Inapproximability of Pure Nash Equilibria*

Alexander Skopalik, Berthold Vöcking
Department of Computer Science
RWTH Aachen, Germany
{skopalik,voecking}@cs.rwth-aachen.de

Abstract

Pure-strategy Nash equilibria are a natural and convincing solution concept for multiplayer games with the *finite improvement property*, i.e., any sequence of improvement steps by individual players is finite and any maximal such sequence terminates in a Nash equilibrium. By far the most literature about games with this property is focussed on *congestion games* that deal with resource allocation by selfish agents. The huge interest in congestion games is not only due to their numerous applications but also because congestion games are isomorphic to *potential games*, i.e., those games in which the finite improvement property is guaranteed by an exact potential function.

The complexity of computing pure Nash equilibria in congestion games was recently shown to be PLS-complete. In this paper, we therefore study the complexity of computing approximate equilibria in congestion games. An α-approximate equilibrium, for $\alpha > 1$, is a state of the game in which none of the players can make an α*-greedy step*, i.e., an unilateral strategy change that decreases the player's cost by a factor of at least α.

Our main result shows that finding an α-approximate equilibrium of a given congestion game is PLS-complete, for any polynomial-time computable $\alpha > 1$. Our analysis is based on a gap introcucing PLS-reduction from FLIP, i.e., the problem of finding a local optimum of a function encoded by an arbitrary circuit. As this reduction is "tight" it additionally implies that computing an α-approximate equilibrium reachable from a given initial state by a sequence of α-greedy steps is PSPACE-complete. Our results are in sharp contrast to a recent result showing that every local search problem in PLS admits a fully polynomial time approximation scheme and, to our knowledge, they are the first inapproximability results with respect to pure Nash equilibria for any kind of games.

In addition, we show that there exist congestion games with states such that any sequence of α-greedy steps leading from one of these states to an α-approximate Nash equilibrium has exponential length, even if the delay functions satisfy a bounded-jump condition. This result shows that a recent result about polynomial time convergence for α-greedy steps in congestion games satisfying the bounded-jump condition is restricted to symmetric games only.

Keywords: congestion games, local search, approximation

*Supported in part by the EU within the 6th Framework Programme under contract 001907 (DELIS) and the German Israeli Foundation (GIF) under contract 877/05.

1 Introduction

Congestion games are an established game theoretic model for resource sharing among selfish players. In such a game, there are n players that share a set of m resources. A strategy of a player corresponds to the selection of a subset of the resources. The delay (cost, payoff) for each player from selecting a particular resource depends on the number of players choosing that resource, and a player's total delay is the sum of the delays associated with the selected resources. Congestion games are of interest to us not only because of their obvious connection to resource sharing in networks and distributed systems with selfish users but also because of their unique game theoric properties.

Rosenthal [**?**] defined the class of congestion games and proved that every game in this class possesses a (pure-strategy) Nash equilibrium. His proof uses a potential function argument that does not only show the existence of Nash equilibria, but it also shows that congestion games have the *finite improvement property*, i.e., any sequence of improvement steps by individual players is finite and any maximal such sequence terminates in a Nash equilibrium. Rosenthal's potential function assigns a value to every state of the game. It is exact in the sense that the delay of a player making an improvement step decreases by the same amount as the potential. Monderer and Shapley [**?**] showed that congestion games are isomorphic to so-called potential games, that is, they are (up to an isomorphism) the only class of games with an exact potential function.

The existence of a potential function suggests to interprete the problem of finding a pure equilibrium as a local search problem in which the neighborhood is defined by improvement steps of individual players. The problem of computing a Nash equilibrium belongs to the complexity class PLS (Polynomial Local Search) containing also other well studied problems like, e.g., MaxSat or MaxCut with the FLIP-Neighborhood, or TSP with the k-OPT neighborhood. A recent result of Fabrikant et al. [**?**] shows that the problem of computing a pure Nash equilibrium in congestion games is PLS-complete. Moreover, their analysis implies that there are initial states of congestion games from which it takes an exponential number of improvement steps to reach a pure Nash equilibrium regardless of the order in which the improvement steps are made.

Nash equilibria in congestion games can be found in pseudopolynomial time by performing improvement steps in arbitrary order because the potential in every state is upper- and lower-bounded by the sum of the input numbers, i.e., the n possible delays for the m resources. This might raise some hope that one can find approximate equilibria in polynomial time. In fact, Orlin et al. [**?**] showed that every problem in PLS admits a fully polynomial time approximation scheme in the following sense: For any given $\epsilon > 0$, there is an algorithm that finds a solution S such that the neighborhood of S does not contain a solution S^* with an objective value better than S by a factor of at least $1 + \epsilon$, and the running time of this algorithm is polynomially bounded in the input length and $1/\epsilon$. The approach of Orlin et al. [**?**] can also be applied to congestion games for finding a state approximating a local optimum with respect to Rosenthal's potential function. However, this is not a reasonable notion of approximate equilibria because individual players do not care about the potential, which is meaningless for them, but only about their actual delays.

A natural and convincing notion of approximation for equilibria (see, e.g., [**?**, **?**, **?**, **?**]) assumes that agents are ambivalent between strategies whose delays differ by less than a factor of α, for some $\alpha > 1$. Correspondingly, an *α-approximate equilibrium* is a state of the game in which none of the players can unilaterally improve by at least a factor of α. Rosenthal's potential function shows that such an equilibrium exists as it can be found by performing *α-greedy steps*, i.e., improvement steps of individual players that decrease the player's delay by at least a factor of α.

