Theory of Computing Systems
https://doi.org/10.1007/s00224-021-10038-9

The Minimum Tollbooth Problem in Atomic Network Congestion Games with Unsplittable Flows

Julian Nickerl[1]

Accepted: 18 February 2021Published online: 29 March 2021

Abstract
This work analyzes the minimum tollbooth problem in atomic network congestion games with unsplittable flows. The goal is to place tolls on edges, such that there exists a pure Nash equilibrium in the tolled game that is a social optimum in the untolled one. Additionally, we require the number of tolled edges to be the minimum. This problem has been extensively studied in non-atomic games, however, to the best of our knowledge, it has not been considered for atomic games before. By a reduction from the **weighted CNF SAT** problem, we show both the **NP**-hardness of the problem and the **W[2]**-hardness when parameterizing the problem with the number of tolled edges. On the positive side, we present a polynomial time algorithm for networks on series-parallel graphs that turns any given state of the untolled game into a pure Nash equilibrium of the tolled game with the minimum number of tolled edges.

Keywords Atomic network congestion games · Minimum tollbooth problem · Series-parallel graph · Social optimum · Unsplittable flow · Weighted CNF SAT

1 Introduction

A class of games that are guaranteed to admit at least one pure Nash equilibrium are congestion games, as introduced by Rosenthal [27]. However, several well-known results (for example presented by Christodoulou and Koutsoupias [10] or Roughgarden and Tardos [28]) indicate, that playing a Nash equilibrium rarely leads to a state that is socially beneficial. Especially in network congestion games, an approach to push the players towards a social optimum is the levy of tolls on edges of the network. The goal of these tolls is to turn a social optimum of the untolled game into a pure Nash equilibrium of the tolled game. The application of such tolls was shown

✉ Julian Nickerl
julian.nickerl@uni-ulm.de

[1] Ulm University, Ulm, Germany

to be highly beneficial [6, 12]. If we allow the number of players to be infinite, each routing an infinitesimal amount of flow through the network, it is well known that tolls fulfilling the property exist and can be efficiently computed [5, 11, 17, 26].

Games with a finite number of players are called atomic. The use of tollbooths in atomic games has a comparably short history. One of the main differences to the previous problem is that the strategy change of a single player has a substantial impact on all other players. Additionally, for the non-atomic games, it can be assumed that the resulting pure Nash equilibrium is unique, requiring only light restrictions, see, e.g., Roughgarden and Tardos [28]. As shown by Orda, Rom, and Shimkin, [25] this is not necessarily the case in atomic games.

Further differentiating the problem leads to the question: Does it suffice that there is at least one Nash equilibrium that corresponds to a social optimum, or must all Nash equilibria correspond to a social optimum? Fotakis, Karakostas, and Kolliopoulos analyzed whether tolls fulfilling the desired properties exist. They conclude in [18] that in the latter case the existence cannot be guaranteed even for simple parallel-link networks. On the other hand, for the former case, they can confirm the existence of tolls in games where all players share the same source node and give a polynomial time algorithm calculating the tolls. Other work on this topic is, e.g., Fotakis and Spirakis [19] or Caragiannis, Kaklamanis, and Kanellopoulos [8]. Another line of research allows the players to split up their flow [14, 30]. We focus on games with unsplittable flows; therefore we do not go into detail with this other branch.

Several papers have focused on special cases of the above problem, restricting, e.g., the height of the tolls [7], the set of tollable edges [20], or knowledge about the number of players [13]. We consider here the minimum tollbooth problem, first introduced by Hearn and Ramana [22]. In this problem, we additionally try to minimize the number of tolled edges. Work on this topic is mainly restricted to non-atomic games, with several heuristics as a result (e.g., [1–3, 21, 29]). The **NP**-hardness of the non-atomic case with multiple commodities was shown by Bai, Hearn, and Lawphongpanich in [1] and later for the single commodity case by Basu, Lianeas, and Nikolova in [4]. On the positive side, Basu, Lianeas, and Nikolova give a polynomial time algorithm, if the network is a series-parallel graph [4].

Contribution In this work, we analyze the complexity of the minimum tollbooth problem for atomic network congestion games with unsplittable flows. As mentioned before, it is already known that such tolls may not exist considering the case in which every pure Nash equilibrium of the tolled game must correspond to a social optimum of the original one [18]. Therefore, we focus on the problem in which it is only required that there exist a pure Nash equilibrium that corresponds to a social optimum of the untolled game. By a reduction from the **weighted CNF SAT** problem, we show that the problem is **NP**-hard. Furthermore, we also prove that the parameterized problem considering the number of tollbooths as the parameter is **W[2]**-hard, giving evidence that the parameterized problem is not in **FPT**. An additional spin-off result originates in the nature of the reduction: Finding a social optimum in atomic network congestion games with unsplittable flows is hard, supporting and extending existing results (see, e.g., Chakrabarty, Mehta, and Nagarajan [9] or Meyers and Schulz [24]).

We also show that for a non-trivial graph class, it is possible to efficiently calculate a solution with the smallest possible number of toll booths. Based on the algorithm by Basu, Lianeas, and Nikolova from [4], we construct a polynomial time algorithm on games based on series-parallel graphs that turns any given state of the untolled game into a pure Nash equilibrium of the tolled game with the minimum number of tolled edges.

2 Preliminaries

This section introduces essential general definitions. In an attempt to shorten some statements, we will sometimes use the expression $[d]$ for the set $\{1, ..., d\}$ for any natural number d throughout the work.

Definition 1 An *atomic network congestion game* is a tuple

$$\Gamma = (G, N, (c_e)_{e\in E}, (s_i, t_i)_{i\in N})$$

with $G = (V, E)$ being an undirected graph, $N = \{1, ..., n\}$ a set of players, c_e non-negative, monotonically non-decreasing cost functions from the set of non-negative integers of to the real numbers, and (s_i, t_i) source-sink pairs with $s_i, t_i \in V$. The strategies for a player i are the paths from s_i to t_i. A state $S = (S_1, ..., S_n)$ is a vector of strategies of players (S_i is a strategy of player i). The congestion $n_e(S)$ on an edge e is the number of players using that edge in their strategy in state S. The cost of a player in state S is $\gamma_i(S) = \sum_{e\in S_i} c_e(n_e(S))$, where S_i is the strategy of player i in S. The *social cost* $SC(S) = \sum_{e\in E} \left(c_e(n_e(S))n_e(S)\right)$ of a state S is the sum of the costs of all players in S. A *social optimum* is a state with minimum social cost. A state S is a pure Nash equilibrium, if for every player i and every state S', $\gamma_i(S) \leq \gamma_i((S'_i, S_{-i}))$. Hereby, the state (S'_i, S_{-i}) denotes player i playing his strategy from S' and all other players remaining on their strategy of state S.

