polynomial-time algorithms are unavailable. Below we list two such examples:

Budgeted Selfish Routing. We consider a variant of selfish routing based on (Jahn et al. 2005): The graph representing a road network has some edges corresponding to toll roads, and players do not choose s–t paths whose total toll is too expensive. Namely, each strategy must satisfy a certain budget constraint. Given a cost value for each edge, which indicates the degree of congestion, to find the minimum-cost s–t path is at least as hard as the NP-hard knapsack problem.

Multi-location Communication. We consider the multi-location communication problem on a network based on (Imase and Waxman 1991). Let $[n]$ be an edge set of a graph representing the communication network. Each edge connects two cities, and there are some special cities called terminals. In this game, a player is a group of people who are in the terminals and have a multi-person conference. Each strategy is a Steiner tree that includes all the terminals, and each player chooses a Steiner tree selfishly to minimize communication delay. Given the degree of delay for each edge, the problem of finding the Steiner tree with the smallest delay reduces to the APX-hard minimum Steiner tree problem.

Congestion games have many other applications related to resource allocation on networks (Shakkottai, Altman, and Kumar 2007; Han et al. 2012), and their combinatorial variants can naturally appear in practice. Therefore, to develop practical equilibrium computation methods that can handle various combinatorial strategy sets is important for advancing studies into real-world congestion games. However, we are currently missing such general methods due to the difficulty of dealing with a wide variety of huge and complicated strategy sets.

1.1 Our Contribution

We develop a practical equilibrium computation algorithm for combinatorial congestion games. Motivated by the fact that many real-world games have a huge number of players, we employ the continuous-player setting (Sandholm 2001), where there are infinitely many players with an infinitesimal mass. As elucidated later, equilibrium computation for

such games can be reduced to constrained potential function minimization problems, which are generally NP-hard due to the complicated structures of strategy sets. We address this hardness by using data structures called zero-suppressed binary decision diagrams (ZDDs), which can represent various combinatorial strategy sets compactly. We minimize potential functions with a Frank–Wolfe-style iterative algorithm that utilizes the compact ZDD representation. In practice, since the algorithm is performed with finite precision, output solutions generally have objective errors. Fortunately, however, we can prove that the solutions achieve an ϵ-approximate Wardrop equilibrium. Below we detail our contributions:

- We propose a Frank–Wolfe-style iterative equilibrium computation algorithm for continuous-player combinatorial congestion games, whose per-iteration complexity is linear in the total size of ZDDs representing strategy sets.
- We prove that our algorithm outputs an ϵ-approximate Wardrop equilibrium in $O(\mathrm{poly}(n)/\epsilon)$ iterations if potential functions have differentiability, convexity, and Lipschitz-continuous gradient. We also improve the result to $O(\mathrm{poly}(n)\log\epsilon^{-1})$ with some additional assumptions.
- We validate the proposed method via experiments. The results demonstrate that our method is useful for practical equilibrium computation problems.

While ZDDs can generally be large exponentially in n, their empirical sizes are often small enough for practical use. Moreover, ZDD sizes can sometimes be polynomial in n (see, Section 2.3). In such cases, our method can compute an ϵ-approximate Wardrop equilibrium in $O(\mathrm{poly}(n)/\epsilon)$ (or $O(\mathrm{poly}(n)\log\epsilon^{-1})$) time.

Our method can also compute the social optimum as in Section 2.1. Therefore, our method can be used for examining the price of anarchy, which is defined by the ratio of the total cost value of the equilibrium to that of the social optimum.

1.2 Related Work

Continuous-player games and their connection to continuous optimization have been widely studied. Beckmann, McGuire, and Winsten (1956) first used a convex optimization formulation to study traffic equilibria. Sandholm (2001) established a general framework of continuous-player games characterized as potential function minimization, called potential games, but combinatorial strategy sets were not considered. As regards the combinatorial setting, many existing works are devoted to selfish routing (Roughgarden 2005): Roughgarden and Tardos (2002) studied the inefficiency of equilibria, or the price of anarchy. Convergence of various dynamics to equilibria were also studied in (Fischer and Vöcking 2004; Blum, Even-Dar, and Ligett 2006; Fischer, Räcke, and Vöcking 2010). Fabrikant, Papadimitriou, and Talwar (2004) developed a polynomial-time algorithm for computing an approximate equilibrium. Optimization-based algorithms, including the Frank–Wolfe algorithm, for equilibrium computation have also been widely studied (Vliet 1987; Bar-Gera 2002; Correa and Stier-Moses 2011; Thai 2017); note that our work is the first that shows the relationship between the precision of a Frank–Wolfe-style algorithm and approximate Wardrop equilibria. Congestion games other than selfish routing have also been studied (Altman et al. 2006; Shakkottai, Altman, and Kumar 2007; Han et al. 2012), but those did not consider addressing the computational hardness caused by the huge and complicated strategy sets. Jahn et al. (2005) addressed a computationally hard problem of finding system-optimal flow for the budgeted routing setting. Their approach is, however, based on a linear integer programming formulation specialized for the budgeted routing setting, while our method can work with various combinatorial strategy sets that can be represented with ZDDs as in Section 2.3.

The Frank–Wolfe algorithm (Frank and Wolfe 1956), a.k.a. conditional gradient algorithm, is a well-established algorithm for constrained convex minimization, and it has recently been attracting much attention thanks to the useful convergence analysis based on the duality gap (Jaggi 2013). Lacoste-Julien and Jaggi (2015) studied several variants of the Frank–Wolfe algorithm and proved their global linear convergence under some assumptions; our method uses one of the variants called the fully-corrective Frank–Wolfe algorithm. Abernethy and Wang (2017) studied the Frank–Wolfe algorithm from a perspective of zero-sum games; note that their research direction is different from ours. As explained in Section 2.2, a typical bottleneck of the Frank–Wolfe algorithm is a linear optimization step. While this bottleneck can be resolved in some cases (Lacoste-Julien et al. 2013; Kerdreux, Pedregosa, and d'Aspremont 2018; Braun, Pokutta, and Zink 2019), these techniques do not work in our case with huge and complicated strategy sets.

Minato (1993) proposed ZDDs as a variant of binary decision diagrams (Bryant 1986). As with our method, many optimization methods that take advantages of decision diagrams have recently been developed (Morrison, Sewell, and Jacobson 2016; Bergman et al. 2016); most of them consider optimizing linear objective functions. To the best of our knowledge, our work provides the first ZDD-based algorithm for constrained optimization with (non-linear) convex objective functions.

2 Background

We introduce our problem settings and basic techniques.

2.1 Problem Settings

Let $[r] \coloneqq \{1, \dots, r\}$ be the set of populations. We consider the combinatorial setting: Given a finite ground set $[n]$, each strategy is given by $S \subseteq [n]$. For each $p \in [r]$, we define strategy set $\mathcal{S}^p \coloneqq \{S_1, \dots, S_{|\mathcal{S}^p|}\} \subseteq 2^{[n]}$. Given any $S \subseteq [n]$, we let $\mathbf{1}_S \in \{0,1\}^n$ denote an indicator vector whose ith entry is 1 if $i \in S$ and 0 otherwise. We sometimes abuse the notation and regard $\mathbf{1}_S$ in the same light as S.

We then explain the continuous-player setting. Each $p \in [r]$ has infinitely many players with an infinitesimal mass; let $m^p \in \mathbb{R}$ be the total mass. We assume $\sum_{p\in[r]} m^p = 1$ w.l.o.g. Let $z^p_S \in [0,1]$ denote the proportion of players who choose strategy $S \in \mathcal{S}^p$; we have $\sum_{S\in\mathcal{S}^p} z^p_S = 1$. Consequently, $m^p z^p_S$ represents the mass of players in p who choose $S \in \mathcal{S}^p$. We call $\boldsymbol{z}^p = (z^p_{S_1}, \dots, z^p_{S_{|\mathcal{S}^p|}})$ a strategy profile of p and $\boldsymbol{z} = (\boldsymbol{z}^1, \dots, \boldsymbol{z}^r)$ a strategy profile.

We finally explain the congestion game setting. Given any strategy profile $\boldsymbol{z}$, we let $\boldsymbol{x}^p = \sum_{S\in\mathcal{S}^p} z_S^p \mathbf{1}_S \in [0,1]^n$ for each $p \in [r]$ and $\boldsymbol{x} = (\boldsymbol{x}^1, \dots, \boldsymbol{x}^r)$. Note that the ith entry of $\boldsymbol{y} \coloneqq \sum_{p\in[r]} m^p \boldsymbol{x}^p$, denoted by y_i, represents the mass of players who choose strategies including $i \in [n]$. For any $p \in [r]$, the cost of strategy $S \in \mathcal{S}^p$ is given by $c_S(\boldsymbol{y}) \coloneqq \sum_{i\in S} c_i(y_i)$, where $c_i(\theta)$ ($\theta \in [0,1]$) is a function that indicates the cost of using $i \in [n]$ when the mass of players using $i \in [n]$ is θ. For convenience, we also define cost functions on $\mathbb{R}^{rn}$ as $C_S(\boldsymbol{x}) \coloneqq c_S(\sum_{p\in[r]} m^p \boldsymbol{x}^p)$. Each player chooses a strategy selfishly to minimize the cost. We define potential function $\Phi : \mathbb{R}^n \to \mathbb{R}$ as $\Phi(\boldsymbol{y}) \coloneqq \sum_{i\in[n]} \int_0^{y_i} c_i(\theta)\mathrm{d}\theta$. Note that we have $c_i(\boldsymbol{y}) = \nabla\Phi(\boldsymbol{y})_i$ and $c_S(\boldsymbol{y}) = \nabla\Phi(\boldsymbol{y})^\top \mathbf{1}_S$. We impose some assumptions on the cost and potential functions as summarized below; particularly, we assume $\Phi(\cdot)$ to be convex. As in (Sandholm 2001), strategy profile $\boldsymbol{z}$ achieves an equilibrium iff it satisfies the KKT condition; thanks to the convexity of $\Phi(\cdot)$, it is characterized as a minimizer of

$$\begin{aligned} \underset{\boldsymbol{z}\geq\mathbf{0}}{\text{minimize}} \quad & \Phi\left(\sum_{p\in[r]} m^p \sum_{S\in\mathcal{S}^p} z_S^p \mathbf{1}_S\right) \\ \text{subject to} \quad & \sum_{S\in\mathcal{S}^p} z_S^p = 1 \quad (\forall p \in [r]). \end{aligned}$$