In this paper, we show that despite the fact that there exists a fully polynomial time approximation scheme with respect to the potential, it is PLS-hard to compute an α-approximate equilibrium, for

any polynomial time computable $\alpha > 1$. To our knowledge this is the first hardness result about the approximability of pure Nash equilibria. Our analysis is based on a technically involved gap introducing PLS-reduction from FLIP, i.e., circuit evaluation with the FLIP-neighborhood. As this reduction is "tight" it also implies that computing an α-approximate equilibrium that is reachable from a given initial state is PSPACE-complete. Moreover, the reduction implies that there exist families of congestion games such that any sequence of α-greedy steps leading from a given state to an α-approximate equilibrium has exponential length.

A recent result of Chien and Sinclair [**?**] shows a polynomial upper bound on the number of $(1+\epsilon)$-greedy steps, for any $\epsilon > 0$, given that the following two conditions hold. At first, they consider only *symmetric* congestion games, i.e., all players have the same set of strategies. At second, they assume that the delay functions satisfy a *bounded-jump condition*, i.e., the increase in the delay of a resource for any additional player is at most β, for a polynomially bounded parameter β. Both of these conditions are violated by our PLS-reduction. Nevertheless, we can prove that this promising convergence result cannot be extended to asymmetric congestion games. Towards this end, we explicitely construct a family of asymmetric congestion games satisfying the bounded-jump condition such that any sequence of α-greedy steps leading from a given state to an α-approximate equilibrium has exponential length.

1.1 Preliminaries

A *congestion game* Γ is a tuple $(\mathcal{N}, \mathcal{R}, (\Sigma_i)_{i\in\mathcal{N}}, (d_r)_{r\in\mathcal{R}})$ where $\mathcal{N} = \{1, \ldots, n\}$ denotes the set of players, $\mathcal{R} = \{1, \ldots, m\}$ the set of resources, $\Sigma_i \subseteq 2^{\mathcal{R}}$ the strategy space of player i, and $d_r : \mathbb{N} \to \mathbb{Z}$ a delay function associated with resource r. In this paper, we focus on positive and increasing delay functions. We call a congestion game *symmetric* if all players share the same set of strategies, otherwise we call it *asymmetric*. We denote by $S = (S_1, \ldots, S_n)$ the *state of the game* where player i plays strategy $S_i \in \Sigma_i$. For a state S, we define the *congestion* $n_r(S)$ on resource r by $n_r(S) = |\{i \mid r \in S_i\}|$, that is, $n_r(S)$ is the number of players sharing resource r in state S. We call a state S an *α-equilibrium*, for $\alpha > 1$, if no player can decrease its delay by at least a factor of α by unilaterally changing its strategy.

Rosenthal [**?**] shows that every congestion games possesses at least one Nash equilibrium by considering the potential function $\phi : \Sigma_1 \times \cdots \times \Sigma_n \to \mathbb{Z}$ with $\phi(S) = \sum_{r\in\mathcal{R}} \sum_{i=1}^{n_r(S)} d_r(i)$. This potential function does not only prove the existence of pure Nash equilibria but it also shows that such an equilibrium is reached in a natural way when players iteratively play improvement steps as the potential decreases by the same amount as the delay of the player making the improvement step. Because of this property, congestion games can be interpreted as local search problems. The set of Nash equilibria coincides with the set of local optima with respect to ϕ.

In general, a local search problem Π is given by its set of instances $\mathcal{I}_\Pi$. For every instance $I \in \mathcal{I}_\Pi$, we are given a finite set of feasible solutions $\mathcal{F}(I) \subseteq \{0,1\}^*$, an objective function $c : \mathcal{F}(I) \to \mathbb{Z}$, and for every feasible solution $S \in \mathcal{F}(I)$, a neighborhood $\mathcal{N}(S, I) \subseteq \mathcal{F}(I)$. Given an instance I of a local search problem, we seek for a *locally optimal solution* S^*, i. e., a solution which does not have a strictly better neighbor with respect to the objective function c.

A local search problem Π belongs to PLS if the following polynomial time algorithms exist: an algorithm A which computes for every instance I of Π an initial feasible solution $S \in \mathcal{F}(I)$, an algorithm B which computes for every instance I of Π and every feasible solution $S \in \mathcal{F}(I)$ the objective value $c(S)$, and an algorithm C which determines for every instance I of Π and every feasible solution $S \in \mathcal{F}(I)$ whether S is locally optimal or not and finds a better solution in the neighborhood of S in the latter case.

Johnson et al. [**?**] introduce the notion of a PLS-*reduction*. A problem Π_1 in PLS is PLS-reducible

to a problem Π_2 in PLS if there are polynomial-time computable functions f and g such that f maps instances I of Π_1 to instances $f(I)$ of Π_2, g maps pairs (S_2, I) where S_2 denotes a solution of $f(I)$ to solutions S_1 of I, and for all instances I of Π_1, if S_2 is a local optimum of instance $f(I)$, then $g(S_2, I)$ is a local optimum of I.

A local search problem Π in PLS is PLS-*complete* if every problem in PLS is PLS-reducible to Π. Schäffer and Yannakakis [**?**] prove completeness results for various local search problem. In a master reduction they prove the PLS-completeness of the local search problem FLIP. In this problem one searches for a local optimum (w.l.o.g. a local minimum) of a function f_C computed by a circuit C, where the neighborhood is defined with respect to flips of single input bits. In the following, for any vector $a = (a_1, \ldots, a_k)$, let $\mathrm{val}(a)$ denote the *value of* a, i.e., $\mathrm{val}(a) = \sum_{i=1}^{k} a_i 2^{i-1}$.

