The Complexity of Optimizing Atomic Congestion

Cornelius Brand[1], **Robert Ganian**[2], **Subrahmanyam Kalyanasundaram**[3], **Fionn Mc Inerney**[2]

[1]Algorithms & Complexity Theory Group, Regensburg University, Germany
[2]Algorithms and Complexity Group, TU Wien, Austria
[3]Department of Computer Science and Engineering, IIT Hyderabad, India
cornelius.brand@ur.de, rganian@gmail.com, subruk@cse.iith.ac.in, fmcinern@gmail.com

Abstract

Atomic congestion games are a classic topic in network design, routing, and algorithmic game theory, and are capable of modeling congestion and flow optimization tasks in various application areas. While both the price of anarchy for such games as well as the computational complexity of computing their Nash equilibria are by now well-understood, the computational complexity of computing a *system-optimal* set of strategies—that is, a centrally planned routing that minimizes the average cost of agents—is severely understudied in the literature. We close this gap by identifying the exact boundaries of tractability for the problem through the lens of the parameterized complexity paradigm. After showing that the problem remains highly intractable even on extremely simple networks, we obtain a set of results which demonstrate that the structural parameters which control the computational (in)tractability of the problem are not vertex-separator based in nature (such as, e.g., treewidth), but rather based on edge separators. We conclude by extending our analysis towards the (even more challenging) min-max variant of the problem.

1 Introduction

Congestion games are a by-now classic and widely studied model of network resource sharing. Introduced by Rosenthal (1973), congestion games and their innumerable variants and extensions have been the focus of a vast body of literature, spanning fields from algorithmic game theory (Cominetti et al. 2019) over routing (Kunniyur and Srikant 2003) and network design (Anshelevich et al. 2004), to diverse contexts within artificial intelligence (Ashlagi, Monderer, and Tennenholtz 2007; Meir et al. 2012; Marchesi, Castiglioni, and Gatti 2019; Harks et al. 2022), both applied and theoretical.

The basic setup of congestion games comprises a network—modeled as a directed graph—and a set of agents that each have an origin and a destination (the setting where there is an infinite number of agents and single agents are infinitesimally small is called *non-atomic* while here we focus on the classic, *atomic* case, where agents are individual entities). The agents need to decide which routes to take in order to reach their destination in a way that minimizes the cost of their route, where the cost can capture, e.g., the amount of time or resources required. The eponymous *congestion* enters the scene as follows: the cost accrued by a single agent when traversing a link in the network depends on the number of agents using that link, as described by the link's *latency function*. In essence, the latency function captures how the cost of using each link changes depending on the number of agents using it; depending on the context, more agents using a link could lead to each of them paying a greater cost (e.g., when dealing with traffic congestion) or a lower cost (e.g., when dealing with logistical supply chains), up to a maximum capacity for that link.

It is well-known that selfish strategies in congestion games may not lead to optimal outcomes for all agents, let alone a *system-optimal* outcome[1] (that is, one achieving the minimum average cost) (Sharon et al. 2018). In fact, the existence of Nash equilibria for these games was the seminal question investigated by Rosenthal (1973), and is still of interest in economics and game theory today. Notably, the *price of anarchy* for these games has by now been determined (Christodoulou and Koutsoupias 2005; Awerbuch, Azar, and Epstein 2005). The price of anarchy in this context is defined as the ratio

$$\sup_S \frac{\mathrm{cost}(S)}{\mathrm{cost}(S_{\mathrm{sys}})}, \tag{PoA}$$

where the supremum ranges over all Nash equilibria, $\mathrm{cost}(S)$ is the cost of a set of strategies S for the agents, and S_{sys} is a *system optimum*, minimizing the average cost over all agents. In addition, the computational complexity of computing Nash equilibria is equally well-studied (Ackermann, Röglin, and Vöcking 2006; Fabrikant, Papadimitriou, and Talwar 2004); see also the many recent works on the problem (Harks et al. 2022; Wang et al. 2022).

While the price of anarchy defined in (PoA) as well as the cost of Nash equilibria $\mathrm{cost}(S)$ (i.e., the numerator in (PoA)) have been extensively treated in the literature on congestion games, it may come as a surprise that, to the best of our knowledge, almost nothing is known about the computational complexity of computing the *denominator* $\mathrm{cost}(S_{\mathrm{sys}})$ of (PoA), that is, determining a *system-optimal set of strategies* for the players. Far from just an intellectual curiosity, applications, e.g., in road or internet traffic routing and

[1]In the literature, such outcomes are sometimes called the (*social* or *collective*) *optimum*.

planning, make this a pressing question, especially given the rapid developments in autonomous driving systems and the widely adopted political strategy of emphasizing public transportation for its lower environmental impact, which is usually centrally planned and routed, as opposed to individual transport (Sharon et al. 2017a,b; Chen et al. 2020; Sharon 2021; Jalota et al. 2023).

One possible reason for this gap may lie in the fact that even the most restricted instances of centrally routing a set of agents across a network in a socially optimal manner become hopelessly hard from the perspective of classical computational complexity theory. To illustrate the severity of this phenomenon, several of these basic classes of intractable instances are described in Section 3. Another explanation for this blind spot in the literature can possibly be found in the fact that, as far as the price of anarchy is concerned, the network structure itself does not appear to play a significant role, whereas this changes drastically when it comes to actually computing a system-optimal set of strategies.

A particular approach that has proven immensely useful for computationally intractable problems lies in employing the rich toolset offered by *parameterized complexity theory* (Downey and Fellows 2013; Cygan et al. 2015) in order to obtain a rigorous, more fine-grained and detailed description of the computational complexity of a problem. The central aim of such an endeavour is to identify the structural properties of the input—captured via numerical *parameters*—which give rise to fixed-parameter algorithms for the problem (see Section 2). Some examples where the parameterized complexity toolset has been successfully applied in the context of Artificial Intelligence research include the series of works on Hedonic Games (Boehmer and Elkind 2020a,b; Ganian et al. 2022a), Integer Programming (Ganian and Ordyniak 2018; Eiben et al. 2019; Dvorák et al. 2021; Chan et al. 2022), Data Completion (Ganian et al. 2018; Dahiya et al. 2021; Ganian et al. 2022b; Koana, Froese, and Niedermeier 2023), and Bayesian Network Learning (Ordyniak and Szeider 2013; Grüttemeier, Komusiewicz, and Morawietz 2021; Ganian and Korchemna 2021; Grüttemeier and Komusiewicz 2022).

Our Contributions. As mentioned above, the problem of computing system-optimal strategies in atomic congestion games (SOAC) is extremely hard in terms of classical complexity theory. The core approach of parameterized complexity analysis is to identify those structural properties that a problem instance should have that, even though exceedingly hard in the general case, give rise to its fixed-parameter tractability. Since the network is modeled as a graph, a natural first choice would be to parameterize by the well-established *treewidth* (of the underlying undirected graph) (Robertson and Seymour 1986). Unfortunately, as our first result, we show that the problem remains NP-hard not only on networks which have treewidth 2, but even on networks consisting of a star plus an additional vertex (Theorem 1). This result rules out not only the use of treewidth, but also of virtually all other reasonable graph measures, as a single parameter to solve the problem in full generality.