Since $\boldsymbol{z}$ generally consists of exponentially many variables, to directly solve this problem is too expensive; hence we consider reformulating it. We define $\mathcal{V} \coloneqq \mathcal{S}^1 \times \cdots \times \mathcal{S}^r$. Since the Cartesian products "$\times$" and convex hulls "$\mathrm{conv}(\cdot)$" are commutative, we have $\mathrm{conv}(\mathcal{S}^1) \times \cdots \times \mathrm{conv}(\mathcal{S}^r) = \mathrm{conv}(\mathcal{V})$, and so the above problem can be equivalently rewritten as

$$\begin{aligned} \underset{\boldsymbol{x}=(\boldsymbol{x}^1,\dots,\boldsymbol{x}^r)\in\mathbb{R}^{rn}}{\text{minimize}} \quad & F(\boldsymbol{x}) \coloneqq \Phi\left(\sum_{p\in[r]} m^p \boldsymbol{x}^p\right) \\ \text{subject to} \quad & \boldsymbol{x} \in \mathrm{conv}(\mathcal{V}), \end{aligned} \tag{1}$$

which has rn variables. We aim to solve this problem in what follows. Note that, for any $p \in [r]$, the gradient of $F(\boldsymbol{x})$ w.r.t. $\boldsymbol{x}^p$, denoted by $\nabla_p F(\boldsymbol{x})$, is given by $m^p \nabla\Phi(\boldsymbol{y}) \in \mathbb{R}^n$, where $\boldsymbol{y} = \sum_{p\in[r]} m^p \boldsymbol{x}^p$.

We need some remarks. Although the minimizer, $\boldsymbol{x}$, of (1) indicates the proportion of players choosing $i \in [n]$ for each $p \in [r]$, it does not give us strategy profile $\boldsymbol{z}$. Fortunately, however, the Frank–Wolfe-style algorithms output a solution as a convex combination of vertices, which enable us to obtain a strategy profile (see, Section 3.2). As in (Roughgarden 2005), we can confirm that any minimizer of (1) has the following uniqueness of cost values: Given any two minimizers, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, of (1), we have $c_i(\boldsymbol{y}_1) = c_i(\boldsymbol{y}_2)$ ($\forall i \in [n]$), where $\boldsymbol{y}_j = \sum_{p\in[r]} \boldsymbol{x}_j^p$ ($j = 1, 2$). Therefore, the total cost of an equilibrium, $\sum_{i\in[n]} y_i \cdot c_i(y_i)$, can be uniquely computed by solving (1). Furthermore, computation of the social optimum can be formulated in almost the same manner as (1): We minimize $\Psi(\boldsymbol{y}) \coloneqq \sum_{i\in[n]} y_i \cdot c_i(y_i)$ instead of $\Phi(\boldsymbol{y})$, which our method can (approximately) solve under some assumptions detailed below. The price of anarchy can be computed as the ratio of those two total costs.

Assumptions on Cost and Potential Functions. We assume $c_i(\cdot)$ ($\forall i \in [n]$) to be non-decreasing on $[0,1]$, which implies that $\Phi(\cdot)$ and $F(\cdot)$ are convex on $[0,1]^n$ and $[0,1]^{rn}$, respectively. We also make the following assumptions to guarantee the convergence the Frank–Wolfe-style algorithm. Let $\overline{m} \coloneqq \max_{p\in[r]} m^p$ and $\underline{m} \coloneqq \min_{p\in[r]} m^p$. We assume $\Phi(\cdot)$ to have L-Lipschitz-continuous gradient ($L > 0$) on $[0,1]^n$, which is equivalent to L-Lipschitz-continuity of $c_i : [0,1] \to \mathbb{R}$ ($\forall i \in [n]$); i.e., $|c_i(y_1) - c_i(y_2)| \leq L|y_1 - y_2|$ for any $y_1, y_2 \in [0,1]$. This is a common assumption (see, e.g., (Fabrikant, Papadimitriou, and Talwar 2004)). Note that, if $\Phi(\cdot)$ has L-Lipschitz-continuous gradient, $F(\cdot)$ also has Lipschitz-continuous gradient with constant $L\overline{m} \leq L$. In Theorem 2, we obtain an improved result by assuming $F(\cdot)$ to be strongly convex with some constant $\mu > 0$: $F(\boldsymbol{x}+\boldsymbol{d}) \geq F(\boldsymbol{x}) + \langle \nabla F(\boldsymbol{x}), \boldsymbol{d}\rangle + \frac{\mu}{2}\|\boldsymbol{d}\|_2^2$ for any $\boldsymbol{x}, \boldsymbol{d} \in \mathbb{R}^{rn}$. Note that, if $\Phi(\cdot)$ is μ-strongly convex, then $F(\cdot)$ is also strongly convex with constant $\mu\underline{m}$. Below we present a standard example setting satisfying the above assumptions.

When computing the social optimum, we need additional assumptions as in (Roughgarden 2005). We assume $c_i(\cdot)$ to be differentiable and semi-convex (i.e., $y \cdot c_i(y)$ is convex w.r.t. y) on $[0,1]$, which makes $\Psi(\cdot)$ differentiable and convex. We also assume $\Psi(\cdot)$ to have Lipschitz-continuous gradient.

Example: Budgeted Selfish Routing. Given a graph with edge set $[n]$, edge weights (or tolls) $w_1, \dots, w_n$, origin-destination pairs $(s_1, t_1), \dots, (s_r, t_r)$, and budget value W, each player at s_p selfishly chooses an s_p–t_p path whose total weight is at most W. In this case, each strategy $S \in \mathcal{S}^p$ is an edge subset that forms an s_p–t_p path and satisfies $\sum_{i\in S} w_i \leq W$. If the congestion degree of each edge $i \in [n]$ increases linearly in the amount of traffic, the cost function of $i \in [n]$ is given by $c_i(y_i) = a_i y_i + b_i$ with some $a_i, b_i \geq 0$. The potential function is $\Phi(\boldsymbol{y}) = \sum_{i\in[n]} \int_0^{y_i} c_i(\theta)\mathrm{d}\theta = \sum_{i\in[n]} (\frac{1}{2} a_i y_i^2 + b_i y_i)$, which is differentiable and convex. Note that each $c_i(\cdot)$ is a_i-Lipschitz continuous and that $\Phi(\cdot)$ is strongly convex with constant $\min_{i\in[n]} a_i$. Furthermore, $\Psi(\boldsymbol{y}) = \sum_{i\in[n]} y_i \cdot c_i(y_i) = \sum_{i\in[n]} (a_i y^2 + b_i y)$ is differentiable, convex, and has Lipschitz-continuous gradient with constant $2 \times \max_{i\in[n]} a_i$.

2.2 Frank–Wolfe Algorithm

The Frank–Wolfe algorithm is an effective approach to constrained convex minimization problem $\min_{\boldsymbol{x}\in\mathcal{D}} f(\boldsymbol{x})$, where $\mathcal{D} \subseteq \mathbb{R}^n$ is a convex feasible region and $f(\cdot)$ is a convex objective function. Specifically, given an initial point $\boldsymbol{x}_0 \in \mathcal{D}$, we update it for $k = 0, \dots, K$ as follows:

1. $\boldsymbol{s}_k \in \mathrm{argmin}_{\boldsymbol{v}\in\mathcal{D}} \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{v}\rangle$,
2. $\boldsymbol{x}_{k+1} = (1 - \gamma_k)\boldsymbol{x}_k + \gamma_k \boldsymbol{s}_k$,

where γ_k is usually set to $\frac{2}{k+2}$. It is known that $\boldsymbol{x}_k$ converges to an optimal solution, $\boldsymbol{x}^*$, at a rate of $O(\mathrm{poly}(n)/k)$ if $f(\cdot)$ has Lipschitz-continuous gradient (Frank and Wolfe 1956; Jaggi 2013). Therefore, if the linear optimization problem on $\mathcal{D}$ can be solved, we can obtain a sequence of solutions that converges to $\boldsymbol{x}^*$.

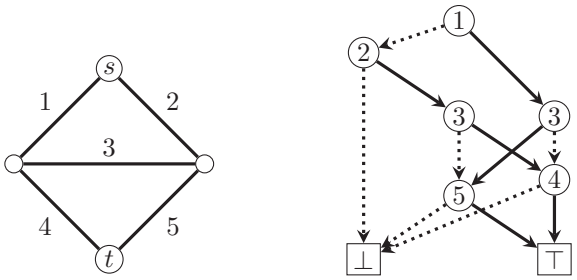


Figure 1: The left figure is a graph with origin s, destination t, and edge set $[n]$, where $n = 5$. Let $\mathcal{S} \subseteq 2^{[n]}$ be the collection of all s–t paths. The right figure illustrates ZDD $\mathsf{Z}_{\mathcal{S}} = (\mathsf{V}, \mathsf{A})$ that represents $\mathcal{S}$. Each node $\mathsf{v} \in \mathsf{V}$ is labeled by $l_{\mathsf{v}} \in [n]$, and the top node labeled by 1 is the root node, $\mathsf{r} \in \mathsf{V}$. Each dashed (solid) arc represents 0-arc (1-arc). The square node labeled with $\bot$ ($\top$) is 0-terminal (1-terminal).