Definition 1 (FLIP)**.** *An instance of the problem* FLIP *consists of a Boolean circuit C with input bits $x_1, \ldots, x_n$ and output bits $y_1, \ldots, y_m$. A feasible solution is a bit vector x and the objective value is defined as $c(x) = \mathrm{val}(y)$. The neighborhood $N(x)$ of solution x is the set of bit vectors x' that differ from x in one bit. The objective is to find a local minimum.*

1.2 Previous results

Complexity of computing equilibria in congestion games. Fabrikant et al. [**?**] show that computing a Nash equilibrium of a congestion game is PLS-complete. This result holds for asymmetric and symmetric congestion games. They also study so-called *network congestion games*. In these games the resources correspond to the edges of a given directed graph $G = (V, E)$. The strategy set of player i corresponds to the set of paths between a given source node s_i and target node t_i of G. Fabrikant et al. show that finding a Nash equilibrium in asymmetric network congestion games is PLS-complete as well. However, they present a polynomial time algorithm computing Nash equilibria in symmetric network congestion games. Ackermann et al. [**?**] show PLS-completeness results for several further classes of congestion games. For example, they show that network congestion games are PLS-complete even if all delay functions are linear.

Convergence time of improvement steps. The PLS-completeness proofs from [**?**] and [**?**] are based on so-called tight PLS-reductions that preserve the lengths of improvement paths and, hence, imply that the considered classes of games contain instances that have states for which any sequence of improvement steps to a Nash equilibrium has an exponential length. As a positive result, Ackermann et al. [**?**] show that the number of best response improvement steps is polynomially bounded if the strategy space of each player corresponds to a matroid. They also show that this is the maximal condition on the players' strategy spaces that ensures a subexponential upper bound on the convergence time. Furthermore, they show that the convergence time of improvement steps in symmetric network congestion games is exponential, although these games admit a polynomial time algorithm for computing Nash equilibria. Goemans et al. [**?**] show that improvement steps in (weighted) congestion games with polynomial delay functions converge towards a so-called *sink equilibrium*. In addition, they show that a randomly generated sequence of best response improvement steps reaches a set of states whose expected average delay approximates the optimal average delay within a constant factor after a polynomial number of steps. However, their approach does not guarantee to reach an α-approximate equilibrium, for any $\alpha > 0$.

Convergence time to approximate equilibria. To our knowledge, α-approximate equilibria have been introduced by Roughgarden and Tardos [**?**] in the context of routing games with an infinite num-

ber of players. Polynomial upper bounds on the convergence time of distributed re-routing processes (similar to improvement steps) for such routing routing games have been shown in [**?**, **?**, **?**]. Goemans et al. [**?**] investigate the price of anarchy for α-greedy players in congestion games. Ackermann et al. [**?**] give an example for a family of congestion games that admit exponentially long sequences of α-greedy steps. This result leaves open, however, whether pivoting rules like, e.g., *largest improvement first* need only a polynomial number of steps to reach a Nash equilibrium. Recently, Chien and Sinclair [**?**] have shown that $(1+\epsilon)$-greedy players need only a polynomial number of improvement steps to reach an $(1+\epsilon)$-approximate equilibrium in symmetric congestion games whose delay functions satisfy the bounded-jump condition.

2 PLS-hardness of α-approximate equilibria

The following theorem shows that finding an α-approximate Nash equilibrium is PLS-complete, where $\alpha > 1$ might be any constant or, more generally, any number that is polynomial-time computable in the input length of the considered game.

Theorem 1. *Finding an α-approximate equilibrium in a congestion game with positive and increasing delay functions is* PLS*-complete, for every polynomial-time computable $\alpha > 1$.*

Consider any instance of FLIP consisting of a circuit C as described in Definition 1. We describe a transformation of C into a congestion game $G(C)$ such that every Nash equilibrium of $G(C)$ corresponds to a local optimum of f_C. The set of players of $G(C)$ is called $\mathcal{N}$ and the set of resources is called $\mathcal{R}$. The size of both of these sets is polynomially bounded in the number of gates of the circuit. Our construction is a "gap introducing" reduction: It ensures that any Nash equilibrium is an α-approximate equilibrium because the delays of different strategies of any player in $G(C)$ deviate at least by a factor of α. Both the transformation from C to $G(C)$ and the mapping from the equilibria of $G(C)$ to the local optima of f_C are polynomial time computable. Thus, our construction is a a PLS-reduction.

To simplify the presentation of $G(C)$, we use non-negative and non-decreasing rather than positive and increasing delay functions. One can easily obtain positive and increasing delay functions by scaling all delay values by a factor of $2 \cdot |\mathcal{R}| \cdot |\mathcal{N}|$ and then increasing the delay for i players on resource r by an amount of i, for every $r \in \mathcal{R}$ and $i \in \mathcal{N}$. This small modification of the delays does not change the preferences of the players as all delays in $G(C)$ are integral and we make sure that there are no ties in the players' preferences. W.l.o.g. assume $\alpha > 2$. After this modification the delays of different strategies of any player deviate at least by a factor of $\alpha/2$ so that every Nash equilibrium is now an $(\alpha/2)$-approximate equilibrium.

Our description of the game $G(C)$ begins with a short description of an unsuccessful attempt that points out the major problem that we have to solve — the feedback problem.

2.1 A first unsuccessful attempt

From the given circuit C, we construct a circuit C' with n inputs $x_1, \ldots, x_n$ and the same number of outputs $z_1, \ldots, z_n$. The function $f_{C'}$ computed by this circuit is defined as follows. For any given input bit vector x, let $\bar{x}^{(i)}$ denote the vector obtained from x when flipping the i-th bit. For $1 \leq i \leq n$, we define

$$z_i \;=\; \begin{cases} 1 - x_i & \text{if } i \text{ is the smallest index such that } f_C(\bar{x}^{(i)}) \leq f_C(x), \\ x_i & \text{otherwise.} \end{cases}$$

Given the circuit C one can construct the circuit C' in polynomial time.