The above lower bound essentially means that, in order to achieve progress, one needs to combine structural parameters with some auxiliary parameterization of the problem instances. In the context of congestion games, it would seem tempting to consider the number of agents as such an auxiliary parameter, however that would severely restrict any obtained algorithms: while one could reasonably expect that networks of interest may be well-structured, the number of agents in relevant instances of congestion games is typically large, and hence, does not constitute a well-applicable parameter. Instead, here we consider the maximum capacity $\mathtt{c}_{\mathtt{max}}$ of a link in the network as an auxiliary parameter—a value which is never larger, but could be much lower, than the total number of agents in the whole network.

It is important to note that SOAC remains extremely challenging even when parameterized by $\mathtt{c}_{\mathtt{max}}$. In fact, even if $\mathtt{c}_{\mathtt{max}}$ is fixed to a small constant, the problem is NP-hard when restricted to networks of constant treewidth (Theorem 2); the same reduction also rules out the use of other network parameters based on decompositions along *small vertex separators* (such as *treedepth* (Nesetril and de Mendez 2012)). However, as we show in our main algorithmic result (Theorem 5), SOAC is fixed-parameter tractable when parameterized by $\mathtt{c}_{\mathtt{max}}$ plus a suitable measure that guarantees the network's decomposability along *small edge cuts*.

Basic examples of such measures include the treewidth plus the maximum degree of the network (Ordyniak and Szeider 2013; Gözüpek et al. 2017), or the *feedback edge number* (i.e., the edge deletion distance to acyclic networks) (Koana et al. 2021; Füchsle et al. 2022). In our contribution, we build on the recently introduced *spanning tree* decompositions (Ganian and Korchemna 2021, 2022) to provide a significantly more general result—in particular, we develop a highly non-trivial dynamic programming algorithm to establish fixed-parameter tractability with respect to the recently introduced slim variant of *treecut width*. We complement Theorem 5 with lower bounds (Theorems 3, 4, and 6) that show the result to be an essentially tight delimitation of the exact boundaries of tractability for SOAC.

In the final section of this article, we focus on a more general variant of SOAC, where instead of requiring all agents to be routed to their destinations, we ask to minimize the system optimum while routing as many agents as possible (or, equivalently, when at most α agents may be left unrouted). This problem, motivated in part by similar lines of investigation conducted for, e.g., multi-agent path finding (Huang et al. 2022) and vehicle routing (Pham et al. 2022; Abu-Monshar and Al-Bazi 2022), can be seen as a "min-max" variant of SOAC, and hence, we denote it MSOAC. Crucially, MSOAC is even more challenging than SOAC: it remains NP-hard on bidirected trees even for $\mathtt{c}_{\mathtt{max}} = 1$ (Erlebach and Jansen 2001), and, perhaps even more surprisingly, is left open on very simple network structures, such as bounded-capacity star networks, when parameterized by α. Be that as it may, as our final result, we show that on bounded-degree networks, the spanning-tree based algorithm obtained for SOAC can be lifted to also solve MSOAC via a fixed-parameter algorithm when parameterized by the treewidth of the network, along with α and $\mathtt{c}_{\mathtt{max}}$.

A mind-map of our results is provided in Figure 1.

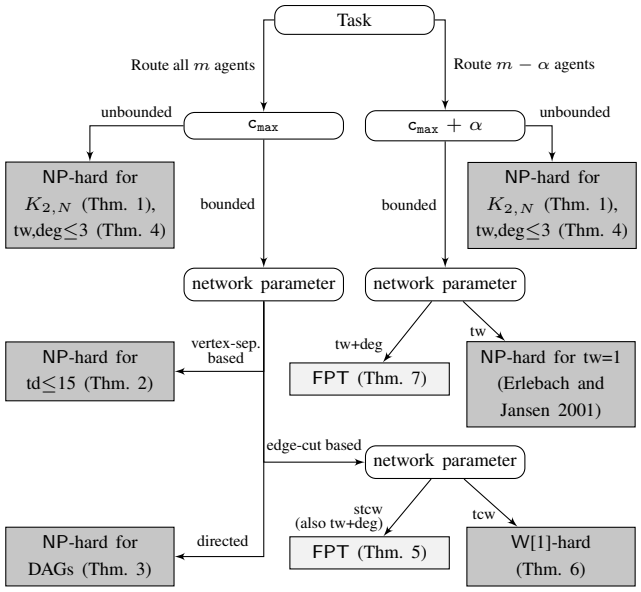

Figure 1: A mind map of our results on computing system-optimal strategies in congestion games. The formal problem definition as well as a discussion of the considered parameters is provided in Section 2; here, tw stands for treewidth, deg stands for maximum degree, td stands for *treedepth*, (s)tcw stands for (slim) treecut width, and DAGs stands for directed acyclic graphs.

2 Preliminaries

For a positive integer $i \in \mathbb{N}$, we let $[i] = \{1, 2, \ldots, i\}$. We refer to the book by Diestel (2012) for standard graph terminology. While the networks are modeled as directed graphs (i.e., digraphs), many of our results use the skeletons (i.e., the underlying undirected graphs) of these digraphs. The *skeleton* $\underline{G}$ of a directed graph G is the simple undirected graph obtained by replacing each arc in G by an undirected edge. The graph class $K_{i,N} = \{K_{i,j} \mid j \in \mathbb{N}\}$ is the class of all complete bipartite graphs where one side has size i.

Formal Problem Definition. Given a digraph $G = (V, E)$, let $\mathcal{P}$ be the set of all directed paths in G. Given a set $A = \{a_1, \ldots, a_m\}$ of agents where each agent a_i is associated with a tuple $(s_i, t_i) \in V^2$, a *flow assignment* F is a mapping from A to $\mathcal{P}$ such that each agent a_i is mapped to a directed path from s_i to t_i. For an arc $e \in E$, let $f_F(e) = |\{a \mid a \in A \wedge e \in F(a)\}|$ be the number of agents whose flow passes through e; when F is clear from the context, we omit it and simply use $f(e)$ instead.

Intuitively, our primary problem of interest asks to compute a flow assignment of all the agents that minimizes the total cost. However, in order to formalize the algorithmic lower bounds obtained for the problem, we follow the standard practice of formulating the problem as an equivalent decision problem (see below). To avoid any doubts, we remark that all our algorithmic results are constructive and can also immediately output the minimum cost of a flow assignment with the required properties.

System Optimum Atomic Congestion (SOAC)
Input: A digraph $G = (V, E)$, a positive integer λ, a set $A = \{a_1, \ldots, a_m\}$ of agents where each agent a_i is associated with a tuple $(s_i, t_i) \in V^2$, and for each arc $e \in E$, a latency function $\ell_e : [m] \to \mathbb{R}_{\geq 0} \cup \{\infty\}$.
Question: Does there exist a flow assignment F such that $\sum_{e \in E} f_F(e) \cdot \ell_e(f_F(e)) \leq \lambda$?