Frank–Wolfe-style algorithms require to solve the linear optimization problem time and again as in Step 1, and thus how to solve it heavily affects the efficiency of the algorithms. For the case of standard selfish routing, the linear optimization can be solved efficiently with algorithms for finding the shortest path, but this is not always the case with other settings. For example, if $\mathcal{S}^p$ ($p \in [r]$) is a collection of Steiner trees on a given graph with terminals, to solve the linear optimization problem on $\mathcal{D} = \mathrm{conv}(\mathcal{V})$ is at least as difficult as the APX-hard minimum Steiner tree problem. Note that, while the Frank–Wolfe algorithm accepts additive errors when solving the linear optimization (Jaggi 2013), the error value must converge to 0 as k increases. To achieve this is still difficult particularly when the linear optimization is APX-hard, which is true if $\mathcal{S}^p$ is the collection of Steiner trees. Jaggi (2013) also mentioned the difficulty of performing the Frank–Wolfe algorithm for the case where the linear optimization step is NP-hard. In Section 3.1, we address this hardness by using ZDDs.

2.3 Zero-suppressed Binary Decision Diagrams

Given a set family, $\mathcal{S} \subseteq 2^{[n]}$, ZDD $\mathsf{Z}_{\mathcal{S}} = (\mathsf{V}, \mathsf{A})$ is a DAG-shaped data structure that compactly represents $\mathcal{S}$. The node set, V, contains one root node r and two terminal nodes $\bot$ and $\top$, called 0- and 1-terminals, respectively. Each non-terminal node $\mathsf{v} \in \mathsf{V} \backslash \{\bot, \top\}$ is labeled by $l_{\mathsf{v}} \in [n]$ and has two outgoing arcs, 0- and 1-arcs. We define 0-child $\mathsf{c}^0_{\mathsf{v}}$ (1-child $\mathsf{c}^1_{\mathsf{v}}$) of v as a node pointed by the 0-arc (1-arc) outgoing from v. We define $\mathcal{R} \subseteq 2^{\mathsf{A}}$ as the set of all directed paths connecting r to $\top$. Given $\mathsf{R} \in \mathcal{R}$, we define $X(\mathsf{R}) \coloneqq \{l_{\mathsf{v}} \in [n] \mid (\mathsf{v}, \mathsf{c}^1_{\mathsf{v}}) \in \mathsf{R}\}$. We have $\mathcal{S} = \{X(\mathsf{R}) \mid \mathsf{R} \in \mathcal{R}\}$, which means there is one-to-one correspondence between $S \in \mathcal{S}$ and $\mathsf{R} \in \mathcal{R}$. In other words, each $S \in \mathcal{S}$ is represented as a directed path, $\mathsf{R} \in \mathcal{R}$, from r to $\top$; we can recover S from R by taking label $l_{\mathsf{v}} \in [n]$ assigned to the tail of every 1-arc in R. Figure 1 illustrates an example of a ZDD for the case where $\mathcal{S}$ is a collection of s–t paths; e.g., an s–t path $\{1, 4\} \in \mathcal{S}$ corresponds to a directed path whose arcs are labeled by $\{(1, 3), (3, 4), (4, \top)\}$ in the ZDD.

As regards the ZDD construction, various efficient methods have been developed. In particular, the frontier-based search (Kawahara et al. 2017), which is based on the Knuth's Simpath algorithm (Knuth 2011), is known to be effective when $\mathcal{S}$ is specified on graphs (e.g., s–t paths, Steiner trees, matchings, and cliques). By utilizing the well-established family algebra (Minato 1993; Knuth 2011), we can also construct ZDDs representing logically constrained strategy sets. For example, given a graph with edge costs and multiple budget constraints, we can construct ZDDs representing the set of all s–t paths satisfying at least one of the budget constraints. The ZDD size, $|\mathsf{V}|$, and the complexity of its construction are generally exponential in n, but its empirical size is often small enough for practical use. Moreover, if $\mathcal{S}$ is defined on a graph with a constant pathwidth, the resulting ZDD size and construction complexity can be $\mathrm{poly}(n)$ (Inoue and Minato 2016; Kawahara et al. 2017).

Algorithm 1 FCFW with ZDDs for equilibrium computation

1: Construct $\mathsf{Z}_{\mathcal{S}^1}, \ldots, \mathsf{Z}_{\mathcal{S}^r}$.
2: Choose $S^p \in \mathcal{S}^p$ for $p \in [r]$.
3: Let $\boldsymbol{x}^p_0 = \mathbf{1}_{S^p}$, $\mathcal{A}^p_0 = \{S^p\}$, and $z^p_{S^p} = 1$ for $p \in [r]$.
4: $\boldsymbol{x}_0 = (\boldsymbol{x}^1_0, \ldots, \boldsymbol{x}^r_0)$.
5: **for** $k = 0, \ldots, K$ **do**
6: $\quad \boldsymbol{y}_k = \sum_{p \in [r]} m^p \boldsymbol{x}^p_k$.
7: $\quad$ **for each** $p \in [r]$ **do**
8: $\quad\quad$ Compute $\boldsymbol{s}^p_k \in \mathrm{argmin}_{\boldsymbol{u} \in \mathcal{S}^p} \langle \nabla \Phi(\boldsymbol{y}_k), \boldsymbol{u} \rangle$ by DP on $\mathsf{Z}_{\mathcal{S}^p}$.
9: $\quad\quad$ Let $\boldsymbol{d}^p_k = \boldsymbol{s}^p_k - \boldsymbol{x}^p_k$ and $g^p_k = \langle -\nabla \Phi(\boldsymbol{y}_k), \boldsymbol{d}^p_k \rangle$.
10: $\quad$ **end for**
11: $\quad$ **if** $\max_{p \in [r]} g^p_k \leq \epsilon$ **then**
12: $\quad\quad$ **return** $\boldsymbol{x}_k$, $\{\mathcal{A}^p_k\}_{p \in [r]}$, and $\{\{z^p_S\}_{S \in \mathcal{A}^p_k}\}_{p \in [r]}$.
13: $\quad$ **end if**
14: $\quad$ Execute **Correction**$(\boldsymbol{x}_k, \{\mathcal{A}^p_k\}_{p \in [r]}, \{\boldsymbol{s}^p_k\}_{p \in [r]}, \epsilon)$ to get $\boldsymbol{x}_{k+1} = (\boldsymbol{x}^1_{k+1}, \ldots, \boldsymbol{x}^r_{k+1})$, $\{\mathcal{A}^p_{k+1}\}_{p \in [r]}$, and $\{\{z^p_S\}_{S \in \mathcal{A}^p_{k+1}}\}_{p \in [r]}$.
15: **end for**

3 Proposed Method

A high-level sketch of our method is as follows: We first construct ZDD $\mathsf{Z}_{\mathcal{S}^p}$ that represent $\mathcal{S}^p$ for each $p \in [r]$, and then we perform the fully corrective Frank–Wolfe algorithm (FCFW) by utilizing the ZDD representation of the strategy sets. Our algorithm, described in Algorithm 1, consists of two main building blocks: Linear optimization with ZDDs (Step 8) and update of solutions (Step 14), called full correction, which we detail in Sections 3.1 and 3.2, respectively. Section 3.2 also explains how to obtain a strategy profile, and we prove that the obtained solution achieves an ϵ-approximate Wardrop equilibrium in Section 3.3.

3.1 Linear Optimization with ZDDs

In Step 8, we need to solve $\boldsymbol{s}^p_k \in \mathrm{argmin}_{\boldsymbol{u} \in \mathcal{S}^p} \langle \nabla \Phi(\boldsymbol{y}_k), \boldsymbol{u} \rangle$ for each $p \in [r]$. Note that, since $\langle \nabla \Phi(\boldsymbol{y}_k), \boldsymbol{u} \rangle$ is linear w.r.t. $\boldsymbol{u}$ and $\nabla_p F(\boldsymbol{x}_k) = m^p \nabla \Phi(\boldsymbol{y}_k)$ holds, we can rewrite the linear optimization as $\boldsymbol{s}^p_k \in \mathrm{argmin}_{\boldsymbol{u} \in \mathrm{conv}(\mathcal{S}^p)} \langle \nabla_p F(\boldsymbol{x}_k), \boldsymbol{u} \rangle$; hence $\boldsymbol{s}_k \coloneqq (\boldsymbol{s}^1_k, \ldots, \boldsymbol{s}^r_k) \in \mathrm{argmin}_{\boldsymbol{v} \in \mathrm{conv}(\mathcal{V})} \langle \nabla F(\boldsymbol{x}_k), \boldsymbol{v} \rangle$

Algorithm 2 Correction$(\boldsymbol{x}_k, \{\mathcal{A}_k^p\}_{p\in[r]}, \{\boldsymbol{s}_k^p\}_{p\in[r]}, \epsilon)$

Find $\boldsymbol{x}_{k+1} = (\boldsymbol{x}_{k+1}^1, \dots, \boldsymbol{x}_{k+1}^r)$, $\{\mathcal{A}_{k+1}^p\}_{p\in[r]}$, and $\{\{z_S^p\}_{S\in\mathcal{A}_{k+1}^p}\}_{p\in[r]}$ satisfying the following conditions:

1. $\boldsymbol{x}_{k+1}^p = \sum_{S\in\mathcal{A}_{k+1}^p} z_S^p \mathbf{1}_S$ for $p \in [r]$.
2. $F(\boldsymbol{x}_{k+1}) \leq \min_{\gamma\in[0,1]} F(\boldsymbol{x}_k + \gamma(\boldsymbol{s}_k - \boldsymbol{x}_k))$, where $\boldsymbol{s}_k = (\boldsymbol{s}_k^1, \dots, \boldsymbol{s}_k^r)$.
3. $\max_{p\in[r]} \max_{\boldsymbol{u}\in\mathcal{A}_{k+1}^p} \left\langle -\nabla\Phi(\boldsymbol{y}_{k+1}), \boldsymbol{x}_{k+1}^p - \boldsymbol{u}\right\rangle \leq \epsilon$, where $\boldsymbol{y}_{k+1} = \sum_{p\in[r]} m^p \boldsymbol{x}_{k+1}^p$.

holds, which implies that Step 8 can be seen as the linear optimization step of the Frank–Wolfe algorithm in Section 2.2. Since this linear optimization problem is generally hard to solve as mentioned in Section 2.2, we consider performing this step efficiently by utilizing ZDDs.