Now the idea is to represent the circuit C' in form of a congestion game $G(C')$. We use n input players $X_1, \ldots, X_n$ and n *output players* $Z_1, \ldots, Z_n$. Input bit x_i is represented by input player X_i. Each of these players has a zero- and a one-strategy. When X_i plays its zero-strategy then $x_i = 0$ and when it plays its one-strategy then $x_i = 1$. Analogously player Z_i represents the output bit z_i in form of a zero- and a one-strategy. In addition, there is a set of *gate players* who simulate the gates of the circuit C'. The task of these players is to "propagate" the values chosen by the input players from gate to gate in topological order to the output players. This propagation can be achieved by defining the delays in such a way that, for each gate, the delay difference of the gate's input players is larger than the delay difference of the gate's output players. In particular, the propagation ensures that the congestion game $G(C')$ is in an equilibrium if and only if $\mathrm{val}(z) = f_{C'}(x)$.

By the definition of the circuit C', we have the following property: In every equilibrium of $G(C')$, there exists at most one index i such that $z_i \neq x_i$. If there is no such index then $x_1, \ldots, x_n$ correspond to a local optimum of the FLIP-instance C. Now suppose, we can transform $G(C')$ into a congestion game that additionally ensures the property that, in an equilibrium, it holds $z_i = x_i$, for $1 \leq i \leq n$. Given this property, the set of equilibria for the constructed congestion game would correspond to the set of local optima for C and thus our construction would be a PLS-reduction.

Now the problem is that an implementation of the above idea would require to propagate values not only in forward direction from input players to output players (via the gate players) but also in backward direction from output players to input players. It is unclear, however, how to implement such a feedback as the forward propagation requires that the delay differences of the output players are (significantly) smaller than the delay differences of the input players such that the backward propagation seems to be impossible.

2.2 Overview of the congestion game $G(C)$

We now describe our way to solve the feedback problem. The general idea is to construct a congestion game that does not only resemble a single circuit but a "processor" consisting of various "circuits", a "controller" and a "clock". The controller "locks" always one of the circuits. The clock will play a major role in our construction. It is build by a set of players that have the highest delay differences among all players and, hence, can trigger the other players. The clock corresponds to a counter that counts downward (sometimes by more than one step) and solves the feedback problem by triggering the input players depending on which circuit the controller has verified.

A subset of the set of states of the game is called *inexpensive*. The other states are called *expensive*. Let M be a very large number. In particular, we assume that M dominates the delays in any inexpensive state at least by a factor of α. In any expensive state, at least one of the resources has a delay of at least M. We will make sure that the expensive states are transient and any sequence of improvement steps leads to the inexpensive states. Once the subset of inexpensive states is reached, it will not be left again, as there are no improvement steps that lead from inexpensive to expensive states. The set of equilibria is a subset of the inexpensive states.

Let us describe the involved players in more detail. There are n *input players* that represent the input bits of the circuit C. Input bit x_i is represented by input player X_i. Each of these players has a zero- and a one-strategy. When X_i plays its zero-strategy then $x_i = 0$ and when it plays its one-strategy then $x_i = 1$. We will ensure, that in every equilibrium of the game, the values $x_1, \ldots, x_n$ represented by the input players are a local optimum with respect to f_C.

The clock is build on the basis of a set of additional players $Y_1, \ldots, Y_m$ representing m bits $y_1, \ldots, y_m$. Recall that m corresponds to the number of output bits of C. The value of the clock

is defined to be $\text{val}(y) = \sum_{i=1}^{m} y_i 2^{i-1}$. Our simulation of a processor proceeds in "supersteps". The clock counts downwards, that is, in every superstep the value of the clock is decreased at least by one. The major difficulty is to ensure that our simulation of a processor does not terminate because the clock has reached the value zero but only because the input players have reached a local optimum. This is ensured by a so-called "upper bound condition" that we will present later.

The *controller* is represented by a single player. First of all, it ensures that the expensive states are transient. In particular, in an expensive state, the controller "resets the clock" such that the upper bound condition is satisfied. In the inexpensive states the controller has the possibility to "lock a circuit". We continue with a formal description of the lockable circuits before we describe the input and the clock players in more detail.

2.3 Lockable circuits

The congestion game $G(C)$ contains subgames for the following circuits:

- The circuit S_0 checks the *upper bound condition*. The input bits of this circuit are $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$. It computes the function $f_{S_0} : \{0,1\}^{n+m} \to \{0,1\}$ with $f_{S_0}(x, y) = 1$ if $\text{val}(y) \geq f_C(x)$ and $f_{S_0}(x, y) = 0$, otherwise.
- For each $j \in \{1, \ldots, m\}$, $i \in \{1, \ldots, n\}$, $b \in \{0, 1\}$, the circuit $S^j_{i,b}$ has the input bits $x' = x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$ and $y' = y_{j+1}, \ldots, y_m$. It computes the function $f_{S^j_{i,b}} : \{0,1\}^{n-1+m-j} \to \{0,1\}$ with $f_{S^j_{i,b}}(x', y') = 1$ if $\sum_{i=j+1}^{m} y_i 2^{i-1} + 2^j > f_C(x|x_i = b)$ and $f_C(x|x_i = b) < f_C(x|x_i = 1 - b)$; $f_{S^j_{i,b}} = 0$, otherwise.

The reason for defining the circuits in this particular way will become clear in Section 2.4. Next we explain how these circuits are implemented in form of congestion games.

Let $\mathcal{S}$ denote the set of these circuits. We encode every $S \in \mathcal{S}$ into a congestion game $G(S)$. Each gate of a circuit S is simulated by a *gate player*. The sets of gate players and resources for different circuits in $\mathcal{S}$ are disjoint and they are a part of the congestion game $G(C)$. However, different circuits from $\mathcal{S}$ share the same input bits. The bits $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$ correspond to the input and clock players of $G(C)$. Besides the player *controller* participates in each of the circuit games. We say the controller *locks* a circuit $S \in \mathcal{S}$ if it chooses a strategy containing some special *locking resources* of $G(S)$. In every inexpensive state, the controller locks exactly one of the circuits from $\mathcal{S}$.