Definition 2 A *toll vector* $\theta = (t_{e_1}, ..., t_{e_{|E|}})$, with $t_i \in \mathbb{R}_0^+$, describes tolls on edges. An edge e_i is called *tolled* if $t_{e_i} > 0$, and *untolled* otherwise. If an edge is tolled, we say it requires a *tollbooth*.

We say a state S is *implemented* in an atomic network congestion game Γ^θ if θ is a toll vector such that $\Gamma^\theta = (G, N, (c_e^\theta)_{e\in E}, (s_i, t_i)_{i\in N})$ with $c_e^\theta(S) = c_e(S) + t_e$ has S as a pure Nash equilibrium. The state is *optimally implemented* if for this purpose the minimum necessary number of tollbooths is required.

The *minimum tollbooth problem* (**MINTB**) on an atomic network congestion game Γ is the task of finding a toll vector θ such that a social optimum of Γ is optimally implemented in Γ^θ.

The upcoming section describes an **FPT**-reduction from the **weighted CNF SAT** problem to atomic **MINTB**.

Definition 3 A Boolean formula F is in conjunctive normal form (**CNF**) if $F = C_1 \wedge ... \wedge C_n$. Hereby, C_i is called clause and must have the form $C_i = (u_{i,1} \vee ... \vee u_{i,k_i})$. The $u_{i,j}$ are called literals, with $u_{i,j} \in \{x_1, ..., x_n\} \cup \{\overline{x_1}, ..., \overline{x_n}\}$. The x_i are called variables, with $x_i \in \{0, 1\}$.

Definition 4 weighted CNF SAT:

Input:	A Boolean formula F in **CNF**
Parameter:	A positive integer k
Question:	Does F have a satisfying assignment with weight k?

The weight of an assignment is its Hamming weight, i.e., the number of variables with value 1.

It is well known, that **weighted CNF SAT** is a **W[2]**-complete problem, see, e.g., chapter 12 of [16] or chapter 13 of [15]. This gives evidence that the problem does not lie in the class **FPT**.

Definition 5 Let $\mathscr{A}$ be a parameterized problem with parameter k. We say that $\mathscr{A}$ is *fixed-parameter tractable* (**FPT**), if there exists an algorithm solving $\mathscr{A}$ with running time in $\mathcal{O}(p(n) \cdot f(k))$. Here, $p(n)$ is a polynomial that depends on the problem size of $\mathscr{A}$ but not on the parameter k, and $f(k)$ is any function that only depends on the parameter k.

Definition 6 The *W-hierarchy* is a collection of complexity classes, with **FPT** = **W[0]** and **W[i]**$\subseteq$**W[j]** for all $i < j$. For a given t, the class **W[t]** encompasses parameterized decision problems of the form (x, k) with the parameter k that are reducible to **Weighted Weft-t Depth-h Circuit-SAT**, for some constant h. **Weighted Weft-t Depth-h Circuit-SAT** is defined as follows:

Input:	A Boolean circuit C, consisting of fan-in-2 and unbounded-fan-in gates. On any path to the root, the total number of fan-in-2 and unbounded-fan-in gates is at most h, while the number of unbounded-fan-in gates is at most t.
Parameter:	A positive integer k
Question:	Does C have a satisfying assignment with weight k?

A problem Q is $W[i]$-hard, if there exists a $W[i]$-hard Problem P and a function $\phi : P \rightarrow Q$ with the following properties [23]:

- $\phi(x)$ is a yes-instance of $Q \Leftrightarrow x$ is a yes-instance of P
- $\phi(x)$ can be computed in time $f(k) \cdot |x|^{\mathcal{O}(1)}$, where k is the parameter of x.
- If the parameter of x and k' is the parameter of $\phi(x)$, then $k' \leq g(k)$ for some function g.

Such a reduction is called **FPT**-reduction.

For more information on parameterized problems, we highly recommend [16] by Downey and Fellows or [15] by Cygan et al.

3 Atomic MINTB is Hard

We will reduce **weighted CNF SAT** to atomic **MINTB** in such a way, that the minimum number of necessary tollbooths is exactly the minimum weight of a satisfying assignment of the formula. The reduction will be polynomial, with the runtime independent of the parameter. Hence the reduction will fulfill the requirements of an **FPT**-reduction. We construct an atomic network congestion game Γ_F based on formula F, where F is in **CNF** with n variables $x_1, ..., x_n$ and m clauses $C_1, ..., C_m$. We call $\Lambda^0_{x_i}$ ($\Lambda^1_{x_i}$) the number of occurrences of variable x_i in F in negative (positive) form, $\Lambda_{x_i} = \Lambda^0_{x_i} + \Lambda^1_{x_i}$ and $\Lambda = \sum_{i=1}^{n} \Lambda_{x_i}$. The game will be played by a total of $m + \Lambda + n$ players, each fulfilling a specific role. The m players, called *clause players*, each try to find a strategy that can be identified as satisfying a different clause. This will be done by each player choosing the occurrence of a literal in the clause in order to satisfy it. The Λ players, or *occurrence players*, ensure that while satisfying the clauses, a variable cannot be assigned both value 0 and 1. Lastly, the n players are the *variable players*, that state which edges have to be tolled to optimally implement a social optimum. Most players have both different sources and different sinks.