Given $\mathsf{Z}_{\mathcal{S}^p} = (\mathsf{V}^p, \mathsf{A}^p)$, $\boldsymbol{s}_k^p \in \operatorname{argmin}_{\boldsymbol{u}\in\mathcal{S}^p} \langle\nabla\Phi(\boldsymbol{y}_k), \boldsymbol{u}\rangle$ can be solved by the standard dynamic programming (DP) on $\mathsf{Z}_{\mathcal{S}^p}$ as follows. We set the length of every 0-arc to 0 and that of 1-arc $(\mathsf{v}, \mathsf{c}_\mathsf{v}^1)$ to $\nabla\Phi(\boldsymbol{y}_k)_{l_\mathsf{v}}$; consequently, the total length of a directed r–$\top$ path, $\mathsf{R} \subseteq \mathsf{A}^p$, is equal to $\sum_{i\in X(\mathsf{R})} \nabla\Phi(\boldsymbol{y}_k)_i = \left\langle\nabla\Phi(\boldsymbol{y}_k), \mathbf{1}_{X(\mathsf{R})}\right\rangle$. Therefore, thanks to the one-to-one correspondence shown in Section 2.3, we can obtain $\boldsymbol{s}_k^p$ by performing DP on $\mathsf{Z}_{\mathcal{S}^p}$ for finding the shortest r–$\top$ path. This ZDD-based linear optimization is suitable as a subroutine of Frank–Wolfe-style algorithms since, once ZDDs are constructed, we can reuse them for $k = 0, \dots, K$. To the best of our knowledge, this is the first Frank–Wolfe-style algorithm that takes advantages of ZDDs. The complexity of Step 8, which is the most expensive step, is linear in the total size of ZDDs: $\sum_{p\in[r]} |\mathsf{V}^p|$. If every $\mathcal{S}^p$ is defined on a graph with a constant pathwidth and $|\mathsf{V}^p|$ is polynomial in n as mentioned in Section 2.3, the per-iteration complexity of our method is also polynomial in n.

3.2 Full Correction

We here detail Step 14, which computes $\boldsymbol{x}_{k+1}$ via full correction (Algorithm 2). For each $p \in [r]$, $\mathcal{A}_k^p \subseteq \mathcal{S}^p$ represents a subset of strategies such that $\boldsymbol{x}_k^p = \sum_{S\in\mathcal{A}_k^p} z_S^p \mathbf{1}_S$ with some $z_S^p \in (0, 1]$; we call $\mathcal{A}_k^p$ the active set. Note that, if $\boldsymbol{x}_k$ is the output solution, the strategy profile of each $p \in [r]$ (or non-zero entries of $\boldsymbol{z}^p \in [0,1]^{|\mathcal{S}^p|}$) is obtained as $\{z_S^p\}_{S\in\mathcal{A}_k^p}$.

Roughly speaking, the full-correction step minimizes $F(\cdot)$ on $\operatorname{conv}(\{\mathcal{A}_k^1 \cup \{\boldsymbol{s}_k^1\}\} \times \cdots \times \{\mathcal{A}_k^r \cup \{\boldsymbol{s}_k^r\}\})$, which can be formulated as convex minimization on a probabilistic simplex. An implementation of a similar full-correction step is presented in (Krishnan, Lacoste-Julien, and Sontag 2015), which uses a Frank–Wolfe-style algorithm called the away-steps Frank–Wolfe, and our full-correction step can be implemented by modifying it as detailed in the appendix. With the implementation of the full-correction step, we have $|\mathcal{A}_k^p| \leq k$. Therefore, while the size of a strategy profile is generally exponential in n, the strategy profile obtained by Algorithm 1 for each $p \in [r]$ is at most as large as the number of iterations, which can be bounded as in Theorem 2.

The fully corrective update of the current solution is more expensive than that of the standard Frank–Wolfe algorithm (Step 2 in Section 2.2). However, thanks to this full-correction step, FCFW achieves a better convergence result for strongly convex objective functions, which we will use in the proof of Theorem 2. Empirically, although the full-correction step increases the per-iteration computation time, it decreases the number of iterations by a great margin, which often reduces the total computation time. In our case, FCFW is a suitable choice because the full-correction step is typically far cheaper than the linear optimization with ZDDs, which means it is effective to reduce the number of iterations via full correction. Furthermore, thanks to the third condition of Algorithm 2, we can prove that the output solution attains an ϵ-approximate Wardrop equilibrium as shown below.

3.3 Approximate Wardrop Equilibrium

Let $\boldsymbol{x} = (\boldsymbol{x}^1, \dots, \boldsymbol{x}^p)$ be the obtained solution. For each $p \in [r]$, we let $\mathcal{A}^p$ be the active set and $\{z_S^p\}_{S\in\mathcal{A}^p}$ be the strategy profile; we have $\boldsymbol{x}^p = \sum_{S\in\mathcal{A}^p} z_S^p \mathbf{1}_S$. If our algorithm is performed with infinite precision ($\epsilon = 0$), the output attains the following Wardrop equilibrium for every $p \in [r]$: $C_S(\boldsymbol{x}) \leq C_{S'}(\boldsymbol{x})$ for any $S \in \mathcal{A}^p$ and $S' \in \mathcal{S}^p$. Namely, no player has an incentive to change his/her strategy. Unfortunately, since $\epsilon = 0$ is impossible in practice, the obtained solution does not always satisfy the above condition. Fortunately, however, we can show that the output of Algorithm 1 achieves an ϵ-approximate Wardrop equilibrium as follows:

Theorem 1. *Fix any $p \in [r]$. Algorithm 1 outputs $\boldsymbol{x}$ satisfying $C_S(\boldsymbol{x}) - \epsilon \leq C_{S'}(\boldsymbol{x}) + \epsilon$ for any $S \in \mathcal{A}^p$ and $S' \in \mathcal{S}^p$.*

Proof. Let $\boldsymbol{y} = \sum_{p\in[r]} m^p \boldsymbol{x}^p$. Thanks to Step 11, we have

$$\langle\nabla\Phi(\boldsymbol{y}), \boldsymbol{x}^p\rangle \leq \min_{\boldsymbol{u}\in\mathcal{S}^p} \langle\nabla\Phi(\boldsymbol{y}), \boldsymbol{u}\rangle + \epsilon = \min_{S'\in\mathcal{S}^p} C_{S'}(\boldsymbol{x}) + \epsilon.$$

From the third condition in Algorithm 2, we obtain

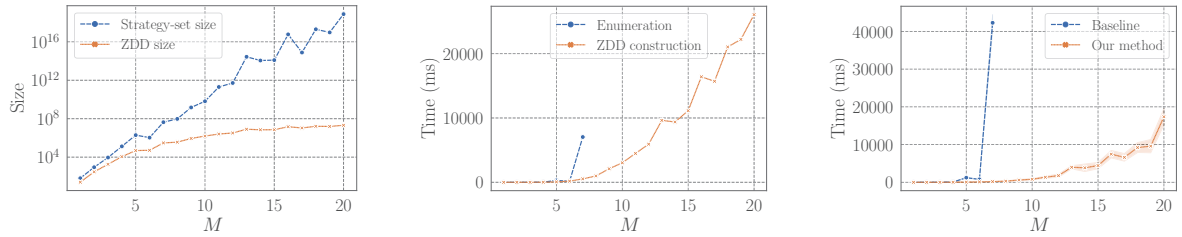
$$\langle\nabla\Phi(\boldsymbol{y}), \boldsymbol{x}^p\rangle \geq \max_{\boldsymbol{u}\in\mathcal{A}^p} \langle\nabla\Phi(\boldsymbol{y}), \boldsymbol{u}\rangle - \epsilon = \max_{S\in\mathcal{A}^p} C_S(\boldsymbol{x}) - \epsilon.$$

Combining these inequalities, we obtain the claim. □

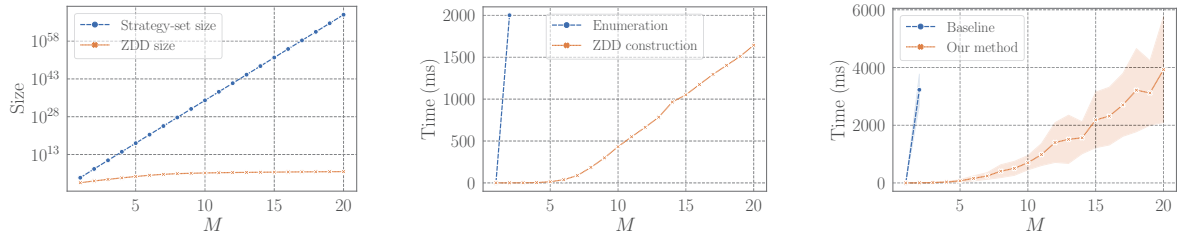
The number of iterations required for computing the ϵ-approximate Wardrop equilibrium can be bounded as follows:

Theorem 2. *Algorithm 1 stops after $O(\mathrm{poly}(n)/\epsilon)$ iterations. If $F(\cdot)$ is strongly convex and the reciprocal of the pyramidal width (Lacoste-Julien and Jaggi 2015) of $\operatorname{conv}(\mathcal{V})$ is $O(\mathrm{poly}(n))$, it stops after $O(\mathrm{poly}(n)\log\epsilon^{-1})$ iterations.*