Lemma 2.

- *In any inexpensive state, the sum of the delays over all resources over all games $G(S)$, for $S \in \mathcal{S}$, is at most $\beta = \alpha^{2K+1}$, where K is the total number of gates over all circuits in $\mathcal{S}$.*
- *In any inexpensive state, the delay differences between the zero- and the one-strategy of any gate player in $G(S)$, for $S \in \mathcal{S}$, is at least α.*
- *A circuit $S \in \mathcal{S}$ with fixed input vector u can be locked by the controller with a delay of less than M if the gate players of $G(S)$ are in equilibrium and $f_S(u) = 1$; otherwise, the delay incurred by locking $G(S)$ is at least M^2.*
- *If the controller locks $S \in \mathcal{S}$ in an inexpensive state then any gate, input and clock player participating in $G(S)$ is* locked *to zero or one, i.e., changing the strategy of the player increases the delay of the player to at least M^2.*

The description of the construction that yields this lemma is shifted to Appendix A.

1. The controller switches from locking S_0 to locking $S^j_{i,b}$.
2. By moving from its one- to its strategy change$^j_{i,b}$, player Y_j
   - triggers player X_i to switch to bit b, and
   - triggers the players $Y_1, \ldots, Y_{j-1}$ to switch to bit 1.
3. After all triggered actions are done, player Y_j can change to its strategy check$^j_{i,b}$, and thereby it triggers the controller to lock S_0.
4. The controller moves back to locking S_0.
5. Player Y_j changes to its zero-strategy.

Figure 1: Description of a *superstep* beginning and ending in a base state.

2.4 Interaction of players and circuits

In this section, we describe how to combine the lockable circuits and the controller with the input players $X_1, \ldots, X_n$ and the clock players $Y_1, \ldots, Y_m$ to simulate sequences of improvement steps of the FLIP-instance by sequences of so-called "supersteps" of the congestion game.

The input players $X_1, \ldots, X_n$ participate in the lockable circuits as described in the previous section. Additionally, the zero-strategy of player X_i contains the resource Trigger$X_{i,1}$, and the one-strategy contains the resource Trigger$X_{i,0}$. Each trigger resource has a delay of 0 when allocated by one player and a delay of $\alpha\beta$ if allocated by two or more players, where the term β (cf. Lemma 2) dominates all delays incurred in inexpensive states in the lockable circuits. We say that a player P *triggers* player X_i to use the zero- or the one-strategy if P uses the resources $TriggerX_{i,1}$ or Trigger$X_{i,0}$, respectively. In an inexpensive state, if any player P triggers X_i to use its zero-strategy (one-strategy), then X_i prefers to use its zero-strategy (one-strategy).

The role of the clock players $Y_1, \ldots, Y_m$ is to trigger the actions of the input players and the controller. Intuitively, they resemble a clock that counts downwards. Let $\gamma = 2\alpha\beta$ be an upper bound on the maximum delay of any input or gate player in inexpensive states. We use γ^j as a factor in the delays of the resources dedicated to clock player Y^j. In the inexpensive states, the delay differences of player Y_j is larger than the delay differences of the gate players, the input players, the controller, and the clock players $Y_1, \ldots, Y_{j-1}$. This way, player Y_j can trigger these players to change their strategy. Clock player Y_j has a zero-strategy, a one-strategy, and the strategies change$^j_{i,b}$ and check$^j_{i,b}$, for $1 \leq i \leq n$ and $b \in \{0, 1\}$.

We will specify the strategies, resources and delays of the clock players and the controller in such a way that any improvement sequence can be partitioned into so-called *supersteps* that begin and end in so-called *base states*, i.e., inexpensive states in which every clock player chooses a zero- or one-strategy and the controller locks the circuit S_0. These states satisfy the upper bound condition:

Lemma 3. *In every base state, it holds* $\mathrm{val}(y) \geq f_C(x)$.

Proof. Circuit S_0 can only be locked if its output is 1, that is, if $\mathrm{val}(y) \geq f_C(x)$. □

Figure 1 gives an overview of the sequence of events in a superstep. If the superstep starts in a base state with x being a local optimum of f_C and $val(y) = f_C(x)$ then the controller has no incentive to lock any of the circuits $S^j_{i,b}$ and the process terminates. Otherwise, the five steps (possibly skipping

step 2) are executed one after the other. During this execution, one of the input players (possibly) changes its strategy. The locked circuit ensures that this change decreases the value of f_C. The input player is triggered by clock player Y_j. The value of the clock, val(y), decreases at least by one during the superstep. The strategies change$^j_{i,b}$ and check$^j_{i,b}$ of the clock player Y_j are used to enforce that the five steps are executed in the right order and the process continues with step 1 of the next superstep.

The exact definitions of the strategies and delays for the input players, the clock players, and the controller are given in Appendix B. These definitions imply that sequences of improvement steps lead to the inexpensive states and then proceed superstep by superstep until a Nash equilibrium is reached.

2.5 Proof of correctness

The correctness of our construction is shown by an analysis of the Nash equilibria of $G(C)$ which implicitly reveals the superstep structure of improvement sequences in the inexpensive states.

Lemma 4. *Every Nash equilibrium is a base state.*

In the proof of this lemma, it is shown that the controller resets the clock in the expensive states and, this way, it is guaranteed that there exists an improvement sequence from every expensive state to an inexpensive state. The inexpensive states are partitioned into five sets $Z_0, \ldots, Z_4$ that correspond to the configurations found at the beginning of each of the five steps of a superstep listed in Figure 1. The set Z_0 corresponds to the base states and it is shown that $Z_1, \ldots, Z_4$ do not contain Nash equilibria. The proof of this lemma is shifted to Appendix C.