We start by constructing a graph for the game. For easier reference, the graph is split into three components:

Let $G(V_1 \cup V_2, E_1 \cup E_2 \cup E_3)$ be the undirected graph with

$$
\begin{aligned}
V_1 &= \{s\} \\
&\cup \{v_i, v_i^0, v_i^1 | i \in [n]\} \\
&\cup \{l^0_{x_i,j,k}, r^0_{x_i,j,k}, z^0_{x_i,j} | i \in [n],\ j \in [\Lambda^0_{x_i}],\ k \in [\Lambda^1_{x_i}]\} \\
&\cup \{l^1_{x_i,j,k}, r^1_{x_i,j,k}, z^1_{x_i,j} | i \in [n],\ j \in [\Lambda^1_{x_i}],\ k \in [\Lambda^0_{x_i}]\} \\
&\cup \{c_i | i \in [m]\}, \\
E_1 &= \{\{s, v_i\}, \{v_i, v_i^0\}, \{v_i, v_i^1\} | i \in [n]\} \\
&\cup \{\{v_i^0, l^0_{x_i,j,1}\}, \{l^0_{x_i,j,k}, r^0_{x_i,j,k}\}, \{r^0_{x_i,j,k}, l^0_{x_i,j,k+1}\} | i \in [n],\ j \in [\Lambda^0_{x_i}], \\
&\qquad\qquad k \in [\Lambda^1_{x_i}]\} \\
&\cup \{\{v_i^1, l^1_{x_i,j,1}\}, \{l^1_{x_i,j,k}, r^1_{x_i,j,k}\}, \{r^1_{x_i,j,k}, l^1_{x_i,j,k+1}\} | i \in [n],\ j \in [\Lambda^1_{x_i}], \\
&\qquad\qquad k \in [\Lambda^0_{x_i}]\} \\
&\cup \{\{r^0_{x_i,j,\Lambda^1_{x_i}}, z^0_{x_i,j}\}, \{z^0_{x_i,j}, c_k\} | i \in [n],\ j \in [\Lambda^0_{x_i}], \\
&\qquad\qquad k \text{ is the } j \text{ th smallest index s.t. } \overline{x_i} \in C_k\} \\
&\cup \{\{r^1_{x_i,j,\Lambda^0_{x_i}}, z^1_{x_i,j}\}, \{z^1_{x_i,j}, c_k\} | i \in [n],\ j \in [\Lambda^1_{x_i}], \\
&\qquad\qquad k \text{ is the } j \text{ th smallest index s.t. } x_i \in C_k\} \\
&\cup \{\{v_i^0, z^0_{x_i,j}\} | \forall i \ s.t.\ \Lambda^1_{x_i} = 0,\ j \in [\Lambda^0_{x_i}]\} \\
&\cup \{\{v_i^1, z^1_{x_i,j}\} | \forall i \ s.t.\ \Lambda^0_{x_i} = 0,\ j \in [\Lambda^1_{x_i}]\}.
\end{aligned}
$$

$$
\begin{aligned}
V_2 &= \{o^0_{i,j} | i \in [n],\, j \in [\Lambda^0_{x_i}]\} \\
&\quad \cup \{o^1_{i,j} | i \in [n],\, j \in [\Lambda^1_{x_i}]\} \\
E_2 &= \{\{o^0_{x_i,j}, z^0_{x_i,j}\} | i \in [n],\, j \in [\Lambda^0_{x_i}]\} \\
&\quad \cup \{\{o^0_{x_i,j}, l^1_{x_i,1,j}\} | i \in [n],\, j \in [\Lambda^0_{x_i}]\} \\
&\quad \cup \{\{o^1_{x_i,j}, z^1_{x_i,j}\} | i \in [n],\, j \in [\Lambda^1_{x_i}]\} \\
&\quad \cup \{\{o^1_{x_i,j}, l^0_{x_i,1,j}\} | i \in [n],\, j \in [\Lambda^1_{x_i}]\} \\
&\quad \cup \{\{r^0_{x_i,j,k}, l^0_{x_i,j+1,k}\} | i \in [n],\, j \in [\Lambda^0_{x_i}],\, k \in [\Lambda^1_{x_i}]\} \\
&\quad \cup \{\{r^1_{x_i,j,k}, l^1_{x_i,j+1,k}\} | i \in [n],\, j \in [\Lambda^1_{x_i}],\, k \in [\Lambda^0_{x_i}]\} \\
&\quad \cup \{\{r^0_{x_i,\Lambda^0_{x_i},k}, c_j\} | i \in [n], k \in [\Lambda^1_{x_i}],\, j \text{ is the } k \text{ th smallest index s.t. } x_i \in C_j\} \\
&\quad \cup \{\{r^1_{x_i,\Lambda^1_{x_i},k}, c_j\} | i \in [n], k \in [\Lambda^0_{x_i}],\, j \text{ is the } k \text{ th smallest index s.t. } \overline{x_i} \in C_j\}. \\
E_3 &= \{\{v^0_i, v^1_i\} | i \in [n]\}.
\end{aligned}
$$

The cost functions for the edges in E_1 are a constant 7 for each $\{v_i, v^1_i\}$ edge, a constant 2 for each $\{v_i, v^0_i\}$ edge and a constant 0 for each $\{s, v_i\}$ edge. All other edges have cost 0 for one, and ∞ (symbolizing a very large number) for any other number of players. Now to edges in E_2. The edges $\{r^b_{x_i,j,k}, c_l\}$, $\{o^1_{x_i,j}, l^0_{x_i,1,j}\}$, and $\{o^0_{x_i,j}, l^1_{x_i,1,j}\}$ have cost 6 for one player and ∞ otherwise, while the edges $\{o^1_{x_i,j}, z^1_{x_i,j}\}$ and $\{o^0_{x_i,j}, z^0_{x_i,j}\}$ have cost 12 for one player and ∞ otherwise. All other edges have cost 0 for one player and ∞ for any other number of players. The cost function for all edges e in E_3, is $c_e(x) = 2x$, where x is the number of players that use edge e.

What remains to define are the source-sink node pairs for each player. Each clause player i has source s and sink c_i. Each of the Λ occurrence players represents the occurrence of a variable x_i in clause C_j. If the occurrence is positive, the player has the source $o^1_{x_i,k}$, where j is the kth-smallest index s.t. x_i occurs positively in C_j. Analogously the source is $o^0_{x_i,k}$ if the occurrence is negative. In both cases, the sink is c_j. A variable player i has source v^0_i and sink v^1_i.