In specific settings studied in the literature, the latency function is sometimes required to satisfy certain additional conditions, such as being non-decreasing. As illustrative examples, observe that when agents represent individual vehicles in a traffic network, the latency function will typically be increasing (the cost of 100 agents using a single link is greater than 100 times the cost of that link when it is used by a single agent), but if agents represent individual parcels or shipments in a logistics network, one would expect it to be decreasing (the cost of 100 agents using a single link would be lower than 100 times the cost of that link when it is used by a single agent). In order to capture as wide a range of scenarios as possible—including, e.g., buffered or batch-wise processing at network nodes leading to decreasing or even oscillating latencies, respectively—our study targets the problem with arbitrary latency functions.

Given a latency function $\ell_e : [m] \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ for an arc e, let the capacity c_e of e be defined as the maximum admissible value of the latency function, i.e., $\max\{z \mid \ell_e(z) \neq \infty\}$. Let the *maximum capacity* of a network be defined as $\mathsf{c}_{\max} := \max_{e \in E} c(e)$. Crucially, while the maximum capacity can never exceed the number m of agents in the network, one can reasonably expect it to be much smaller than m in more complex networks.

Parameterized Complexity Theory. In parameterized algorithmics (Cygan et al. 2015; Downey and Fellows 2013; Niedermeier 2006), the running-time of an algorithm is studied with respect to a parameter $k \in \mathbb{N}$ and input size n. The basic idea is to find a parameter that describes the structure of the instance such that the combinatorial explosion can be confined to this parameter. In this respect, the most favorable complexity class is FPT (*fixed-parameter tractable*), which contains all problems that can be decided by an algorithm running in time $f(k) \cdot n^{\mathcal{O}(1)}$, where f is a computable function. Algorithms with this running-time are called *fixed-parameter algorithms*. A basic way of excluding fixed-parameter tractability for a parameterized problem is to show that it is W[1]-hard or that it remains NP-hard even when the parameter value is fixed.

In this work, we identify the exact boundaries of tractability for SOAC in the context of fundamental graph parameters, as depicted in Figure 2. For two of these parameters—notably treewidth (Robertson and Seymour 1986) and treedepth (Nesetril and de Mendez 2012)—we do not need to provide explicit definitions since the respective lower bounds we obtain for SOAC construct instances whose skeletons are well-known to have bounded treewidth and treedepth. The *feedback edge number* (Koana et al. 2021; Füchsle et al. 2022) is simply the minimum number of edges that need to be removed from an undirected graph in order

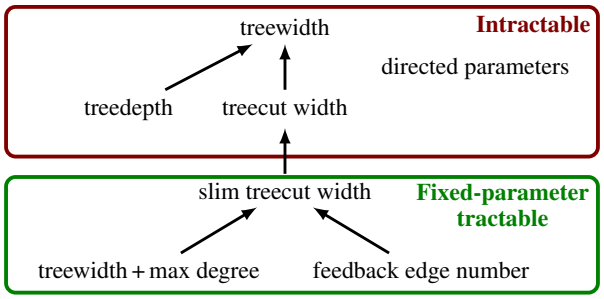

Figure 2: A pictorial view of the tractability of SOAC when capacities are bounded. An arc from a parameter x to a parameter y indicates that x is dominated by y, i.e., a bound on x implies a bound on y but the opposite does not hold.

to obtain a forest. The *treecut width* is an edge-cut based analog of treewidth (Wollan 2015; Wollan and Marx 2014). The term "directed graph parameters" here broadly refers to all parameters that achieve constant values on directed acyclic graphs; this includes directed treewidth (Johnson et al. 2001), Kelly-width (Hunter and Kreutzer 2008), and DAG-width (Berwanger et al. 2012), to name a few.

The proofs of our algorithmic results require the definition of the spanning-tree decompositions recently introduced for edge-cut based parameterizations (Ganian and Korchemna 2021, 2022). For a graph G and a tree T over $V(G)$, let the *local feedback edge number* at $v \in V(G)$ be $E^{G,T}_{\text{loc}}(v) = \{uw \in E(G) \setminus E(T) \mid \text{the path between } u \text{ and } w \text{ in } T \text{ contains } v\}$. The *edge-cut width* of the pair (G,T) is $\mathbf{ecw}(G,T) = 1 + \max_{v \in V} |E^{G,T}_{\text{loc}}(v)|$. This allows us to characterize both the *slim treecut width* parameter used in Theorem 5 and the combined parameter of treewidth plus maximum degree used in Theorem 7 in a way which is well-suited for dynamic programming.

Lemma 1 (Prop. 27 and Thm. 30, Ganian and Korchemna 2022)**.** *Every graph G with slim treecut width k admits a spanning tree T over some supergraph G' of G such that (G',T) has edge-cut width at most $3(k+1)^2$. Moreover, such a pair (G',T) can be computed in time $2^{k^{\mathcal{O}(1)}} \cdot |V(G)|^4$.*

Lemma 2 (Prop. 22, Prop. 26, and Thm. 30, Ganian and Korchemna 2022)**.** *Every graph G with maximum degree d and treewidth w admits a spanning tree T over some supergraph G' of G such that both the edge-cut width of (G',T) and the maximum degree of T are upper-bounded by $\mathcal{O}(d^2w^2)$. Moreover, such a pair (G',T) can be computed in time $2^{(dw)^{\mathcal{O}(1)}} \cdot |V(G)|^4$.*

3 The Surprising Difficulty of Solving Atomic Congestion Games

Before initializing our more fine-grained parameterized analysis of SOAC, we first consider the computational complexity of the problem from the classical point of view, that is, with respect to NP-hardness. In fact, we show that even surprisingly simple input instances turn out to be intractable.

To begin, we show that SOAC is NP-hard even when the underlying undirected graph of the input digraph is restricted to the class $K_{2,N}$. We prove this result (as well as Theorem 4 later on) via a reduction from the following variant of the classical SUBSET SUM problem over d-dimensional vectors:

Multidimensional Uniform 0/1 Knapsack (MUKS)
Input: A set $S = \{\vec{v}_1, \ldots, \vec{v}_n\} \subseteq \mathbb{N}^d$ of d-dimensional vectors containing natural numbers, a positive integer k, and a target vector $\vec{T} \in \mathbb{N}^d$.
Question: Is there a subset $S' \subseteq S$ of at least k vectors such that $\sum_{\vec{s} \in S'} \vec{s} \leq \vec{T}$ holds?