Lemma 5. *Suppose s is a base state in Nash equilibrium. Then the bit vector x represented by the input players is a local optimum of f_C.*

Proof. Consider any base state s in which x is not a local optimum of f_C. We show that the controller can make an improvement step.

As x is not a local optimum, there are indices $i \in \{1, \ldots, N\}$ and $b \in \{0, 1\}$ with $f_C(x|x_i = b) < f_C(x)$. Lemma 3 implies val(y) $\geq f_C(x) > f_C(x|x_i = b)$. Hence, there is an index $j \in \{1, \ldots, m\}$ such that $f_C(x|x_i = b) \leq \sum_{i=j+1}^{m} y_i 2^{i-1} + 2^j - 1$. Thus, circuit $S^j_{i,b}$ has result 1 and can be locked. In such a situation, the controller can make an improvement step as it has a delay of at least α on strategy LockS_0 and a delay of only 1 when switching to strategy Lock$S^j_{i,b}$. □

As a consequence, every Nash equilibrium of the congestion game $G(C)$ corresponds to a local optimum of the circuit C. This completes the proof of Theorem 1.

3 Tightness of the PLS-reduction and consequences

A closer look at the PLS-reduction in Section 2 shows that this reduction preserves the structure of the state graph of FLIP in the following way: Suppose we merge all base states of $G(C)$ in which the input players $X_1, \ldots, X_n$ are using the same strategies to a *supernode* and add a directed edge from a supernode A to a supernode B if there is a state in A from which we can reach a state in B using only one superstep. The directed graph built by the supernodes and these edges is isomorphic to the state graph of the FLIP-instance C because an edge exists if and only if the input vectors a and b represented by the supernodes A and B, respectively, differ in one bit only and it holds $f_C(b) < f_C(a)$. This implies that our PLS-reduction is *tight* as defined by Schäffer and Yannakakis [**?**]. As a consequence, we can transfer the following results from FLIP.

Corollary 6. *Let $\alpha > 1$. For every $\ell \in \mathbb{N}$, there is a congestion game whose description length is polynomial in ℓ such that this game has at least one state with the property that every sequence of α-greedy steps leading from this state to an α-approximate equilibrium has a length that is exponential in ℓ.*

Corollary 7. *It is* PSPACE*-hard to compute an α-equilibrium that is reachable by α-greedy steps from a given state in a given congestion games, for every polynomial-time computable $\alpha > 1$.*

4 Convergence

In this section, we show that asymmetric congestion games with α-greedy players do not converge quickly to an approximate Nash equilibrium even if the delay functions satisfy the bounded-jump condition.

Theorem 8. *For every $\alpha > 2$, there is a $\beta > 1$ such that, for every $n \in \mathbb{N}$, there is a congestion game $G(n)$ and a state s with the following properties. The description length of $G(n)$ is polynomial in n. The length of every sequence of α-greedy improvement steps leading from s to an α-approximate equilibrium is exponential in n. All delay functions of $G(n)$ satisfy the β-bounded jump condition.*

Proof. We prove the theorem by constructing a congestion game $G(n)$ that resembles a recursive run of n *programs*, i.e., sequences of α-greedy steps. After its activation, program i triggers a run of program $i-1$, waits until it finishes its run, and triggers it a second time. These sequences are deterministic apart from the order in which some auxiliary players make their improvement steps.

A program i is implemented by a gadget G_i consisting of a main player that we call Main_i and eight auxiliary players called $\text{Block}_i^1, \ldots, \text{Block}_i^8$. The main player has nine strategies numbered from 1 to 9. Each auxiliary player has two strategies, a first and a second one. A gadget G_i is *idle* if all of its players play their first strategy. Gadget G_{i+1} *activates* gadget G_i by increasing the delay of the first strategy of player Main_i. In the following sequence of improvement steps the player Main_i successively changes to the strategies $2, \ldots, 8$. We call this sequence a *run* of G_i. During each run, $Main_i$ activates gadget G_{i-1} twice by increasing the delay of the first strategy of Main_{i-1}. Gadget G_{i+1} is blocked (by player Block_i^8) until player Main_i reaches its strategy 9. Then G_{i+1} continues its run, that is, it decreases the delay of the first strategy of player Main_i, waits until gadget G_i becomes idle again, and afterwards triggers a second run of G_i.

The role of the auxiliary players of G_i is to control the strategy changes of Main_i and Main_{i+1}. In particular they ensure the following three properties:

a) Gadget G_i is activated by gadget G_{i+1} only if it is idle.

b) When gadget G_i is activated, it starts a run.

c) Gadget G_{i+1} waits until G_i has finished its run.

In the initial state s, every gadget G_i with $1 \leq i \leq n-1$ is idle. Gadget G_n is activated. In every improvement path starting from s, gadget G_i is activated 2^{n-i} times, which yields the theorem.

Now we go into the details of our construction. Tables defining all strategies and delays can be found in Appendix D.

Without considering the auxiliary players, the delays of the strategies $1, \ldots, 9$ of a main player decrease by a factor of at least α if the gadget is activated. If it is not activated, only the delay of

strategy one differs. In this case, it is by a factor of at least α smaller than the delay of strategy 9. Furthermore, the delays of Main_i are scaled with some factor δ^i. If Main_i plays strategy 3 or 7, it activates gadget G_{i-1} by increasing the delay of strategy 1 of Main_{i-1}. Note, that the delay of Main_i decreases when Main_{i-1} starts its run and changes from its first to its second strategy. However, due to the scaling factor δ^{i-1} this change does not affect the preferences of player Main_i.