Figure 1 illustrates the graph on the example $F = A \wedge (\overline{A} \vee B) \wedge (\overline{A} \vee \overline{B} \vee \overline{C})$. In an attempt to make the graph clearer, some nodes appear multiple times; however they all represent one single node. Nodes with this property are displayed as rectangles. If they have the same name, they represent the same single node.

We continue to prove several useful properties of the above construction that will allow us to prove the desired result. In the following, we will say that a clause player chooses to assign 0 (1) to a variable x, if in her strategy she leaves node v^0 (v^1) to a node $l^0_{x,i,j}$ or $z^0_{x,i}$ ($l^1_{x,i,j}$ or $z^1_{x,i}$).

Lemma 1 *Let F be satisfiable. There exists a state in Γ_F where each player has cost less than ∞.*

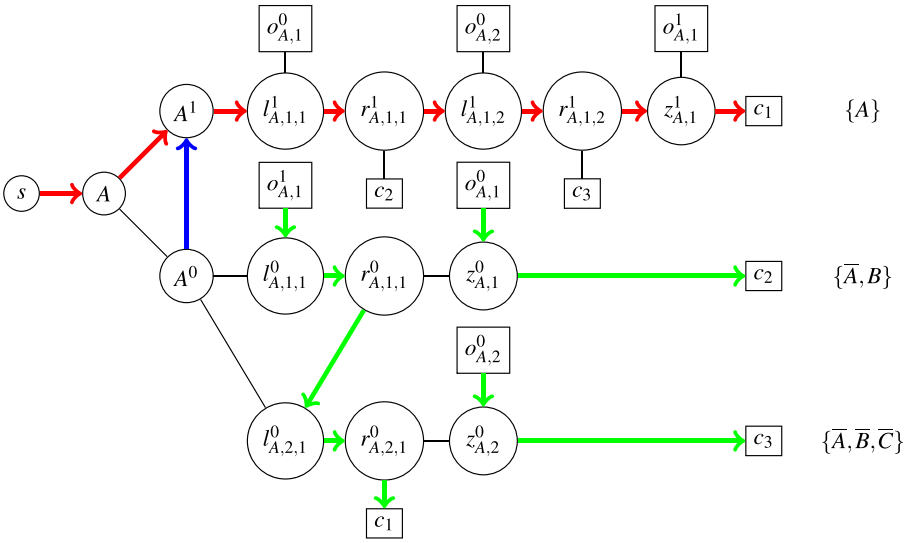

Fig. 1 The segment of the graph G for the formula $F = A \wedge (\overline{A} \vee B) \wedge (\overline{A} \vee \overline{B} \vee \overline{C})$ concerning variable A. Depicted are the strategies of one clause player (red arrows), three occurrence players (green arrows), and one variable player (blue arrow) in a social optimum

Proof Figure 1 demonstrates the intended behavior of the players. Observe that if all clause players are consistent with their assignments of the variables, the only edges that are used by more than one player are the $\{s, v_i\}$ edges. Hence none of the edges produces a cost of ∞ and a state is reached where every player has cost less than ∞. The clause players are forced to make the assignments inconsistent only if the formula is unsatisfiable.

Let $\mathscr{A}$ be a satisfying assignment of F. For each clause player, there exists a variable x, s.t., she can reach her clause by assigning $\mathscr{A}(x)$ to it. The cost for each clause player is at most 7. All variable players can choose the corresponding edges $\{v^0, v^1\}$, resulting in a cost of 2. The occurrence players reach their corresponding clauses again based on $\mathscr{A}$. If x occurs negatively in a clause C_i and $\mathscr{A}(x) = 0$, then all clause players assign 0 to the variable if they use it. Consequently, all of the $\{l^1_{x,j,k}, r^1_{x,j,k}\}$ edges are unused and the occurrence player can choose the zigzag path, resulting in a cost of 12. If $\mathscr{A}(x) = 1$ the clause players assign 1 to x if appropriate and the edge $\{z^0_{x,j}, C_i\}$ is unused. Hence the occurrence player can choose the direct path to the clause with cost 12. The symmetric cases apply when the variable occurs positively. □

Lemma 1 implies, that the social cost in the social optimum is less than ∞ if F is satisfiable.

Lemma 2 *Let F be satisfiable. In a social optimum, if two clause players choose the same variable in order to satisfy their clause, they either both assign it 0 or both assign it 1, but not a mixture.*

Proof The occurrence players ensure this property. Observe, that this type of player always has total cost 12 in a social optimum, because they have to either choose edges $\{o^b_{x,i}, l^{b'}_{x,1,i}\}$ and $\{r^b_{x,\Lambda^0_{x,j}}, c_k\}$ or an edge $\{o^b_{x,i}, z^b_{x,i}\}$ in order to reach their clause and exactly one of these choices always suffices. The other edges in their strategies have cost 0. We prove by contradiction that the occurrence players lead to the effect stated in the lemma.

Assume that there are two clause players, P_1 and P_2, that, in a social optimum, satisfy their respective clause by choosing variable x, however P_1 assigns x the value 1, while P_2 assigns it 0. There exists an occurrence player representing the positive occurrence of x in the clause satisfied by P_1. If she chooses the edge $\{o^1_{x,i}, z^1_{x,i}\}$ in the beginning, she can only leave the z node through an edge already use by P_1. This increases the cost of both players to ∞. If, on the other hand, she chooses the edge $\{r^0_{x,\Lambda^0_{x},i}, c_k\}$, she has to follow a zigzag path of the form $o^1_{x,i} - l^0_{x,1,i} - r^0_{x,1,i} - l^0_{x,2,i} - r^0_{x,2,i} - \cdots - r^0_{x,\Lambda^0_{x},i}$. In doing so, she has used at least one edge on every path that can be used to assign the value 0 to x. However, P_2 has to use at least one of these edges as well, hence the cost of P_2 and the occurrence player increases to ∞. Since a state exists where all players have cost less than ∞, none of the two options can be a social optimum. This is a contradiction. □

The property of Lemma 2 guarantees that in a social optimum, the clause players choose a consistent assignment of the variables. It is not clear yet however, that the chosen assignment is one that satisfies the formula.