Lemma 3. MUKS *is* NP*-hard and also* W[1]*-hard parameterized by d, even if all numbers are encoded in unary.*

Before proceeding to the reduction, let us briefly remark on the significance of the result in the context of this article. Essentially, establishing NP-hardness on $K_{2,N}$ rules out not only fixed-parameter tractability under almost all commonly considered graph parameterizations (including not only the classical *treewidth* (Robertson and Seymour 1986), but also the *treedepth* (Nesetril and de Mendez 2012), the *vertex cover number* (Korhonen and Parvianen 2015), and the more recently introduced edge-cut variants of treewidth (Marx and Wollan 2014; Ganian and Korchemna 2021, 2022)), but even polynomial-time algorithms for instances where such parameters are bounded by a constant.

Theorem 1. SOAC *is* NP*-hard even when restricted to networks whose skeletons belong to the class $K_{2,N}$.*

Proof Sketch. We provide a polynomial-time reduction from MUKS where numbers are encoded in unary to SOAC. Recall that MUKS is NP-hard by Lemma 3. The reduction is as follows. For every input $\vec{v}_i \in S$ in the MUKS instance, there is a source vertex s_i. For each of the d entries $T_1, \ldots, T_d$ in the target vector $\vec{T}$, there is a target vertex t_j. There are also two vertices h_0 and h_1. Each of the n source vertices s_i is connected by outgoing arcs to both h_0 and h_1. The latencies are defined as $\ell_{s_i,h_0} = 0$ for all arcs (s_i,h_0), and as $\ell_{s_i,h_1}(x) = 1/x$ for all arcs (s_i,h_1). Both h_0 and h_1 have outgoing arcs to all the target vertices t_j. For arcs (h_0,t_j), $\ell_{h_0,t_j}(x) = 0$ if $x \leq T_j$, and otherwise, $\ell_{h_0,t_j}(x) = \infty$. That is, the capacity of the arc (h_0,t_j) is equal to T_j, for each $j \in [d]$. For arcs (h_1,t_j), $\ell_{h_1,t_j}(x) = 0$.

For the i-th input vector $\vec{v}_i$, let $v_{i,j}$ be its j-th entry. For all $i \in [n]$ and $j \in [d]$, we want to route $v_{i,j}$ agents from s_i to t_j. That is, there are $v_{i,j}$ copies of the pair (s_i,t_j) for all $i \in [n]$ and $j \in [d]$. To complete the proof, it suffices to show that the constructed instance admits a solution of cost at most $n-k$ if and only if the original instance of MUKS has a solution comprised of at least k vectors. □

Since Theorem 1 essentially rules out fixed-parameter algorithms based on structural network parameters alone, we turn our attention to parameterizing by the maximum capacity $\mathtt{c_{max}}$. As our first result in this direction, we show that SOAC remains NP-hard on networks of constant treewidth, even if the maximum capacity of a link is just 1. This is done via a reduction from the following classical graph problem.

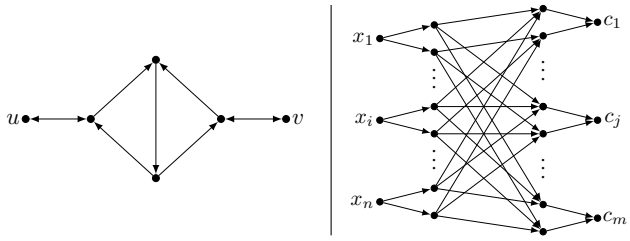

Figure 3: The arc gadget replacing each undirected edge uv in the reductions from the proofs of Thms. 2 and 6 (left), and the digraph constructed in the proof of Thm. 3 (right).

Edge-Disjoint Paths (EDP)
Input: A graph G and a set P of terminal pairs, that is, a set of subsets of $V(G)$ of size two.
Question: Is there a set of pairwise edge-disjoint paths connecting each set of terminal pairs in P?

Theorem 2. SOAC *is* NP*-hard even when restricted to networks with* $\mathtt{c_{max}} = 1$ *and whose skeletons have treewidth or treedepth at most* 15.

Proof Sketch. It is known that EDP is NP-hard when restricted to the class $K_{3,N}$ (Fleszar, Mnich, and Spoerhase 2018). The reduction takes an instance $\mathcal{I}$ of EDP on $K_{3,N}$ and replaces each edge uv in $\mathcal{I}$ with the gadget depicted in Figure 3 (left). It uses agents to represent the paths in $\mathcal{I}$, and sets the latency functions in a way that each arc can only support a single path. To complete the proof, it suffices to show that the reduction is correct and that the constructed instances have the claimed properties. □

Theorem 2 rules out using classic vertex-separator based parameters like treewidth, pathwidth or treedepth to solve SOAC, even when $\mathtt{c_{max}} = 1$. All of these parameters are tied to the skeleton of the network and do not account for the orientations of the arcs. One could wonder whether structural parameters introduced specifically for directed graphs would be better suited for the task. The unifying feature of the most widely studied of these parameters (Johnson et al. 2001; Hunter and Kreutzer 2008; Berwanger et al. 2012) is that they achieve constant values—typically 1—on directed acyclic graphs (DAGs). Below, we show that bounded-capacity SOAC is NP-hard even on simple DAGs, ruling out the use of these directed network parameters.

Theorem 3. SOAC *is* NP*-hard even when restricted to networks which are DAGs, have* $\mathtt{c_{max}} = 3$*, and whose skeletons have maximum degree* 4.

Proof Sketch. We reduce from a variant of SAT known as monotone cubic exact 1-in-3 SAT, which is known to be NP-hard (Porschen et al. 2014; Schmidt 2010). Here, clauses have size three and must be satisfied by a single literal, each variable appears in three clauses, and all literals are positive.

Suppose we are given such an n-variate m-clause positive cubic 3-CNF formula φ. The instance of SOAC is constructed as follows. It contains the sets of vertices $X = \{x_1, \ldots, x_n\}$ and $C = \{c_1, \ldots, c_m\}$ that are identified

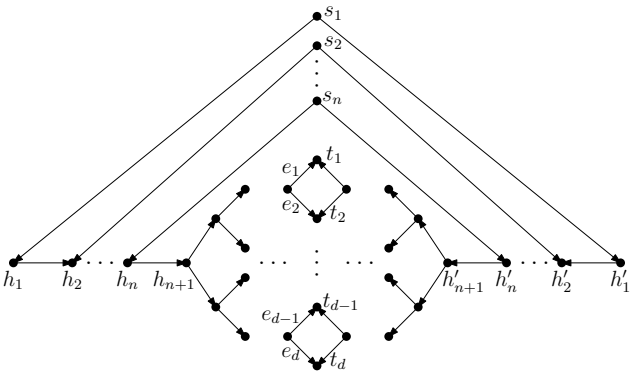

Figure 4: The digraph constructed in the proof of Thm. 4.

with the variables and clauses appearing in φ, respectively. We also add two more copies of X, namely $X^T = \{x_1^T, \ldots, x_n^T\}$ and $X^F = \{x_1^F, \ldots, x_n^F\}$, and, similarly, two more copies C^T and C^F of C. Now, x_i has an arc to x_i^T and one to x_i^F, both with latency $\ell(1) = \ell(2) = 1$ and $\ell(3) = 0$, and capacity 3. Whenever x_i appears in the j-th clause of φ, then x_i^T has an arc to c_j^T and x_i^F has an arc to c_j^F, both of latency 0. Further, c_i^T has an arc to c_i of capacity 1 and c_i^F has an arc to c_i of capacity 2, for all $i \in [m]$, and latency 0 otherwise. If x_i appears in the j-th clause of φ, then there is an agent with origin x_i and destination c_j. An illustration of the construction is provided in Figure 3 (right). To complete the proof, it suffices to show that the constructed instance has a solution of cost 0 if and only if φ is 1-in-3-satisfiable. □

In fact, we can also show that our problem of interest remains intractable even on extremely simple networks which are DAGs (complementing Theorem 1).