The auxiliary players implement a locking mechanism. The first strategy of player Block_i^j is $\{t_i^j, b_i^j\}$ and its second strategy is $\{c_i^j\}$. The delays of the resources b_i^j and c_i^j are relatively small (δ^{i-1} and $2\alpha\delta^{i-1}$, respectively) if allocated by only one player. If they are allocated by two or more players, however, then each of them induce a significantly larger delay of δ^{i+2}. Theses resources are also part of the strategies of Main_i or Main_{i+1}. Note, that neither Main_i nor Main_{i+1} has an incentive to change to a strategy having a delay of δ^{i+1} or more. The delay of the resource t_i^j is chosen such that Block_i^j has an incentive to change to its second strategy if Main_i allocates this resource. If Main_i neither allocates this resource nor the resource b_i^j, it has an incentive to change to its first strategy. Due to scaling factor δ^{i-1} the delays of the resource t_i^j do not affect the preferences of Main_i.

These definitions yield the following properties. If auxiliary player Block_i^j of gadget G_i plays its first strategy then this prevents Main_i from choosing strategy $i+2$. Player Block_i^j has an incentive to change to its second strategy only if player Main_i chooses its strategy $i+1$. By this mechanism, we ensure that Main_i chooses the strategies 1 to 8 in the right order. In addition, the first strategy of Block_i^8 prevents Main_{i+1} from going to strategy 4 or 8. This ensures that Main_{i+1} waits until the run of player Main_i is completed. Furthermore, Main_{i+1} can enter into strategy 3 or 7 only if all auxiliary players of gadget G_i use their first strategy. This ensures that a run starts with all auxiliary players being in their first strategy.

We have shown that in every sequence of improvement steps from s to a Nash equilibrium in the congestion game $G(n)$ each gadget i is activated 2^{n-i} times. One can easily check that every improvement step of a player decreases its delay by a factor of at least α and every delay function satisfies the β-bounded jump condition with $\beta = \delta^3$ with $\delta = 10\alpha^9$. □

5 Conclusion

We have shown the inapproximability of pure Nash equilibria in congestion games with positive and increasing delay functions. Our work already inspired other research to circumvent the general inapproximability by placing restrictions on the delay functions or considering other notions of approximation.

The delay functions that we use in our PLS-reduction have the property that the delay increases by a rather large factor when increasing the number of players from k to $k+1$ for some $k \in \{1, 2, 3\}$. This observation inspired recent research [**?**] about the approximability of equilibria in congestion games with latency functions like polynomials with relatively large positive offset and queueing theoretic functions for queues with a relatively high service rate (in comparison to the equilbrium load). It is investigated which approximation ratios can be obtained by the method of randomized rounding depending on the parameters of the latency functions.

In another recent work [**?**], it is shown that $(1+\epsilon)$-greedy steps converge in polynomial time to a state approximating the social optimum (assuming the delay functions satisfy the bounded jump condition). This result is not in contrast to our negative result about convergence. Combining their result with our construction in Section 4 shows that $(1+\epsilon)$-greedy steps quickly converge to a state with cost close to *social optimum · price of anarchy* but afterwards the progress towards an approximate equilibrium is extremely slow.

References

A Proof of Lemma 2

Every $S \in \mathcal{S}$ can be encoded into a congestion game $G'(S)$ as follows. W.l.o.g., S consists of only NAND gates with fan-in 2. The two inputs of a gate are called a and b, respectively. Let k denote the number of gates in S. Let $g_1, \ldots, g_k$ denote the gates of the circuit in reverse topological order. Gate g_i is associated with a player G_i that has a zero-and a one-strategy. For $1 \leq i \leq k$, the zero-strategy of player G_i contains the resources Bit0a_i and Bit0b_i. Both of these resources have delay 0 when allocated by one player and delay α^{2i} if two or more players are on that resource. The one-strategy of player G_i contains the resource Bit1$_i$ with delay 0 if allocated by at most two players and delay α^{2i} if three or more players are on that resource. Now let $j \in \{1, \ldots, i-1\}$, denote an index of a resource g_j with gate g_i as input. Then the one-strategy of g_i additionally contains Bit1$_j$, and the zero-strategy of g_i additionally contains the resource Bit0a_j if g_i corresponds to input a and the resource Bit0b_j if g_i corresponds to input b. The inputs of the circuit also correspond to players with a zero- and a one-strategy. The strategies of these players contain the bit resources of the gates to which they are connected in the same way as the gate players. For the time being, we assume that each of these players independently of the other circuit players prefers either its zero- or its one-strategy and, hence, these players represent a fixed vector of input bits.

Lemma 9. *Fix any input vector u for circuit S. The delay differences between the zero- and the one-strategy of any gate player in any state of $G'(S)$ is more than α. $G'(S)$ has a unique equilibrium in which the output player uses the strategy $f_S(u)$.*

Proof. The lemma follows by an induction on the gates in topological order. Consider a gate g_i. Let G_a and G_b denote players corresponding to the two inputs of g_i. We distinguish two cases.

Suppose G_a and G_b both use their one-strategy. Then the cost of the one-strategy of G_i is at least α^{2i}, and the cost of the zero strategy is at most $\sum_{j=1}^{i-1} \alpha^{2j} < \alpha^{2i-1}$ because $\alpha \geq 2$. Thus, player G_i has an advantage of more than a factor of α to use the zero-strategy, which corresponds to the semantics of the NAND gate.

Now suppose G_a or G_b use their zero-strategy. Then the cost of the one strategy of G_i is at most $\sum_{j=1}^{i-1} \alpha^{2j} < \alpha^{2i-1}$, and the cost of the zero-strategy is at least α^{2i}. Thus, player G_i has an advantage of more than a factor of α to use the one-strategy, which again corresponds to the semantics of the NAND gate. □

We now add some further resources. These are the resources Lock0a_i, Lock0b_i, and Lock1$_i$, for every $1 \leq i \leq k$. These resources are contained in the same strategies of the gate players as the resources Bit0a_i, Bit0b_i, and Bit1$_i$, respectively. The resources Lock0a_i and Lock0b_i have a delay 0 if they are used by at most two players and a delay of M^2 if used by at least three players. The resource Lock1$_i$ has a delay of 0 if it is used by at most three players and a delay of M^2 if it is used by at least four players. Observe that these resources have always a delay of 0 as long as we assume that they are used only by the gate players as there are at most two gate players containing each of the resources Lock0a_i and Lock0b_i in their strategies, and at most three gate players containing the resource Lock1$_i$. The only other player that is interested in these resources is the controller. For every circuit S, the controller has a distinct strategy that contains all lock resources of $G'(S)$. We say that the game $G'(S)$ is locked if the controller chooses the strategy occupying the lock resources of $G'(S)$.