Lemma 3 *Let F be satisfiable. In a social optimum, each clause player in Γ_F chooses a strategy that represents the satisfaction of a different clause.*

Proof Each clause player has the node of a different clause as sink. In order to reach the clause she has to choose a variable x and then follow the l, r, z nodes until she finally reaches the clause. Such a path costs her at most 7 and indicates, that she chooses variable x in order to satisfy the clause. Defecting to another node on the way will only increase her cost by at least 6 without reducing another player's cost, hence it is not beneficial to the social cost. □

Since every clause player satisfies a different clause and the number of clause players coincides with the number of clauses, Lemma 3 shows that the assignment encoded in the strategies of the clause players satisfies F. If there exists a variable that is not chosen by any of the players to satisfy a clause, we can assume that its value is irrelevant and assign it the value 0. We call this assignment *encoded* by the clause players. A final property is that in the social optimum indeed the minimum satisfying assignment is encoded. However, before we further discuss this property, note the effect of the variable players illustrated in Fig. 2. It makes clear, that in a social optimum a clause player chooses the path $v_i - v^1_i$ with cost 7 instead of $v_i - v^0_i - v^1_i$ with cost at least 6.

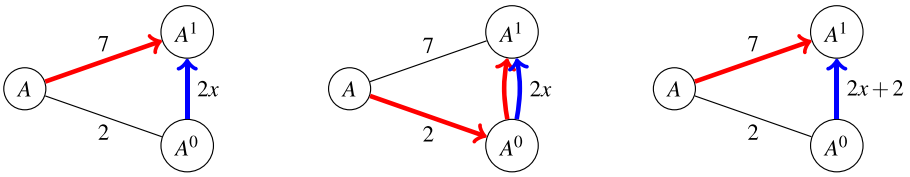

Fig. 2 A demonstration of the effect of the variable player (blue arrow). The clause player (red arrow) leaves the social optimum (left graph) for the pure Nash equilibrium (center graph). Tolling an edge prevents this defection (right graph)

Lemma 4 *Let F be satisfiable. In a social optimum, the assignment $\mathscr{A}$ encoded by the clause players is the minimum satisfying assignment.*

Proof We prove this property through contradiction. Let $\mathscr{A}$ be the minimum satisfying assignment, and $\mathscr{B}$ the assignment encoded by the clause players in the social optimum with a higher weight than $\mathscr{A}$. In both cases, each occurrence player has cost 12 and each variable player has cost 2. A clause player that assigns a variable the value 0 has cost 2, and one who assigns it the value 1 has cost 7. Let $\omega(X)$ indicate the weight of an assignment X. The difference in the costs is then $7\cdot(\omega(\mathscr{A})-\omega(\mathscr{B}))+2\cdot((n-\omega(\mathscr{A}))-(n-\omega(\mathscr{B}))) < 0$. Hence, the social cost of the state where the clause players encode $\mathscr{A}$ is less than the social cost of the state where $\mathscr{B}$ is encoded. This is a contradiction, as we have assumed that the state encoding $\mathscr{B}$ is the social optimum. □

Now everything is set up to finalize the reduction.

Theorem 1 *Solving atomic **MINTB** is **NP**-hard and **W[2]**-hard with the number of required toll booths as the parameter.*

Proof Let F be a Boolean formula in **CNF**. If F is satisfiable, Lemma 4 ensures that a social optimum of Γ_F encodes a minimum satisfying assignment. This state is not a Nash equilibrium, unless all variables are assigned 0. Let x be a variable that is assigned 1. For exactly one clause player, choosing the edges $v_i - v_i^0 - v_i^1$ is cheaper (cost of 6) than the direct path $v_i - v_i^1$ (cost of 7). However, the cheaper route increases the social cost. Placing a single tollbooth on the $\{v_i^0, v_i^1\}$ edge with toll 2 resolves this problem. Figure 2 illustrates the scenario.

From the placement of the tollbooths, a minimum satisfying assignment $\mathscr{A}$ for F can be constructed: If there is a tollbooth on $\{v^0, v^1\}$, set $\mathscr{A}(x) = 1$. Otherwise set $\mathscr{A}(x) = 0$. This assignment exactly corresponds to the one encoded by the clause players as mentioned in Lemma 4. The number of tollbooths coincides with the weight of the encoded assignment.

If the formula is unsatisfiable, we can construct an assignment in the same way as above, and in the end check whether the assignment satisfies the formula. If it does not, F is unsatisfiable.

Given the solution to **MINTB** on Γ_F it can be decided in time $f(k) \cdot |F|^{\mathcal{O}(1)}$ whether F is a yes instance or not. Γ_F can be constructed from F in polynomial

time, independent of the parameter. In particular, F is a yes-instance iff Γ_F is a yes-instance. The parameter in both cases is the same. Therefore, the reduction is an **FPT**-reduction.

This finalizes the reduction from **weighted CNF SAT**, leading to the stated result. □

To know where to place the tollbooths and therefore the minimal satisfying assignment of the original formula, it suffices to know the social optimum. A direct consequence is that finding a social optimum must be hard, extending the existing **NP**-hardness results [9, 24] to **W[2]**-hardness of the parameterized variant.

Corollary 1 *Finding a social optimum of a game Γ is **NP**-hard. Additionally, finding a social optimum that can be implemented in Γ with at most k tollbooths is **W[2]**-hard, with k being the parameter.*

4 Optimally Implementing a State in Polynomial Time

This section presents an algorithm that optimally implements a state S in an atomic network congestion game based on a series-parallel graph in polynomial time. We base the procedure on a similar approach from Basu, Lianeas, and Nikolova, who show the same result in [4] for non-atomic games. Starting at the simple base case of parallel-link networks, we inductively decide on the number of tolled edges for larger components, based on the optimality of the smaller ones. We do so by exploiting the recursive structure of series-parallel graphs.

Definition 7 A graph G with source s and sink t is called *series-parallel*, if

i) it consists of only a single edge.
ii) it is the result of combining two series-parallel graphs in series.
iii) it is the result of combining two series-parallel graphs in parallel.