Theorem 4. SOAC *is* NP*-hard even when restricted to DAGs whose skeletons are planar graphs with treewidth and maximum degree both upper-bounded by* 3.

Proof Sketch. The proof is based on a reduction from MUKS. An illustration of the reduction is provided in Figure 4. Intuitively, for each vector $\vec{v}_i$, we add a vertex s_i, and if the j-th coordinate of $\vec{v}_i$ is γ, then we route γ many agents from s_i to the vertex t_j representing the j-th coordinate of the target vector. The construction allows all the agents starting in s_i to reach their destinations via either h_{n+1} or h'_{n+1}: the former route imposes a constraint which ensures that the sum of the agents passing through it does not exceed the restrictions imposed on T, while the latter route imposes a constraint which ensures that the agents originating from at least k vectors must be routed through the former route. □

4 A Fixed-Parameter Algorithm for SOAC

Recapitulating the results so far, in view of Theorem 1, we are focusing our efforts on identifying structural properties of networks which would allow us to solve SOAC on bounded-capacity networks. Theorem 2 rules out tractability via standard vertex-separator based graph parameters, while

Theorem 3 excludes the use of commonly studied directed variants of network parameters. A notable class of structural measures that has not been addressed yet by the obtained lower bounds are parameters that are tied to the existence of *edge cuts* as opposed to *vertex separators*.

Two "baseline" graph measures which allow for a structural decomposition along bounded-sized edge cuts are the *feedback edge number* (`fen`) (Koana et al. 2021; Füchsle et al. 2022), and the combined parameter of *treewidth plus maximum degree* (`twd`) (Ordyniak and Szeider 2013; Gözüpek et al. 2017). Unfortunately, both of these measures place very strong restrictions on the network: the former is only small on networks whose skeletons are trees plus a small number of arcs, while the latter cannot be used on networks with high-degree nodes.

As our main algorithmic contribution, we establish fixed-parameter tractability of bounded-capacity SOAC with respect to a recently introduced edge-cut based parameter called *slim treecut width* (`stcw`) (Ganian and Korchemna 2022). Crucially, this result immediately implies and generalizes fixed-parameter tractability with respect to either `fen` or `twd`, while also circumventing both of the aforementioned shortcomings: networks with skeletons of bounded slim treecut width can have high degree as well as be significantly more complex than just a tree.

Theorem 5. SOAC *is fixed-parameter tractable when parameterized by the slim treecut width of the skeleton $\underline{G}$ of the input digraph G plus the maximum capacity* $\mathtt{c_{max}}$.

Proof Sketch. We begin by invoking Lemma 1 to compute a pair (H, T) such that H is a supergraph of $\underline{G}$ and (H, T) has edge-cut width $k \leq 3 \cdot (\kappa + 1)^2$, where κ is the slim treecut width of $\underline{G}$. Our algorithm is based on a leaf-to-root dynamic programming procedure that traverses T while storing certain (carefully defined and bounded-sized) records about the part of T that has been processed so far. To this end, it will be useful to assume that T is rooted, and thus, we mark an arbitrary leaf of T as the root and denote it r.

Before we define the records used in the dynamic program, we will need some terminology. For a vertex $v \in V(H)$ with a child $w \in V(H)$, we say that w is a *simple* child of v if v and w belong to different connected components of $H - vw$; otherwise, we say that w is a *complex* child of v. Observe that, while the number of simple children of v is not bounded by k (none of the subtrees rooted at simple children contain any edges in $E^{H,T}_{\text{loc}}(v)$), the number of complex children of v is upper-bounded by $2k$ (each subtree rooted at a complex child contains an endpoint of at least one edge in $E^{H,T}_{\text{loc}}(v)$). Furthermore, we use $\mathcal{G}_v$ to denote the subdigraph of G induced on the vertices that are descendants of v (including v itself), and ∂_v to denote those arcs of G which have precisely one endpoint in $\mathcal{G}_v$; recall that $|\partial_v| \leq 2(k+1)$. An agent a_i is *outgoing* for v if $s_i \in \mathcal{G}_v$ and, at the same time, $t_i \notin \mathcal{G}_v$. Similarly, an agent a_i is *incoming* for v if $s_i \notin \mathcal{G}_v$ and, at the same time, $t_i \in \mathcal{G}_v$.

We are now ready to formalize the dynamic programming records used in our algorithm. A *snapshot* at a vertex v is a tuple of the form $(\mathcal{S}_{out}, \mathcal{S}_{in}, \mathcal{D}, \mathcal{R})$, where:

- $\mathcal{S}_{out}$ is a mapping from the set of all outgoing agents for v to arcs in ∂_v which are outgoing from $\mathcal{G}_v$;
- $\mathcal{S}_{in}$ is a mapping from the set of all incoming agents for v to arcs in ∂_v which are incoming to $\mathcal{G}_v$;
- $\mathcal{D}$ is a multiset of pairs (e, f) such that e is an incoming arc into $\mathcal{G}_v$, f is an outgoing arc from $\mathcal{G}_v$, and ef is not a 2-cycle;
- $\mathcal{R}$ is a multiset of pairs (e, f) such that e is an outgoing arc from $\mathcal{G}_v$, f is an incoming arc into $\mathcal{G}_v$, and ef is not a 2-cycle;
- each arc in $\partial(v)$ may only appear in at most $\mathtt{c_{max}}$ tuples over all of the entries in $\mathcal{S}_{out}, \mathcal{S}_{in}, \mathcal{D}, \mathcal{R}$.

As the number of outgoing and incoming agents for v cannot exceed $\mathtt{c_{max}} \cdot (2k+2)$ in a positive instance, the number of snapshots at v can be upper-bounded by $(\mathtt{c_{max}} + k)^{\mathcal{O}(\mathtt{c_{max}} k^2)}$; let $\text{Snap}(v)$ denote the set of all possible snapshots at v.