Proof. If $F(\cdot)$ has differentiability, convexity, and Lipschitz-continuous gradient, the $\max_{\boldsymbol{v}\in\operatorname{conv}(\mathcal{V})} \langle\nabla F(\boldsymbol{x}_k), (\boldsymbol{x}_k - \boldsymbol{v})\rangle$ value, so-called the duality gap, is bounded by $O(\mathrm{poly}(n)/k)$ as in (Jaggi 2013). This fact implies that $\sum_{p\in[r]} m^p g_k^p \leq \epsilon$ holds after $k = O(\mathrm{poly}(n)/\epsilon)$ iterations. Since each g_k^p is non-negative, we see that the algorithm stops after at most $\lceil k/\min_{p\in[r]} m^p\rceil = O(\mathrm{poly}(n)/\epsilon)$ iterations. We then obtain the improved result under the assumptions. Note that the third condition in Algorithm 2 implies $\max_{\boldsymbol{v}\in\operatorname{conv}(\mathcal{A}_{k+1}^1\times\cdots\times\mathcal{A}_{k+1}^r)} \left\langle -\nabla F(\boldsymbol{x}_{k+1}), \boldsymbol{x}_{k+1} - \boldsymbol{v}\right\rangle \leq \epsilon$ since $\sum_{p\in[r]} m^p = 1$, which means that the so-called away gap is less than or equal to ϵ (see, (Lacoste-Julien and Jaggi



(a) BSR, Sizes (semi-log) (b) BSR, Construction times (c) BSR, FCFW times



(d) BSR, Sizes (semi-log) (e) MC, Construction times (f) MC, FCFW times

Figure 2: Figures 2a to 2c show the results of BSR instances, and Figures 2d to 2f show those of MC instances. Figures 2a and 2d indicate the sizes of strategy sets and ZDDs; the strategy-set sizes are computed by DP on ZDDs. Figures 2b and 2e show the times required for constructing $\mathsf{Z}_{\mathcal{S}}$ and enumerating all $S \in \mathcal{S}$, where the enumerated strategy sets are used by the baseline method. Figures 2c and 2f present the running times of FCFW (Algorithm 1) performed with the baseline and our methods, where each curve and error band indicate the mean and standard deviation, respectively, over 50 random instances.

2015)). Consequently, the procedures of Algorithms 1 and 2 completely recover those of FCFW, and so, if the reciprocal of the pyramidal width is $O(\mathrm{poly}(n))$, the duality gap can be bounded by $O(\exp(-\Theta(k/\mathrm{poly}(n))))$ as in (Lacoste-Julien and Jaggi 2015). Therefore, the algorithm stops after $O(\mathrm{poly}(n)\log\epsilon^{-1})$ iterations. □

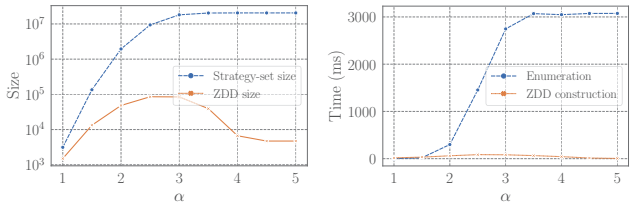
Lacoste-Julien and Jaggi (2015) conjectured that the pyramidal width is lower bounded by $\Omega(1/\sqrt{n})$ in general. Under the conjecture, its reciprocal is $O(\sqrt{n})$, and so the corresponding assumption in Theorem 2 is satisfied.

4 Experiments

We evaluate the proposed method via experiments. In Section 4.1, we show the empirical efficiency of our method with synthetic instances. In Section 4.2, we demonstrate that our method can work with real-world instances. For constructing ZDDs, we use Graphillion (Inoue et al. 2016). All the algorithms are implemented in C++, and the codes are complied with clang++ (Apple LLVM v10.0.0). We set the ϵ value of our algorithm to 10^{-10}. All the experiments are conducted on a 64-bit macOS (High Sierra) machine with 2.5 GHz Intel Core i7 CPU (1 thread) and 16 GB RAM.

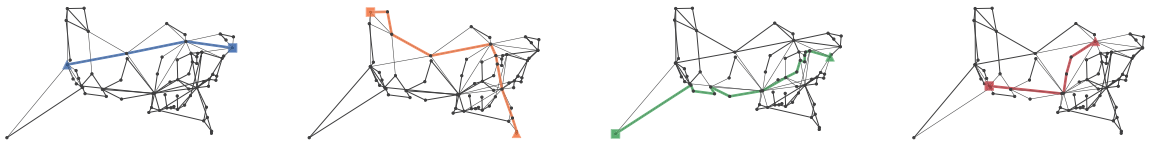
4.1 Synthetic Instances

We consider budgeted selfish routing (BSR) and multi-location communication (MC) instances on undirected grid graphs with $6 \times M$ edges ($M = 1, 2, \ldots, 20$); $[n]$ is the edge set with $n = 13M + 6$. For simplicity, we consider the symmetric case ($r = 1$), and we omit index $p \in [r]$; e.g., $\mathcal{S}^1$ is denoted by $\mathcal{S}$. In BSR instances, each $S \in \mathcal{S}$ is an s–t path satisfying a budget constraint, $\sum_{i \in S} w_i \leq W$, where s and t are placed on the diagonal corners of the grid. Edge weight w_i ($i \in [n]$) is chosen uniformly at random from $\{0, 1, \ldots, 10\}$, and we let $W = 5\alpha \times (6 + M)$; note that, if $\alpha = 1$, W corresponds to the total weight of the shortest s–t path on average. The choice of α affects the sizes of $\mathcal{S}$ and $\mathsf{Z}_{\mathcal{S}}$. Figure 3a presents their sizes for the case with $M = 5$ and $\alpha = 1.0, 1.5, \ldots, 5.0$, where $\mathcal{S}$ includes all s–t paths when $\alpha \geq 4.5$. In Figure 3b, we also present the times required for constructing $\mathsf{Z}_{\mathcal{S}}$ and enumerating all $S \in \mathcal{S}$. Note that, while $|\mathcal{S}|$ increases with α, this is not always the case with $|\mathsf{Z}_{\mathcal{S}}|$, implying that our ZDD-based method can be applied to BSR instances with large W values. In the following experiments, we let $\alpha = 2$. In MC instances, $\mathcal{S}$ is the collection of all Steiner trees connecting four terminals placed on the corners of the grid. With both BSR and MC instances, we use cost

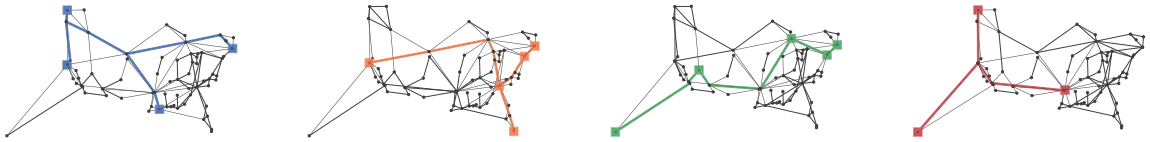


(a) Sizes (b) Construction times

Figure 3: Figure 3a shows the strategy-set and ZDD sizes for the BSR instances with $M = 5$ and $\alpha = 1.0, 1.5, \ldots, 5.0$. Figure 3b indicates the times required for constructing $\mathsf{Z}_{\mathcal{S}}$ and enumerating all $S \in \mathcal{S}$.



(a) BSR, $p = 1$ (b) BSR, $p = 2$ (c) BSR, $p = 3$ (d) BSR, $p = 4$



(e) MC, $p = 1$ (f) MC, $p = 2$ (g) MC, $p = 3$ (h) MC, $p = 4$

Figure 4: Each figure illustrates the topology of TW Telecom network. Figures 4a to 4d: For each $p = 1, \dots, 4$, the square and triangular vertices indicate s_p and t_p, respectively, and the colored (bold) s_p–t_p path is the most-used strategy. Figures 4e to 4h: For each $p = 1, \dots, 4$, the square vertices indicate the terminals, and the colored (bold) Steiner tree is the most-used strategy.

Table 1: Strategy-set sizes, ZDD sizes, and ZDD construction times for real-world instances.

	p	Strategy-set size	ZDD size	Time (ms)
BSR	1	9.099×10^5	3.925×10^4	2340
	2	2.725×10^6	8.382×10^4	4474
	3	6.219×10^5	3.504×10^4	2565
	4	9.426×10^4	9.941×10^3	485.9
MC	1	9.796×10^{28}	9.315×10^4	121.5
	2	4.286×10^{28}	7.930×10^4	81.04
	3	8.462×10^{28}	8.315×10^4	102.5
	4	9.281×10^{28}	8.798×10^4	152.0

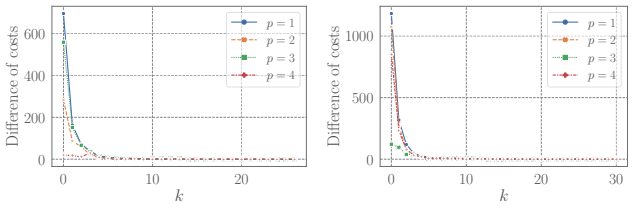
functions defined as $c_i(\theta) = a_i\theta + 1$ ($i \in [n]$), where a_i is drawn uniformly at random from $[0, 10]$. We make 50 random copies of cost functions, and all results that can vary with the objective function (i.e., those related to FCFW) are shown by the mean and standard deviation calculated over the 50 random instances. As a baseline method, we employ the following enumeration-based algorithm: We first enumerate all $S \in \mathcal{S}$ and then execute FCFW (Algorithm 1) whose linear optimization step (Step 8) is performed by choosing the best one from the enumerated $\mathcal{S}$. By comparing our ZDD-based method with the baseline, we examine how much the use of ZDD makes the algorithm efficient.

Figure 2 presents the results. The baseline method does not work with BSR and MC instances with $M \geq 8$ and $M \geq 3$, respectively, due to memory shortage; note that the strategy-set sizes of those instances are calculated by using ZDDs. We see that ZDDs are far smaller than strategy sets, and ZDD construction can be performed far more efficiently than the enumeration of strategies. Thanks to the compactness of ZDDs, our method runs far faster than the baseline method. Our method is about 236 and 984 times faster than the baseline in the BSR instance with $M = 7$ and MC instance with $M = 2$, respectively.