A combination of graphs G_1 and G_2 in series means declaring the source of G_1 as the new global source, and the sink of G_2 as the new global sink. Additionally, the sink of G_1 and the source of G_2 are identified as one node in the new graph.

Combining G_1 and G_2 in parallel means identifying both sources as one node and setting it as the new global source, and respectively identifying both sinks as the new global sink.

Figure 3 illustrates the combinations. The center and right graph demonstrate the combination of two basic series parallel graphs in parallel and series, respectively.

A *series-parallel parse tree* of a series-parallel graph is the representation of the graph as a tree, where every leaf stands for a single edge, and every inner node represents either a combination of its two children in series or parallel. It can be constructed in linear time in the number of edges of the graph. For simplicity, in this paper, the leaves will represent parallel-link networks, i.e., networks consisting of two nodes connected by possibly several edges.

Given an atomic network congestion game on a series-parallel network, let T be the parse tree representation of that network, and S a given state that is to be

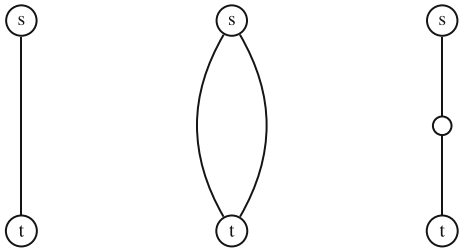

Fig. 3 Left: Basic series-parallel graph; Center: combination in parallel; Right: combination in series

implemented in that game. We will call a cost λ *enforceable* with η tollbooths, if by placing at most η tollbooths the costs can be adapted such that a newly joining player has cost at least λ, assuming all other players follow there strategy from S. For each node of T, we will create a list of tuples (η, λ) where λ is the highest cost enforceable with η tollbooths in the network represented by the regarded node. The values for η range from the minimum necessary to guarantee that all player follow their strategy from S to the minimum necessary to toll all paths through the network. Additionally, we will remember a single value λ_0. This value indicates the lowest cost that has to be enforced. It is the maximum of the highest cost of the player in S and the lowest cost of a newly joining player. The number of tuples per node cannot grow too large, as it is bounded by the number of edges.

As a side note we want to mention that this last value λ_0 is not necessary in the non-atomic setting, because there the cost functions are continuous. In non-atomic games it is assumed that single players have an infinitesimally small effect on the cost functions, so λ_0 would simply coincide with the cost of a player already in the game. Especially, in an equilibrium, all players would have the same cost.

Algorithm 1 creates the list for a leaf of the tree, which is a parallel-link network. For simplicity, we denote by $l_e = c_e(n_e(S))$, and by $l_e^+ = c_e(n_e(S) + 1)$.

Algorithm 1 ParallelLink.

Data: Graph $G = (V, E)$ represented by leaf v of the parse tree, state S
Set $list_v = []$;
Set $l_{\max} = \max\{l_e \mid e \in E\}$;
Set $\eta_{\min} = |\{e \mid e \in E \text{ and } l_e^+ < l_{\max}\}|$;
Set $\eta_{\max} = |E|$;
Set $\lambda_0 = \max\{\max\{l_e\}, \min\{l_e^+ \mid e \in E\}\}$
for $\eta = \eta_{\min}$ *to* $\eta_{\max}$ **do**
    Let $e_1, ..., e_k$ be all edges, and $l_1^+ \leq ... \leq l_k^+$
    $\lambda = \infty$;
    if $\eta \neq \eta_{\max}$ **then**
        $\lambda = l_{\eta+1}^+$;
    Append (η, λ) to $list_v$;

Lemma 5 *Algorithm 1 generates the correct list for a parallel link network.*

Proof Lines 2 and 3 of Algorithm 1 ensure, that at least the minimum number of edges necessary to implement the given state S in the parallel-link network is tolled. Every edge that would decrease the cost of a player, were she to change her strategy to it from the given state, has to be tolled. To find these edges, it is sufficient to compare their cost l_e^+ to the highest cost of any player ($l_{\max}$), and check whether the player would benefit from a defection.

Line 10 calculates the highest cost enforceable on a new player entering the network. When tolling η edges, the highest cost we could force on a newly entering player is reached when the η cheapest edges are tolled. The joining player then has at least the cost of the $\eta+1$ cheapest edge. Only when all edges are tolled, we can guarantee an arbitrarily high cost, marked by the ∞ symbol. Thus, the tuple $(\eta_{\max}, \infty)$ forms the last element of the list. □

From now on, since the lists are ordered with respect to the values of η, we will address elements from a list by index, so, e.g., η_1 refers to $\eta_{\min}$, and λ_i is the cost enforceable by η_i edges. Whenever we refer to the last element of a list, we use max as the index.

Following an inductive argument, we now show how to correctly form these lists for the inner nodes of the parse tree, assuming that the lists for the children are already correctly formed. To differentiate between the two children of the inner node r, we address one of them by v, and the other one by w. Some values will be labeled with r, v or w accordingly. A tuple for the list of r will originate from two tuples, one for each child node. We add two pointers, idx_v and idx_w, to each new tuple, referencing their origin.

Series Composition The series composition can be thought of as an upper graph and a lower graph, joined in a common center node. Observe, that all paths pass through this center node. This means in particular, that tollbooths placed in the upper graph cannot have an effect on the lower graph and vice versa. The two graphs can be handled separately and their lists joined together as follows:

The minimum number of edges that need to be tolled in the parent node r is the sum of the minimum necessary number of edges from the children, so $\eta_1^r = \eta_1^v + \eta_1^w$. The maximum $\eta^r_{\max^r}$ can be calculated by $\eta^r_{\max^r} = \min\{\eta^w_{\max^w}, \eta^v_{\max^v}\}$. If the enforceable cost in one of the graphs is ∞, the same holds for the whole graph, as any player has to pass through both the upper and the lower graph. To complete the list, we have to add a tuple for every η_i^r with $1 < i < \max^r$.

The corresponding costs can be calculated similarly. For a number of tolled edges η_i^r, we distribute them in the upper and lower graph such that the enforceable cost is maximized. It follows that $\lambda_i^r = \max\{\lambda_a^v + \lambda_b^w \mid \eta_i^r = \eta_a^v + \eta_b^w\}$, for all $i \in [\max^r]$. Accordingly, we set $\lambda_0^r = \lambda_0^v + \lambda_0^w$.