We can now formalize the syntax of the record at v, denoted $\text{Record}(v)$, as a mapping from $\text{Snap}(v)$ to $\mathbb{R}_{\geq 0} \cup \{\infty\}$. As for the semantics, $\text{Record}(v)$ will capture the minimum cost required to (1) route the outgoing and incoming agents to the designated arcs in $\mathcal{S}_{out}$ and $\mathcal{S}_{in}$, while assuming that (2) some (unidentified and arbitrary) agents will use arcs of ∂_v to enter and then exit $\mathcal{G}_v$ via the arcs designated in $\mathcal{D}$, and that (3) some (unidentified and arbitrary) agents will use arcs of ∂_v to exit and then return to $\mathcal{G}_v$ via the arcs designated in $\mathcal{R}$.

If no flow with these properties exists, we simply set $\text{Record}(v)(\Upsilon) := \infty$. This completes the formal definition of the records $\text{Record}(v)$. Observe that, since $\mathcal{G}_r = G$ and $\partial_r = \emptyset$, the only snapshot at r is $\Upsilon_r = (\{\emptyset\}, \{\emptyset\}, \emptyset, \emptyset)$ and $\mathcal{I}_{\Upsilon_r}$ is precisely the input instance to our problem. Hence, if we successfully compute the record $\text{Record}(r)$ for the root vertex r, then $\text{Record}(r)(\Upsilon_r)$ must be equal to the minimum cost of a flow assignment F in the input instance. Thus, to conclude the proof it remains to show how to compute the records at each vertex in a leaf-to-root fashion.

The computation of the records for the leaves can be done via brute-force enumeration. The core of the dynamic program lies in the computation of $\text{Record}(v)$ for a non-leaf vertex v. We do so by considering each snapshot $\Upsilon = (\mathcal{S}_{out}, \mathcal{S}_{in}, \mathcal{D}, \mathcal{R})$ at v independently, and computing the minimum cost of a flow assignment F in the subinstance $\mathcal{I}_\Upsilon$ as follows. For each simple child w of v, we observe that there is only a single snapshot $\Psi_w = (\mathcal{S}^w_{out}, \mathcal{S}^w_{in}, \{\emptyset\}, \{\emptyset\})$ of w where $\mathcal{S}^w_{out}$ maps all outgoing agents for w and $\mathcal{S}^w_{in}$ maps all incoming agents for w to the respective unique arcs in ∂_w. This corresponds to the fact that since a flow assignment is inherently loopless, there is only a unique way it may pass through the arcs in ∂_w. Let us define the *base cost* of v, denoted $b(v)$, as $\sum_{w \text{ is a simple child of v}} \text{Record}(w)(\Psi_w)$; intuitively, $b(v)$ captures the minimum cost required by a flow assignment in all simple children of v.

Next, we construct the instance $\mathcal{I}_\Upsilon$ for which we need to compute the minimum cost of a flow assignment. Essentially, our aim is to compute this cost via brute-forcing over all possible flow assignments, but at first glance this seems infeasible since the size of $\mathcal{I}_\Upsilon$ is not bounded by our parameters. The core insight we use to overcome this is that even

though $\mathcal{I}_\Upsilon$ could be large, all but only a parameter-bounded number of interactions have already been taken into account in the records of the children of v. Formally, we make use of this by constructing a "kernelized" instance $\mathcal{I}_\Upsilon^+$ where each flow assignment in $\mathcal{I}_\Upsilon$ can be mapped to a flow in $\mathcal{I}_\Upsilon^+$, but the number of flow assignments in the latter is upper-bounded by our parameters, and hence, can be efficiently enumerated. The overall running time of the algorithm can be upper-bounded by $\mathtt{c_{max}}^{\kappa^{\mathcal{O}(\kappa^4)}} \cdot n + 2^{\kappa^{\mathcal{O}(1)}} \cdot n^4$. □

We observe that the parameterization by $\mathtt{c_{max}}$ cannot be dropped from Theorem 5 in view of the lower bound in Theorem 4; indeed, every network of bounded treewidth and maximum degree also has bounded slim treecut width (Lemma 2). We conclude by turning our attention to whether Theorem 5 could be generalized to use the better-known *treecut width* parameter instead of the slim variant used in that algorithm. Treecut width is a structural graph parameter that also guarantees the decomposability of instances via bounded-sized edge cuts, and has previously been used to establish tractability for several NP-hard problems (Ganian, Kim, and Szeider 2022). Crucially, it forms an intermediary between the slim treecut width (which suffices for fixed-parameter tractability) and treewidth (for which SOAC remains intractable, even when restricted to bounded-capacity instances). We show that Theorem 5 is tight in the sense that fixed-parameter tractability cannot be lifted to treecut width.

Theorem 6. SOAC *is* W[1]*-hard when parameterized by the treecut width of the skeleton of the input network, even when restricted to networks with* $\mathtt{c_{max}} = 1$.

5 The Min-Max Atomic Congestion Problem

In this section, we turn our attention to the more general setting where instead of requiring *all* agents to be routed to their destinations, we allow for some agents to remain unrouted. In essence, this asks for a flow assignment that counterbalances the number of agents that reach their destinations with the total cost. As is usual in complexity-theoretic analysis, we state this task as a decision problem where we consider specific bounds on both the cost and the number of agents which need not be routed. For the purposes of this section, we formally extend the notion of *flow assignment* (defined in Section 2) to a mapping from a subset of agents to paths.

Min-Max System Opt. Atomic Congestion (MSOAC)
Input: A digraph $G = (V, E)$, positive integers λ and α, a set $A = \{a_1, \ldots, a_m\}$ of agents where each agent a_i is associated with a tuple $(s_i, t_i) \in V^2$, and for each arc $e \in E$, a latency function $\ell_e : [m] \to \mathbb{R}_{\geq 0} \cup \{\infty\}$.
Question: Is there a flow assignment F routing at least $m - \alpha$ agents such that $\sum_{e \in E} f_F(e) \cdot \ell_e(f_F(e)) \leq \lambda$?

We begin by noting that, in spite of the seemingly minor difference between the two problems, MSOAC is much more challenging than SOAC from a structural point of view. Indeed, on one hand, if we set $\alpha = 0$, the min-max variant becomes equivalent to SOAC. On the other hand, without this restriction, it can be observed that MSOAC remains NP-hard even on networks with a maximum capacity of 1 whose skeletons are trees. This follows by an immediate reduction from the MAXIMUM ARC DISJOINT PATHS problem, which has been shown to be NP-hard in precisely this setting (Erlebach and Jansen 2001); the reduction simply replaces each path with an agent and sets the latency function for each arc in the same way as in the proof of Theorem 2.

As the final contribution of this article, we show that MSOAC is fixed-parameter tractable when parameterized simultaneously by $\mathtt{c_{max}}$, α, and the combined parameter of treewidth and maximum degree.

Theorem 7. MSOAC *is fixed-parameter tractable when parameterized by the treewidth and maximum degree of the skeleton* $\underline{G}$ *of the input digraph* G *plus the maximum capacity* $\mathtt{c_{max}}$ *and* α.