4.2 Real-world Instances

We consider BSR and MC instances on a real-world network. We use TW Telecom dataset of Internet Topology Zoo (Knight et al. 2011), which is a U.S. communication network. The original network has some isolated vertices and multiple edges; we remove them and obtain a graph with 71 vertices and 115 edges. Figure 4 presents the topology of the graph. In both BSR and MC instances, we let $r = 4$ and $(m^1, m^2, m^3, m^4) = (0.4, 0.3, 0.2, 0.1)$, and we use cost function $c_i(\theta) = a_i\theta^2 + b_i$; i.e., the cost increases quadratically in the mass of players. We let b_i be the Euclid distance of edge i and $a_i = b_i u_i$, where u_i is drawn uniformly at random from $[0, 100]$. In the BSR instance, $(s_1, t_1), \dots, (s_4, t_4)$ are placed as in Figures 4a to 4d. We set edge weight w_i at $\lfloor b_i(100 - u_i) \rfloor$. For each $p = 1, \dots, 4$, budget value W^p is set at $100L^p$, where L^p is the length of the shortest s_p–t_p path w.r.t. edge length b_i. In the MC instances, the terminals are placed as in Figures 4e to 4h.

We apply our method to the instances and study the obtained results. Table 1 presents the strategy-set size, ZDD size, and ZDD construction time for each $p = 1, \dots, 4$; notably, with the MC instance, ZDDs are about 10^{24} times smaller than the strategy sets. Additionally, FCFW takes 2391 ms and 3316 ms for BSR and MC instances, respectively. In total, our method computes approximate equilibria in 1.226×10^4 ms and 3773 ms for BSR and MC instances, respectively. Since our method can output a strategy profile, we can obtain the most-used strategy for each $p = 1, \dots, 4$ as in Figure 4. We see that the strategy of each p tends to avoid using the same edges (resource) to each other. For example, in Figures 4g and 4h, the leftmost vertex is chosen as a terminal, which has two edges, and each strategy in $p = 3$ and 4 use one of the two edges that is different from each other. This result implies that the players successfully avoid congestion at the equilibria, and so the price-of-anarchy (PoA) values are expected to be close to 1; i.e., the equilibria are almost as efficient as the social optima. In fact, the PoA values of the BSR and MC instances are both about 1.01, which are obtained by computing the social optima with our method. Figure 5 presents the



(a) BSR (b) MC

Figure 5: $\max_{S\in\mathcal{A}_k^p} C_S(\boldsymbol{x}_k) - \min_{S'\in\mathcal{S}^p} C_{S'}(\boldsymbol{x}_k)$ values ($p = 1, \dots, 4$) for real-world instances.

decrease in $\max_{S\in\mathcal{A}_k^p} C_S(\boldsymbol{x}_k) - \min_{S'\in\mathcal{S}^p} C_{S'}(\boldsymbol{x}_k)$ values over the iterations for $p = 1, \dots, 4$, which converge to 0 at a rate of $O(\mathrm{poly}(n)/k)$ (or $O(\exp(-\Theta(k/\mathrm{poly}(n))))$) as in Theorems 1 and 2. Consistent with the theoretical results, the values converge to 0 very quickly as k increases.

5 Conclusion

We developed a practical Frank–Wolfe-style equilibrium computation method for continuous-player combinatorial congestion games, which utilizes the empirical compactness of ZDDs. We proved that the algorithm computes an ϵ-approximate Wardrop equilibrium in $O(\mathrm{poly}(n)/\epsilon)$ (or $O(\mathrm{poly}(n)\log\epsilon^{-1})$) iterations. Experiments demonstrated that our algorithm is useful for computing and studying equilibria of realistic continuous-player combinatorial congestion games, for which alternative methods are prohibitively costly.

Appendix: Implementation of Full-correction

We here detail the implementation of the full-correction step (Algorithm 2). As mentioned before, the full correction **Correction**$(\boldsymbol{x}, \{\mathcal{A}^p\}_{p\in[r]}, \{\boldsymbol{s}^p\}_{p\in[r]}, \epsilon)$ is, roughly speaking, performed by minimizing $F(\cdot)$ on $\mathrm{conv}((\mathcal{A}^1\cup\{\boldsymbol{s}^1\})\times\cdots\times(\mathcal{A}^r\cup\{\boldsymbol{s}^r\}))$. Since this feasible region is again a convex hull of some points in $\mathbb{R}^{rn}$, the full correction can also be performed with Frank–Wolfe-style algorithms. For example, Krishnan, Lacoste-Julien, and Sontag (2015) implemented the full-correction step with the away-steps Frank–Wolfe (AFW) (Lacoste-Julien and Jaggi 2015), a variant of the Frank–Wolfe algorithm that achieves linear convergence for strongly convex functions. More precisely, they performed the full correction with AFW whose stopping criterion requires the away gap, as well as the Frank–Wolfe gap (or the duality gap), to be small enough; thus they guaranteed that the conditions required in the full-correction step of (Lacoste-Julien and Jaggi 2015, Algorithm 4) is satisfied. The conditions required by our full correction (Algorithm 2) are analogous to those of (Lacoste-Julien and Jaggi 2015, Algorithm 4), but they have a slight difference. Below we detail the difference and explain how to implement the full-correction step that works for our case.

In the original full-correction step of (Lacoste-Julien and Jaggi 2015, Algorithm 4), the away gap is defined on the full dimension, which corresponds to $\mathbb{R}^{rn}$ in our case, and thus the aforementioned AFW-based full correction performed on $\mathbb{R}^{rn}$ works. In our case, however, the third condition of Algorithm 2 requires the away gap $\max_{\boldsymbol{u}\in\mathcal{A}_{k+1}^p}\left\langle -\nabla\Phi(\boldsymbol{y}_{k+1}), \boldsymbol{x}_{k+1}^p - \boldsymbol{u}\right\rangle$ to be small enough for every $p \in [r]$; this is the difference to be considered. Note that, since the "max" is taken for each $p \in [r]$, the away gap of each $p \in [r]$ is not guaranteed to be small no matter how small the away gap on $\mathbb{R}^{rn}$ is; this is the reason why the original full correction does not work. Thus, we consider modifying the AFW algorithm to make it work for our use. The pseudocode of our AFW-based full correction is presented in Algorithm 3. The differences from the original AFW-based full correction (Krishnan, Lacoste-Julien, and Sontag 2015) is as follows: We maintain the active set $\mathcal{A}_l^p$ for each $p \in [r]$, instead of the one defined on $\mathbb{R}^{rn}$, and the away direction, $\boldsymbol{d}_l^{p,\mathrm{A}}$, is computed for each $p \in [r]$ in Step 9. Note that, as with the original full correction of (Krishnan, Lacoste-Julien, and Sontag 2015), we employ the stopping criterion that requires the away and Frank–Wolfe gaps to be small (Step 11). The other parts are almost the same as those

Algorithm 3 Modified AFW-based full correction: **AFWCorrection**$(\boldsymbol{x}, \{\mathcal{A}^p\}_{p\in[r]}, \{\boldsymbol{s}^p\}_{p\in[r]}, \epsilon)$

1: $\boldsymbol{x}_0 = \boldsymbol{x} = (\boldsymbol{x}^1, \dots, \boldsymbol{x}^r)$.
2: **for each** $p \in [r]$ **do**
3: $\quad\mathcal{B}^p = \mathcal{A}^p \cup \{\boldsymbol{s}^p\}$ and $\mathcal{A}_0^p = \mathcal{A}^p$.
4: **end for**
5: **for** $l = 0, \dots, L$ **do**
6: $\quad\boldsymbol{y}_l = \sum_{p\in[r]} m^p \boldsymbol{x}_l^p$.
7: $\quad$**for each** $p \in [r]$ **do**
8: $\quad\quad$Let $\boldsymbol{s}_l^p \in \mathrm{argmin}_{\boldsymbol{u}\in\mathcal{B}^p}\langle\nabla\Phi(\boldsymbol{y}_l), \boldsymbol{u}\rangle$ and $\boldsymbol{d}_l^{p,\mathrm{FW}} = \boldsymbol{s}_l^p - \boldsymbol{x}_l^p$.
9: $\quad\quad$Let $\boldsymbol{v}_l^p \in \mathrm{argmax}_{\boldsymbol{u}\in\mathcal{A}_l^p}\langle\nabla\Phi(\boldsymbol{y}_l), \boldsymbol{u}\rangle$ and $\boldsymbol{d}_l^{p,\mathrm{A}} = \boldsymbol{x}_l^p - \boldsymbol{v}_l^p$.
10: $\quad$**end for**
11: $\quad$**if** $\max_{p\in[r]}\left\langle -\nabla\Phi(\boldsymbol{y}_l), \boldsymbol{d}_l^{p,\mathrm{FW}} + \boldsymbol{d}_l^{p,\mathrm{A}}\right\rangle \le \epsilon$ **then**
12: $\quad\quad$**return** $\boldsymbol{x}_l$ as $\boldsymbol{x}_{k+1}$, $\{\mathcal{A}_l^p\}_{p\in[r]}$ as $\{\mathcal{A}_{k+1}^p\}_{p\in[r]}$, and $\{\{z_S^p\}_{S\in\mathcal{B}^p}\}_{p\in[r]}$.
13: $\quad$**end if**
14: $\quad g_l^{\mathrm{FW}} = \sum_{p\in[r]} m^p\left\langle -\nabla\Phi(\boldsymbol{y}_l), \boldsymbol{d}_l^{p,\mathrm{FW}}\right\rangle$.
15: $\quad g_l^{\mathrm{A}} = \sum_{p\in[r]} m^p\left\langle -\nabla\Phi(\boldsymbol{y}_l), \boldsymbol{d}_l^{p,\mathrm{A}}\right\rangle$.
16: $\quad$**if** $g_l^{\mathrm{FW}} \ge g_l^{\mathrm{A}}$ **then**
17: $\quad\quad\boldsymbol{d}_l = (\boldsymbol{d}_l^{1,\mathrm{FW}}, \dots, \boldsymbol{d}_l^{r,\mathrm{FW}})$ and $\gamma_{\max} = 1$.
18: $\quad$**else**
19: $\quad\quad\boldsymbol{d}_l = (\boldsymbol{d}_l^{1,\mathrm{A}}, \dots, \boldsymbol{d}_l^{r,\mathrm{A}})$ and $\gamma_{\max} = \min_{p\in[r]} z_{\boldsymbol{v}_l^p}^p/(1 - z_{\boldsymbol{v}_l^p}^p)$.
20: $\quad$**end if**
21: $\quad\gamma_l \in \mathrm{argmin}_{\gamma\in[0,\gamma_{\max}]} F(\boldsymbol{x}_l + \gamma\boldsymbol{d}_l)$.
22: $\quad$Update $\boldsymbol{x}_{l+1} = \boldsymbol{x}_l + \gamma_l\boldsymbol{d}_l$.
23: $\quad$Update $\{z_S^p\}_{S\in\mathcal{B}^p}$ accordingly for $p \in [r]$ (see (Lacoste-Julien and Jaggi 2015)).
24: $\quad$Update $\mathcal{A}_{l+1}^p = \{S \in \mathcal{B}^p \mid z_S^p > 0\}$ for $p \in [r]$.
25: **end for**