Parallel Composition In the parallel composition a player from one component may be able to reduce his cost by joining the other component, so there is more interaction between the two graphs than in the series composition. Let $c_{\max}$ be the highest cost of

a player in both v and w, if all players play according to the state S. Without loss of generality, let it be a player from v that has this cost $c_{\max}$. To determine the minimum necessary number of tolled edges, it suffices to check how many edges in w have to be tolled to force a joining player to have cost at least $c_{\max}$. The enforceable cost is then the minimum of the enforceable costs of the individual components.

Explicitly, the values of the tuples are $\eta_1^r = \eta_1^v + \min\{\eta_i^w \mid \lambda_i^w \geq c_{\max}, i > 0\}$, and $\eta_{\max^r}^r = \eta_{\max^v}^v + \eta_{\max^w}^w$. Again, we add a tuple for every η_i^r with $1 < i < \max^r$. The corresponding enforceable costs are $\lambda_i^r = \max\{\min\{\lambda_a^v, \lambda_b^w\} \mid \eta_i^r = \eta_a^v + \eta_b^w\}$, for all $i \in [\max^r]$. We set $\lambda_0^r = \max\{\lambda_0^v, \lambda_0^w\}$. It must be the maximum (let w.l.o.g. $\lambda_0^v \geq \lambda_0^w$), because otherwise a player from v may benefit from diverting to w if only λ_0^w is enforced.

Once the lists for all nodes are created, the value λ_0 of the root node tells us the cost that has to be enforced. Using the lists, we can decide the minimum number of tollbooths necessary for this, namely the η_i with the smallest index i s.t. $\lambda_i \geq \lambda_0$. We can then retrace the creation of the tuple (η_i, λ_i) to the leaf nodes of the tree, where we can decide which edges are to be tolled with which value. The recursive algorithm 2 tolls the edges accordingly. For simplicity, we assume the lists of each node to be globally accessible. The algorithm is initially called with r being the root node of the parse tree and $c_{in} = \lambda_0^r$ which is the smallest cost that has to be enforced.

Algorithm 2 PlaceTolls.

Data: Parse Tree T, current node r, cost c_{in}
if *r is a leaf node* **then**
  Let (V, E) be the parallel-link network represented by r;
  for *each edge $e \in E$ with $l_e^+ < c_{in}$* **do**
    set toll $t_e = c_{in} - l_e^+$;
  return
Let v = left child of r;
Let w = right child of r;
Set $i = \arg\min_i \{\lambda_i^r \geq c_{in} \mid i > 0\}$;
Let j be the idx_v of tuple (η_i^r, λ_i^r);
Let k be the idx_w of tuple (η_i^r, λ_i^r);
if *r is a series composition* **then**
  Set c_{out}^v and c_{out}^w, s.t.:
  - $c_{out}^v + c_{out}^w = c_{in}$,
  - $\lambda_j^v \geq c_{out}^v > \lambda_{j-1}^v$,
  - $\lambda_k^w \geq c_{out}^w > \lambda_{k-1}^w$,
  - $c_{out}^v \geq \lambda_0^v$,
  - $c_{out}^w \geq \lambda_0^w$;
else
  Set $c_{out}^v = \max\{c_{in}, \lambda_0^v\}$;
  Set $c_{out}^w = \max\{c_{in}, \lambda_0^w\}$;
Call PlaceTolls(T, v, c_{out}^v);
Call PlaceTolls(T, w, c_{out}^w);

Theorem 2 *Optimally implementing a state S in an atomic network congestion game based on a series-parallel graph with m edges is possible in runtime $\mathcal{O}(m^3)$.*

Proof We demonstrate that with the lists in each node of the parse tree as described above, Algorithm 2 optimally implements the given state S. The input c_{in} is the minimum cost that has to be enforced in the network represented by node r. The goal is to use as few tollbooths as possible for this task. The number of tollbooths necessary is the smallest η_i^r, s.t., $\lambda_i^r \geq c_{in}$

If r represents a parallel composition, we have to make sure that at least c_{in} is enforced in both parallel components. In some components it can occur that λ_0 is above c_{in}. In these cases we can instead continue with λ_0. This does not create a conflict, because this property was already considered in the calculation of c_{in} while generating the lists and no further tollbooths are required.

Secondly, if we consider a series composition, c_{in} must be appropriately split up into $c_{in} = c_{out}^v + c_{out}^w$. Several of these decompositions are possible, however we cannot guarantee that all of them lead to a consistent placement of the tollbooths. To resolve this issue, we first determine which tuple of r is appropriate to enforce c_{in}. Its index is $\arg\min_i\{\lambda_i^r \leq c_{in} | i > 0\}$, which requires the lowest number of tollbooths to enforce c_{in}. For this index, we have remembered which combination of list elements of the child nodes have lead to the desired tuple. We then choose c_{out}^v and c_{out}^w such that the remembered indices are the appropriate list elements under the same argument in v and w respectively.

Note that, for both compositions, the new c_{out} are chosen such that for the appropriate list elements, $\eta_{in}^r = \eta_{out}^v + \eta_{out}^w$.

Lastly, r can be a leaf node, representing a parallel link network. This is the base case of the recursive algorithm. All edges e with $l_e^+ < c_{in}$ have to be tolled.

Runtime Let m be the number of edges in the original network. The parse tree can be created in $\mathcal{O}(m)$ time, and also has size $\mathcal{O}(m)$. Creating the list at a leaf node is possible in $\mathcal{O}(m)$ time. For an inner node, we have to check at most $\mathcal{O}(m^2)$ combinations of elements from the children's lists. Therefore, creating the list for every node is possible in $\mathcal{O}(m^3)$ time. The computation time on each node in Algorithm 2 is bounded by $\mathcal{O}(m)$. The algorithm visits every node of the parse tree exactly once, leading to a runtime of $\mathcal{O}(m^2)$. In total, the runtime of the algorithm is $\mathcal{O}(m^3)$. □

5 Conclusion

This work analyses the complexity of the minimum tollbooth problem in atomic network congestion games with unsplittable flows.