Proof Sketch. On a high level, the algorithm follows the same overall approach as the one employed in the proof of Theorem 5; however, the dynamic programming steps and records required here are different (and significantly more involved), and the spanning tree T has bounded degree as per Lemma 2. The crucial difference is that in our snapshots, we need to additionally take into account (1) which of the outgoing and incoming agents are not being routed by the captured flow, and (2) how many agents have already been left unrouted in the part of the network processed so far. In our dynamic programming step at each vertex v, this additional tracked information then requires us to perform supplementary branching subroutines to ensure that the flows computed on the kernelized subinstance $\mathcal{I}_\Upsilon^+$ can be mapped to specific snapshots of the children of v. □

6 Concluding Remarks

Our results provide an essentially comprehensive complexity landscape for the problem of computing system-optimal flow assignments in atomic congestion games, closing a gap in the literature that contrasts with the significant attention other aspects of congestion games have received to date. We remark that our tractability results only require the input network to have the necessary structural properties and do not impose any restrictions on the possible origins and destinations of the agents. Moreover, all of the obtained algorithms can also be used to compute Nash-equilibria in atomic congestion games as long as an upper-bound on the cost of the flow is provided in the input. Future work could also consider the recently proposed setting of having some agents follow a greedily computed route (Sharon et al. 2018).

Another interesting avenue for future work would be to resolve the complexity of the min-max variant of the problem (i.e., MSOAC) on well-structured networks of unbounded degree. This problem is left open even on stars when parameterized by $\mathtt{c_{max}} + \alpha$, and we believe novel ideas will be required to breach this barrier; in particular, the techniques for solving the maximization variant of the related ARC DISJOINT PATHS problem on stars (Erlebach and Jansen 2001) do not generalize to MSOAC. As a longer-term goal, one would be interested in settling whether Theorem 7 could be lifted towards an analog of Theorem 5 that relies on the same structural measures of the network.

Acknowledgements

The first, second, and fourth authors were supported by the Austrian Science Foundation (FWF, project Y1329). The third author was supported by SERB-DST via grants MTR/2020/000497 and CRG/2022/009400.

References

Abu-Monshar, A.; and Al-Bazi, A. 2022. A multi-objective centralised agent-based optimisation approach for vehicle routing problem with unique vehicles. *Appl. Soft Comput.*, 125: 109187.

Ackermann, H.; Röglin, H.; and Vöcking, B. 2006. On the Impact of Combinatorial Structure on Congestion Games. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*, 613–622. IEEE Computer Society.

Anshelevich, E.; Dasgupta, A.; Kleinberg, J.; Tardos, E.; Wexler, T.; and Roughgarden, T. 2004. The price of stability for network design with fair cost allocation. In *45th Annual IEEE Symposium on Foundations of Computer Science*, 295–304.

Ashlagi, I.; Monderer, D.; and Tennenholtz, M. 2007. Learning Equilibrium in Resource Selection Games. In *Proc. of the 22nd AAAI Conference on Artificial Intelligence*, 18–23.

Awerbuch, B.; Azar, Y.; and Epstein, A. 2005. The Price of Routing Unsplittable Flow. In *Proc. of the 37th Annual ACM Symposium on Theory of Computing*, 57–66.

Berwanger, D.; Dawar, A.; Hunter, P.; Kreutzer, S.; and Obdrzálek, J. 2012. The dag-width of directed graphs. *J. Comb. Theory, Ser. B*, 102(4): 900–923.

Boehmer, N.; and Elkind, E. 2020a. Individual-Based Stability in Hedonic Diversity Games. In *Proc. of the 34th AAAI Conference on Artificial Intelligence*, 1822–1829.

Boehmer, N.; and Elkind, E. 2020b. Stable Roommate Problem with Diversity Preferences. In *Proc. of the 29th International Joint Conference on Artificial Intelligence*, 96–102.

Chan, T. F. N.; Cooper, J. W.; Koutecký, M.; Král, D.; and Pekárková, K. 2022. Matrices of Optimal Tree-Depth and a Row-Invariant Parameterized Algorithm for Integer Programming. *SIAM J. Comput.*, 51(3): 664–700.

Chen, Z.; Lin, X.; Yin, Y.; and Li, M. 2020. Path controlling of automated vehicles for system optimum on transportation networks with heterogeneous traffic stream. *Transportation Research Part C: Emerging Technologies*, 110: 312–329.

Christodoulou, G.; and Koutsoupias, E. 2005. The price of anarchy of finite congestion games. In *Proc. of the 37th Annual ACM Symposium on Theory of Computing*, 67–73.

Cominetti, R.; Scarsini, M.; Schröder, M.; and Moses, N. E. S. 2019. Price of Anarchy in Stochastic Atomic Congestion Games with Affine Costs. In *Proc. of the 2019 ACM Conference on Economics and Computation*, 579–580.

Cygan, M.; Fomin, F. V.; Kowalik, L.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer.

Dahiya, Y.; Fomin, F. V.; Panolan, F.; and Simonov, K. 2021. Fixed-Parameter and Approximation Algorithms for PCA with Outliers. In *Proc. of the 38th International Conference on Machine Learning, ICML 2021*, volume 139 of *Proc. of Machine Learning Research*, 2341–2351.

Diestel, R. 2012. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer.

Downey, R. G.; and Fellows, M. R. 2013. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer.

Dvorák, P.; Eiben, E.; Ganian, R.; Knop, D.; and Ordyniak, S. 2021. The complexity landscape of decompositional parameters for ILP: Programs with few global variables and constraints. *Artif. Intell.*, 300: 103561.

Eiben, E.; Ganian, R.; Knop, D.; and Ordyniak, S. 2019. Solving Integer Quadratic Programming via Explicit and Structural Restrictions. In *Proc. of the 33rd AAAI Conference on Artificial Intelligence*, 1477–1484.

Erlebach, T.; and Jansen, K. 2001. The Maximum Edge-Disjoint Paths Problem in Bidirected Trees. *SIAM J. Discrete Math.*, 14(3): 326–355.

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

Fleszar, K.; Mnich, M.; and Spoerhase, J. 2018. New algorithms for maximum disjoint paths based on tree-likeness. *Math. Program.*, 171(1-2): 433–461.

Füchsle, E.; Molter, H.; Niedermeier, R.; and Renken, M. 2022. Delay-Robust Routes in Temporal Graphs. In *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022*, volume 219 of *LIPIcs*, 30:1–30:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Ganian, R.; Hamm, T.; Knop, D.; Schierreich, S.; and Suchý, O. 2022a. Hedonic Diversity Games: A Complexity Picture with More than Two Colors. In *Proc. of the 36th AAAI Conference on Artificial Intelligence*, 5034–5042.

Ganian, R.; Hamm, T.; Korchemna, V.; Okrasa, K.; and Simonov, K. 2022b. The Complexity of k-Means Clustering when Little is Known. In *International Conference on Machine Learning, ICML 2022*, volume 162 of *Proc. of Machine Learning Research*, 6960–6987.

Ganian, R.; Kanj, I. A.; Ordyniak, S.; and Szeider, S. 2018. Parameterized Algorithms for the Matrix Completion Problem. In *Proc. of the 35th International Conference on Machine Learning, ICML 2018*, volume 80 of *Proc. of Machine Learning Research*, 1642–1651.