of AFW (Lacoste-Julien and Jaggi 2015). In Steps 15 and 16, we use $\nabla_p F(\boldsymbol{x}) = m^p \nabla\Phi(\boldsymbol{y})$ $(\forall p \in [r])$ to compute the Frank–Wolfe gap, g_l^{FW}, and away gap, g_l^{A}, defined on $\mathbb{R}^{rn}$. The resulting modified AFW exhibits the same convergence behavior as the standard AFW as follows:

Theorem 3. *After l iterations of Algorithm 3, the Frank–Wolfe gap (FW gap) satisfies $g_l^{FW} \leq O(\mathrm{poly}(n)/l)$. Let $\mathcal{B} \coloneqq \mathcal{B}^1 \times \cdots \times \mathcal{B}^r$. If $F(\cdot)$ is strongly convex and the reciprocal of the pyramidal width of $\mathrm{conv}(\mathcal{B})$ is $O(\mathrm{poly}(n))$, then we have $g_l^{FW} \leq O(\exp(-\Theta(l/\mathrm{poly}(n))))$ after l iterations. Moreover, the FW gap of each $p \in [r]$, defined by $g_l^{p,FW} \coloneqq \left\langle -\nabla\Phi(\boldsymbol{y}_l), \boldsymbol{d}_l^{p,FW} \right\rangle$, also converges at the same rates.*

Note that, once the convergence of the FW gap of each $p \in [r]$ is obtained, the away gap of each $p \in [r]$, defined by $g_l^{p,\mathrm{A}} \coloneqq \left\langle \nabla\Phi(\boldsymbol{y}_l), \boldsymbol{d}_l^{p,\mathrm{A}} \right\rangle$, also asymptotically converges to 0 as follows: Since $\boldsymbol{x}_l^p$ is can be written as $\sum_{S\in\mathcal{A}_l^p} z_S^p \mathbf{1}_S$ in each iteration, $\boldsymbol{x}_l^p + \frac{z_S^p}{1-z_S^p}(\boldsymbol{x}_l^p - \mathbf{1}_S) = \frac{1}{1-z_S^p}\boldsymbol{x}_l^p - \frac{z_S^p}{1-z_S^p}\mathbf{1}_S \in \mathrm{conv}(\mathcal{B}^p)$ holds for any $S \in \mathcal{B}^p$. Therefore, the definition of the FW gap implies $\left\langle -\nabla\Phi(\boldsymbol{y}_l), \frac{z_S^p}{1-z_S^p}(\boldsymbol{x}_l^p - \mathbf{1}_S) \right\rangle = \frac{z_S^p}{1-z_S^p}\left\langle -\nabla\Phi(\boldsymbol{y}_l), (\boldsymbol{x}_l^p - \mathbf{1}_S) \right\rangle \leq g_l^{p,\mathrm{FW}}$ for any $S \in \mathcal{B}^p$. From the definition of the away gap and $\mathcal{A}_l^p \subseteq \mathcal{B}^p$, if we let $\zeta_l^p \coloneqq \min_{S\in\mathcal{A}_l^p} z_S^p > 0$, we have $\zeta_l^p g_l^{p,\mathrm{A}} \leq g_l^{p,\mathrm{FW}}$. Hence the away gap converges to 0 if the FW gap does. To conclude, Algorithm 3 can update the input so as to satisfy the third condition in Algorithm 2. We can easily confirm that the first and second conditions are also satisfied due to the procedure of AFW (Algorithm 3).

Proof of Theorem 3. We first see that the convergence of g_l^{FW} implies that of the individual FW gaps, $g_l^{p,\mathrm{FW}}$ $(p \in [r])$. Since $\boldsymbol{s}_l^p \in \mathrm{argmin}_{\boldsymbol{u}\in\mathrm{conv}(\mathcal{B}^p)} \left\langle \nabla\Phi(\boldsymbol{y}_l), \boldsymbol{u} \right\rangle$ and $\boldsymbol{x}_l^p \in \mathrm{conv}(\mathcal{B}^p)$, we have $g_l^{p,\mathrm{FW}} = \left\langle -\nabla\Phi(\boldsymbol{y}_l), \boldsymbol{s}_l^p - \boldsymbol{x}_l^p \right\rangle \geq 0$ for all $p \in [r]$. Since $g_l^{\mathrm{FW}} = \sum_{p\in[r]} m^p g_l^{p,\mathrm{FW}}$, we have $g_l^{p,\mathrm{FW}} \leq g_l^{\mathrm{FW}}/m^p$. Therefore, individual gaps $g_l^{p,\mathrm{FW}}$ $(p \in [r])$ converge to 0 if g_l^{FW} does.

Next, we show the $O(\mathrm{poly}(n)/l)$ convergence by following the argument of (Lacoste-Julien and Jaggi 2015). Let $\boldsymbol{x}^*$ be the minimizer of $F(\cdot)$ on $\mathrm{conv}(\mathcal{B})$ and $h_l = F(\boldsymbol{x}_l) - F(\boldsymbol{x}^*)$ be the suboptimality gap. Since $F(\cdot)$ has L-Lipschitz-continuous gradient and convexity, for any $\gamma \in [0, \gamma_{\max}]$,

$$\begin{aligned} F(\boldsymbol{x}_{l+1}) &\leq F(\boldsymbol{x}_l + \gamma \boldsymbol{d}_l) \\ &\leq F(\boldsymbol{x}_l) + \gamma \left\langle \nabla F(\boldsymbol{x}_l), \boldsymbol{d}_l \right\rangle + \gamma^2 L \|\boldsymbol{d}_l\|^2/2 \\ &\leq F(\boldsymbol{x}_l) - \gamma g_l^{\mathrm{FW}} + \gamma^2 L D^2/2 \end{aligned}$$

holds, where D is the diameter of $\mathrm{conv}(\mathcal{B})$; note that $D \leq O(\mathrm{poly}(n))$ holds. The third inequality is due to $\|\boldsymbol{d}_l\| \leq D$ and $\left\langle \nabla F(\boldsymbol{x}_l), \boldsymbol{d}_l \right\rangle = \min\{-g_l^{\mathrm{FW}}, -g_l^{\mathrm{A}}\} \leq -g_l^{\mathrm{FW}}$ (see, Step 16). By subtracting $F(\boldsymbol{x}^*)$ from both sides, we obtain $h_{l+1} \leq h_l - \gamma g_l^{\mathrm{FW}} + \gamma^2 L D^2/2$. Below we discuss the convergence by using the number of steps where the line search (Step 21) does not result in $\gamma = \gamma_{\max}$; such steps are called good steps, and the number of good steps among the first l steps, denoted by $G(l)$, can be lower bounded as follows: In a step with $\gamma = \gamma_{\max}$ (called a drop step), we set at least one of $z_S^p > 0$ to zero, which means the sum of the sizes of active sets decreases by at least one. Thus, the number of drop steps among the first l steps is upper bounded by $\sum_{p\in[r]} |\mathcal{A}^p| + (l - \sum_{p\in[r]} |\mathcal{A}^p|)/2$, which implies $G(l) \geq (l - \sum_{p\in[r]} |\mathcal{A}^p|)/2 = \Theta(l)$. Following the proof of (Jaggi 2013, Theorems 1 and 2), we can show that $g_l^{\mathrm{FW}} = O(\mathrm{poly}(n)/G(l)) = O(\mathrm{poly}(n)/l)$ holds; hence the first clam is obtained.