By a reduction from the **weighted CNF SAT** problem, we show both the **NP**-hardness of the problem and, more importantly, the **W[2]**-hardness of a parameterized version. The parameter in the latter case is the number of necessary tollbooths to turn a social optimum of the untolled game into a pure Nash equilibrium of the tolled one.

A restriction to networks that are series-parallel graphs yields a polynomial time algorithm. The algorithm gives tolls, such that a given state is a pure Nash equilibrium of the tolled game while using the minimum number of toll booths. This algorithm is based on a similar method presented by Basu, Lianeas, and Nikolova in [4], who consider non-atomic games, exploiting the recursive structure of series-parallel graphs.

We assume that the last result holds due to the locally restricted effect of tollbooths in series-parallel graphs. The strong structure allows us to represent the necessary information of a whole component through a comparably small list. This would not be possible if slight changes in one part of the graph could have a big impact on some other distant part of the graph. Research on network classes with strong properties in locality may be fruitful to find further cases of polynomial time algorithms for **MINTB**.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Bai, L., Hearn, D.W., Lawphongpanich, S.: A heuristic method for the minimum toll booth problem. J. Glob. Optim. **48**(4), 533–548 (2010)
2. Bai, L., Rubin, P.A.: Combinatorial benders cuts for the minimum tollbooth problem. Oper. Res. **57**(6), 1510–1522 (2009)
3. Bai, L., Stamps, M.T., Harwood, R.C., Kollmann, C.J.: An evolutionary method for the minimum toll booth problem: The methodology Journal of Management Information and Decision Sciences (2008)
4. Basu, S., Lianeas, T., Nikolova, E.: New complexity results and algorithms for the minimum tollbooth problem. In: International Conference on Web and Internet Economics, pp. 89–103. Springer (2015)
5. Beckmann, M., McGuire, C.B., Winsten, C.B.: Studies in the Economics of Transportation. Yale University Press, London (1956)
6. Bilò, V., Vinci, C.: Dynamic taxes for polynomial congestion games. In: Proceedings of the 2016 ACM Conference on Economics and Computation, pp. 839–856. ACM (2016)
7. Bonifaci, V., Salek, M., Schäfer, G.: Efficiency of restricted tolls in non-atomic network routing games. In: International Symposium on Algorithmic Game Theory, pp. 302–313. Springer (2011)
8. Caragiannis, I., Kaklamanis, C., Kanellopoulos, P.: Taxes for linear atomic congestion games. In: ESA, pp. 184–195. Springer (2006)
9. Chakrabarty, D., Mehta, A., Nagarajan, V.: Fairness and optimality in congestion games. In: Proceedings of the 6th ACM Conference on Electronic Commerce, pp. 52–57. ACM (2005)
10. Christodoulou, G., Koutsoupias, E.: The price of anarchy of finite congestion games. In: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, pp. 67–73. ACM (2005)

11. Cole, R., Dodis, Y., Roughgarden, T.: Pricing network edges for heterogeneous selfish users. In: Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing, pp. 521–530. ACM (2003)
12. Cole, R., Dodis, Y., Roughgarden, T.: How much can taxes help selfish routing? J. Comput. Syst. Sci. **72**(3), 444–467 (2006)
13. Colini-Baldeschi, R., Klimm, M., Scarsini, M.: Demand-independent optimal tolls (2018)
14. Cominetti, R., Correa, J.R., Stier-Moses, N.E.: The impact of oligopolistic competition in networks. Oper. Res. **57**(6), 1421–1437 (2009)
15. Cygan, M., Fomin, F.V., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms, vol. 3. Springer, Berlin (2015)
16. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Berlin (1999)
17. Fleischer, L., Jain, K., Mahdian, M.: Tolls for heterogeneous selfish users in multicommodity networks and generalized congestion games. In: Proceedings of the Fourty-Fifth Annual IEEE Symposium on Foundations of Computer Science, pp. 277–285. IEEE (2004)
18. Fotakis, D., Karakostas, G., Kolliopoulos, S.G.: On the existence of optimal taxes for network congestion games with heterogeneous users. In: International Symposium on Algorithmic Game Theory, pp. 162–173. Springer (2010)
19. Fotakis, D., Spirakis, P.G.: Cost-balancing tolls for atomic network congestion games. Internet Math. **5**(4), 343–363 (2008)
20. Harks, T., Kleinert, I., Klimm, M., Möhring, R.H.: Computing network tolls with support constraints. Networks **65**(3), 262–285 (2015)
21. Harwood, R.C., Kollmann, C.J., Stamps, M.T.: A genetic algorithm for the minimum tollbooth problem (2005)
22. Hearn, D.W., Ramana, M.V.: Solving congestion toll pricing models. In: Equilibrium and Advanced Transportation Modelling, pp. 109–124. Springer (1998)
23. Marx, D.: W[1]-hardness. Recent advances in parameterized complexity (2017)
24. Meyers, C.A., Schulz, A.S.: The complexity of congestion games. Massachusetts Institute of Technology, Cambridge: 1–16 (2008)
25. Orda, A., Rom, R., Shimkin, N.: Competitive routing in multiuser communication networks. IEEE/ACM Transactions on Networking (ToN) **1**(5), 510–521 (1993)
26. Pigou, A.C.: The Economics of Welfare. McMillan&Co., London (1920)
27. Rosenthal, R.W.: A class of games possessing pure-strategy nash equilibria. Int. J. Game Theory **2**(1), 65–67 (1973)
28. Roughgarden, T., Tardos, É.: How bad is selfish routing? J. ACM (JACM) **49**(2), 236–259 (2002)
29. Stefanello, F., Buriol, L.S., Hirsch, M.J., Pardalos, P.M., Querido, T., Resende, M.G., Ritt, M.: On the minimization of traffic congestion in road networks with tolls. Ann. Oper. Res. **249**(1-2), 119–139 (2015)
30. Swamy, C.: The effectiveness of stackelberg strategies and tolls for network congestion games. In: Proceedings of the Eighteenth Annual ACM-SIAM symposium on Discrete Algorithms, pp 1133–1142. Society for Industrial and Applied Mathematics (2007)