Ganian, R.; Kim, E. J.; and Szeider, S. 2022. Algorithmic Applications of Tree-Cut Width. *SIAM J. Discret. Math.*, 36(4): 2635–2666.

Ganian, R.; and Korchemna, V. 2021. The Complexity of Bayesian Network Learning: Revisiting the Superstructure. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, 430–442.

Ganian, R.; and Korchemna, V. 2022. Slim Tree-Cut Width. In *17th International Symposium on Parameterized and Exact Computation, IPEC 2022*, volume 249 of *LIPIcs*, 15:1–15:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Ganian, R.; and Ordyniak, S. 2018. The complexity landscape of decompositional parameters for ILP. *Artif. Intell.*, 257: 61–71.

Gözüpek, D.; Özkan, S.; Paul, C.; Sau, I.; and Shalom, M. 2017. Parameterized complexity of the MINCCA problem on graphs of bounded decomposability. *Theor. Comput. Sci.*, 690: 91–103.

Grüttemeier, N.; and Komusiewicz, C. 2022. Learning Bayesian Networks Under Sparsity Constraints: A Parameterized Complexity Analysis. *J. Artif. Intell. Res.*, 74: 1225–1267.

Grüttemeier, N.; Komusiewicz, C.; and Morawietz, N. 2021. On the Parameterized Complexity of Polytree Learning. In *Proc. of the 30th International Joint Conference on Artificial Intelligence*, 4221–4227.

Harks, T.; Henle, M.; Klimm, M.; Matuschke, J.; and Schedel, A. 2022. Multi-Leader Congestion Games with an Adversary. In *Proc. of the 36th AAAI Conference on Artificial Intelligence*, 5068–5075.

Huang, T.; Li, J.; Koenig, S.; and Dilkina, B. 2022. Anytime Multi-Agent Path Finding via Machine Learning-Guided Large Neighborhood Search. In *Proc. of the 36th AAAI Conference on Artificial Intelligence*, 9368–9376.

Hunter, P.; and Kreutzer, S. 2008. Digraph measures: Kelly decompositions, games, and orderings. *Theoretical Computer Science*, 399(3): 206–219.

Jalota, D.; Solovey, K.; Tsao, M.; Zoepf, S.; and Pavone, M. 2023. Balancing fairness and efficiency in traffic routing via interpolated traffic assignment. *Autonomous Agents and Multi-Agent Systems*, 37(2): 32.

Johnson, T.; Robertson, N.; Seymour, P. D.; and Thomas, R. 2001. Directed Tree-Width. *J. Comb. Theory, Ser. B*, 82(1): 138–154.

Koana, T.; Froese, V.; and Niedermeier, R. 2023. The complexity of binary matrix completion under diameter constraints. *J. Comput. Syst. Sci.*, 132: 45–67.

Koana, T.; Korenwein, V.; Nichterlein, A.; Niedermeier, R.; and Zschoche, P. 2021. Data Reduction for Maximum Matching on Real-World Graphs: Theory and Experiments. *ACM J. Exp. Algorithmics*, 26: 1.3:1–1.3:30.

Korhonen, J. H.; and Parviainen, P. 2015. Tractable Bayesian Network Structure Learning with Bounded Vertex Cover Number. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, 622–630.

Kunniyur, S.; and Srikant, R. 2003. End-to-end congestion control schemes: utility functions, random losses and ECN marks. *IEEE/ACM Trans. on Networking*, 11(5): 689–702.

Marchesi, A.; Castiglioni, M.; and Gatti, N. 2019. Leadership in Congestion Games: Multiple User Classes and Non-Singleton Actions. In *Proc. of the 28th International Joint Conference on Artificial Intelligence*, 485–491.

Marx, D.; and Wollan, P. 2014. Immersions in Highly Edge Connected Graphs. *SIAM J. Discret. Math.*, 28(1): 503–520.

Meir, R.; Tennenholtz, M.; Bachrach, Y.; and Key, P. B. 2012. Congestion Games with Agent Failures. In *Proc. of the 26th AAAI Conference on Artificial Intelligence*.

Nesetril, J.; and de Mendez, P. O. 2012. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and its Applications. Oxford: Oxford University Publishing.

Ordyniak, S.; and Szeider, S. 2013. Parameterized Complexity Results for Exact Bayesian Network Structure Learning. *J. Artif. Intell. Res.*, 46: 263–302.

Pham, Q. A.; Hà, M. H.; Vu, D. M.; and Nguyen, H. H. 2022. A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Roaming Delivery Locations. In *Proc. of the 32nd International Conference on Automated Planning and Scheduling*, 297–306.

Porschen, S.; Schmidt, T.; Speckenmeyer, E.; and Wotzlaw, A. 2014. XSAT and NAE-SAT of linear CNF classes. *Discret. Appl. Math.*, 167: 1–14.

Robertson, N.; and Seymour, P. D. 1986. Graph Minors. II. Algorithmic Aspects of Tree-Width. *J. Algorithms*, 7(3): 309–322.

Rosenthal, R. W. 1973. A class of games possessing pure-strategy Nash equilibria. *International J. of Game Theory*.

Schmidt, T. 2010. *Computational complexity of SAT, XSAT and NAE-SAT for linear and mixed Horn CNF formulas*. Ph.D. thesis, University of Cologne.

Sharon, G. 2021. Alleviating Road Traffic Congestion with Artificial Intelligence. In *IJCAI*, 4965–4969.

Sharon, G.; Albert, M.; Rambha, T.; Boyles, S. D.; and Stone, P. 2018. Traffic Optimization for a Mixture of Self-Interested and Compliant Agents. In *Proc. of the 32nd AAAI Conference on Artificial Intelligence*, 1202–1209.

Sharon, G.; Hanna, J. P.; Rambha, T.; Levin, M. W.; Albert, M.; Boyles, S. D.; and Stone, P. 2017a. Real-time adaptive tolling scheme for optimized social welfare in traffic networks. In *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems*.

Sharon, G.; Levin, M. W.; Hanna, J. P.; Rambha, T.; Boyles, S. D.; and Stone, P. 2017b. Network-wide adaptive tolling for connected and automated vehicles. *Transportation Research Part C: Emerging Technologies*, 84: 142–157.

Wang, K.; Xu, L.; Perrault, A.; Reiter, M. K.; and Tambe, M. 2022. Coordinating Followers to Reach Better Equilibria: End-to-End Gradient Descent for Stackelberg Games. In *Proc. of the 36th AAAI Conference on Artificial Intelligence*, 5219–5227.

Wollan, P. 2015. The structure of graphs not admitting a fixed immersion. *J. Comb. Theory, Ser. B*, 110: 47–66.

Wollan, P.; and Marx, D. 2014. Immersions in Highly Edge Connected Graphs. *SIAM J. Discrete Math.*, 28(1): 503–520.