Finally, we prove the linear convergence under the assumptions of the strong convexity and bounded pyramidal width. We define error vector $\boldsymbol{e}_l \coloneqq \boldsymbol{x}^* - \boldsymbol{x}_l$ and its normalized version $\hat{\boldsymbol{e}}_l \coloneqq \boldsymbol{e}_l/\|\boldsymbol{e}_l\|$. Thanks to the argument of (Lacoste-Julien and Jaggi 2015, Section 2.1), we have

$$h_l - h_{l+1} \geq \frac{\mu}{L\|\boldsymbol{d}_l\|^2} \frac{\left\langle -\nabla F(\boldsymbol{x}_l), \boldsymbol{d}_l \right\rangle^2}{\left\langle -\nabla F(\boldsymbol{x}_l), \hat{\boldsymbol{e}}_l \right\rangle^2} h_l$$

if the lth iteration is a good step. Since $\left\langle -\nabla F(\boldsymbol{x}_l), \boldsymbol{d}_l \right\rangle \geq \max\{g_l^{\mathrm{FW}}, g_l^{\mathrm{A}}\} \geq (g_l^{\mathrm{FW}} + g_l^{\mathrm{A}})/2 = \left\langle -\nabla F(\boldsymbol{x}_l), \boldsymbol{s}_l - \boldsymbol{v}_l \right\rangle/2$, where $\boldsymbol{s}_l \coloneqq (\boldsymbol{s}_l^1, \ldots, \boldsymbol{s}_l^r)$ and $\boldsymbol{v}_l \coloneqq (\boldsymbol{v}_l^1, \ldots, \boldsymbol{v}_l^r)$, we have

$$h_l - h_{l+1} \geq \frac{\mu}{4LD^2} \frac{\left\langle -\nabla F(\boldsymbol{x}_l), \boldsymbol{s}_l - \boldsymbol{v}_l \right\rangle^2}{\left\langle -\nabla F(\boldsymbol{x}_l), \hat{\boldsymbol{e}}_l \right\rangle^2} h_l.$$

Here, $\boldsymbol{v}_l \in \mathbb{R}^{rn}$ is selected from $\mathcal{A}_l^1 \times \cdots \times \mathcal{A}_l^r$ to maximize $\left\langle \nabla F(\boldsymbol{x}_l), \boldsymbol{v}_l \right\rangle$. Moreover, we can regard $\mathcal{A}_l^1 \times \cdots \times \mathcal{A}_l^r$ as the active set of $\boldsymbol{x}_l \in \mathbb{R}^{rn}$ since

$$\begin{aligned} \sum_{\boldsymbol{u}^1\in\mathcal{A}_l^1} \cdots \sum_{\boldsymbol{u}^r\in\mathcal{A}_l^r} z_{\boldsymbol{u}^1}^1 \cdots z_{\boldsymbol{u}^r}^r (\boldsymbol{u}^1, \ldots, \boldsymbol{u}^r) &= \boldsymbol{x}_l, \\ \sum_{\boldsymbol{u}^1\in\mathcal{A}_l^1} \cdots \sum_{\boldsymbol{u}^r\in\mathcal{A}_l^r} z_{\boldsymbol{u}^1}^1 \cdots z_{\boldsymbol{u}^r}^r &= 1. \end{aligned}$$

Therefore, by using (Lacoste-Julien and Jaggi 2015, Theorem 3), we can show that $\left\langle -\nabla F(\boldsymbol{x}_l), \boldsymbol{s}_l - \boldsymbol{v}_l \right\rangle / \left\langle -\nabla F(\boldsymbol{x}_l), \hat{\boldsymbol{e}}_l \right\rangle$ is lower bounded by pyramidal width $\delta > 0$ of $\mathcal{B}$. Consequently, we obtain the geometric decrease of the suboptimality gap: $h_{l+1} \leq (1 - \frac{\mu}{4L}(\frac{\delta}{D})^2)h_l$. Therefore, if $1/\delta \leq O(\mathrm{poly}(n))$, we have $h_l = O(\exp(-\Theta(G(l)/\mathrm{poly}(n)))) = O(\exp(-\Theta(l/\mathrm{poly}(n))))$. The linear convergence of the FW gap g_l^{FW} can be obtained from (Lacoste-Julien and Jaggi 2015, Theorem 2), which shows $g_l^{\mathrm{FW}} \leq O(D\sqrt{h_l})$. □

As mentioned in Section 3.3, δ is conjectured to be lower bounded by $\Omega(1/\sqrt{n})$ (Lacoste-Julien and Jaggi 2015). More precisely, the pyramidal width of the rn-dimensional unit cube $\{0,1\}^{rn}$ is $1/\sqrt{rn}$, and it is conjectured in (Lacoste-Julien and Jaggi 2015) that the pyramidal width does not increase when an another vertex is added. Following the conjecture, the pyramidal width of any $\mathcal{B}$ is at least $1/\sqrt{rn}$.

References

Abernethy, J. D., and Wang, J.-K. 2017. On Frank–Wolfe and equilibrium computation. In *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. 6584–6593.

Altman, E.; Boulogne, T.; El-Azouzi, R.; Jiménez, T.; and Wynter, L. 2006. A survey on networking games in telecommunications. *Comput. Oper. Res.* 33(2):286–311.

Bar-Gera, H. 2002. Origin-based algorithm for the traffic assignment problem. *Transp. Sci.* 36(4):398–417.

Beckmann, M.; McGuire, C. B.; and Winsten, C. B. 1956. *Studies in the Economics of Transportation*. Yale University Press.

Bergman, D.; Cire, A. A.; van Hoeve, W.-J.; and Hooker, J. 2016. *Decision Diagrams for Optimization*. Springer, first edition.

Blum, A.; Even-Dar, E.; and Ligett, K. 2006. Routing without regret: On convergence to Nash equilibria of regret-minimizing algorithms in routing games. In *Proceedings of the 25th Annual ACM Symposium on Principles of Distributed Computing*, 45–52. ACM.

Braun, G.; Pokutta, S.; and Zink, D. 2019. Lazifying conditional gradient algorithms. *J. Mach. Learn. Res.* 20(71):1–42.

Bryant, R. E. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.* 100(8):677–691.

Correa, J. R., and Stier-Moses, N. E. 2011. Wardrop equilibria. In *Wiley Encyclopedia of Operations Research and Management Science*. Wiley Online Library.

Fabrikant, A.; Papadimitriou, C.; and Talwar, K. 2004. The complexity of pure Nash equilibria. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, 604–612. ACM.

Fischer, S., and Vöcking, B. 2004. On the evolution of selfish routing. In *Proceedings of the 12th European Symposium on Algorithms*, 323–334. Springer-Verlag.

Fischer, S.; Räcke, H.; and Vöcking, B. 2010. Fast convergence to Wardrop equilibria by adaptive sampling methods. *SIAM J. Comput.* 39(8):3700–3735.

Frank, M., and Wolfe, P. 1956. An algorithm for quadratic programming. *Naval Res. Logis. Quart.* 3(1–2):95–110.

Han, Z.; Niyato, D.; Saad, W.; Baar, T.; and Hjrungnes, A. 2012. *Game Theory in Wireless and Communication Networks: Theory, Models, and Applications*. Cambridge University Press, 1st edition.

Imase, M., and Waxman, B. M. 1991. Dynamic Steiner tree problem. *SIAM J. Discrete. Math.* 4(3):369–384.

Inoue, Y., and Minato, S. 2016. Acceleration of ZDD construction for subgraph enumeration via path-width optimization. Technical report, TCS-TR-A-16-80, Hokkaido University.

Inoue, T.; Iwashita, H.; Kawahara, J.; and Minato, S. 2016. Graphillion: software library for very large sets of labeled graphs. *Int. J. Software Tool. Tech. Tran.* 18(1):57–66.

Jaggi, M. 2013. Revisiting Frank–Wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, 427–435. PMLR.

Jahn, O.; Möhring, R. H.; Schulz, A. S.; and Stier-Moses, N. E. 2005. System-optimal routing of traffic flows with user constraints in networks with congestion. *Oper. Res.* 53(4):600–616.

Kawahara, J.; Inoue, T.; Iwashita, H.; and Minato, S. 2017. Frontier-based search for enumerating all constrained subgraphs with compressed representation. *IEICE Trans. Fund. Electron. Comm. Comput. Sci.* E100.A(9):1773–1784.

Kerdreux, T.; Pedregosa, F.; and d'Aspremont, A. 2018. Frank–Wolfe with subsampling oracle. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, 2591–2600. PMLR.

Knight, S.; Nguyen, H. X.; Falkner, N.; Bowden, R.; and Roughan, M. 2011. The Internet topology zoo. *IEEE J. Sel. Areas Commum.* 29(9):1765–1775.

Knuth, D. E. 2011. *The Art of Computer Programming: Combinatorial Algorithms, Part 1*, volume 4A. Addison-Wesley Professional, 1st edition.

Krishnan, R. G.; Lacoste-Julien, S.; and Sontag, D. 2015. Barrier Frank–Wolfe for marginal inference. In *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc. 532–540.

Lacoste-Julien, S., and Jaggi, M. 2015. On the global linear convergence of Frank–Wolfe optimization variants. In *Advances in Neural Information Processing Systems 28*, 496–504. Curran Associates, Inc.

Lacoste-Julien, S.; Jaggi, M.; Schmidt, M.; and Pletscher, P. 2013. Block-coordinate Frank–Wolfe optimization for structural SVMs. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, 53–61. PMLR.

Minato, S. 1993. Zero-suppressed BDDs for set manipulation in combinatorial problems. In *Proceedings of the 30th International Design Automation Conference*, 272–277. IEEE.

Morrison, D. R.; Sewell, E. C.; and Jacobson, S. H. 2016. Solving the pricing problem in a branch-and-price algorithm for graph coloring using zero-suppressed binary decision diagrams. *INFORMS J. Comput.* 28(1):67–82.

Roughgarden, T., and Tardos, E. 2002. How bad is selfish routing? *J. ACM* 49(2):236–259.

Roughgarden, T. 2005. *Selfish Routing and the Price of Anarchy*. The MIT Press.

Sandholm, W. H. 2001. Potential games with continuous player sets. *J. Econ. Theory* 97(1):81–108.

Shakkottai, S.; Altman, E.; and Kumar, A. 2007. Multihoming of users to access points in WLANs: A population game perspective. *IEEE J. Sel. A. Commun.* 25(6):1207–1215.

Thai, J. 2017. *On learning Game-Theoretical models with Application to Urban Mobility*. Ph.D. Dissertation, UC Berkeley. ProQuest ID: Thai_berkeley_0028E_17598. Merritt ID: ark:/13030/m59s6nbq. Retrieved from https://escholarship.org/uc/item/3b61v84v.

Vliet, D. V. 1987. The Frank-Wolfe algorithm for equilibrium traffic assignment viewed as a variational inequality. *Transport. Res. Part B: Method.* 21(1):87–89.