Lemma 10. *Fix any input vector u for circuit S. The controller can lock the game $G'(S)$ if and only if $G'(S)$ is in its unique equilibrium, unless at least one of the resources has a delay of at least M^2.*

Proof. If $G'(S)$ is in its equilibrium then, for every gate g_i, each of the resources Lock0a_i and Lock0b_i is used by at most one gate player, and the resource Lock1$_i$ is used by at most two gate players. Even if also the controller uses the resources, the delay on all of these resources is 0.

Now assume $G'(S)$ is not in the equilibrium. Then there is a gate g_i violating the NAND-semantics. We distinguish the following two cases:

Suppose both input players of g_i play their one-strategies and player G_i plays the one-strategy as well. Then the resource Lock1$_i$ is used by three gate players. Taking into account the controller that allocates Lock1$_i$ as well, this resource has a delay of M^2.

Suppose one of the input players of g_i plays its zero-strategy and player G_i plays its zero-strategy as well. Then at least one of the two resources Lock0a_i and Lock0b_i is used by two players. Taking into account the controller also allocating Lock0a_i and Lock0b_i, at least one of these two resources has a delay of M^2. □

We now modify the congestion game $G'(S)$ such that it can only be locked for those inputs u with $f_S(u) = 1$. We achieve this property by simply fixing the player that represents the value of the circuit to its one-strategy, that is, we remove the output player's zero-strategy from the game. The congestion game obtained in this way, is called $G(S)$. Observe that the controller cannot lock the game $G(S)$ for any input u with $f_S(u) = 0$ as in this case always one of the gate players has to violate the NAND semantics. This way, we obtain the following lemma.

Lemma 11. *Fix any input vector u for circuit S. The controller can lock the game $G(S)$ if and only if $f_S(u) = 1$ and if $G(S)$ is in its unique equilibrium, unless at least one of the resources has a delay of at least M^2.* □

The game $G(C)$ contains the players and resources $G(S)$, for every $S \in \mathcal{S}$. The sets of gate players and resources over all $G(S)$ are disjoint. However, the circuits share some of their input bits and so the players representing these bits contain resources of different circuits from $\mathcal{S}$. Until now we assumed that these players are fixed to either their zero or one strategy. Now we relax this assumption: Let I be any of the players representing the input values of the circuits in S. At this point, we only assume that in any state of $G(C)$ the delay difference between the zero and the one strategy of I is so large that the choices of all the players in the gates to which I is connected do not affect the preferences of I. W.l.o.g., we assume, in every circuit S for which I represents an input, I is connected to at least one NAND gate in which the other input is fixed to the value 1, i.e., this input is occupied by an auxiliary player having only a one-strategy. Then we have the following property.

Lemma 12. *Consider a circuit S. Assume the controller has locked $G(S)$ in an inexpensive state and player I represents an input of S in its zero- or one-strategy. Then the other, zero- or one-strategy of I incurs a delay of at least M^2.*

Proof. Let gate g_i be the gate with input a represented by player I and input b represented by a player fixed to its one-strategy. If player I changes from the zero- to the one-strategy then the resource Lock1$_i$ is used by three gate players and the controller such that its delay is M^2. If player I changes from the one- to the zero-strategy then the resource Lock0a_i is used by two gate players and the controller such that its delay is M^2 as well. □

This completes the proof of the properties summarized in Lemma 2.

B Details of the clock players, the input players, and the controller

We define the delay function of the resources by specifying the delay values in form of a list, e.g. $0/1/M$ corresponds to delay of 0 for one player, delay of 1 for two and delay of M for three or more players. Recall that $\beta = \alpha^{2K+1}$ with K being the total number of gates over all circuits and $\gamma = 2\alpha\beta$. That is, $\alpha \ll \beta \ll \gamma \ll M$.

Strategies of Y_j	Resources	Delays
One	One_j	$4\alpha^4\gamma^j$
	Bit1_k of $G(S)$ (if gate g_k of circuit S has y_j as input)	$0/0/\alpha^{2k}$
	Lock1_k of $G(S)$ (if gate g_k of circuit S has y_j as input)	$0/0/0/M^2$
$\text{change}^j_{i,b}$	Change_j	$3\alpha^3\gamma^j$
(for all $i \in \{1,\ldots,n\}$	$\text{Trigger}Y_{j'}$ (for all $1 \le j' \le j$)	$0/5\alpha^5\gamma^{j'}$
and $b \in \{0,1\}$)	$\text{Trigger}X_{i,b}$	$0/\alpha\beta$
	$\text{Block}S_0$	$0/M^2$
	$\text{Block}S^{j'}_{i',b'}$ (for all vectors (i',j',b') except (i,j,b) with $i' \in \{1,\ldots,n\}$, $j' \in \{1,\ldots,m\}$ and $b' \in \{0,1\}$)	$0/M^2$
	Bit1_k of $G(S)$ (if gate g_k of circuit S has y_j as input)	$0/0/\alpha^{2k}$
	Lock1_k of $G(S)$ (if gate g_k of circuit S has y_j as input)	$0/0/0/M^2$
$\text{check}^j_{i,b}$	Check_j	$2\alpha^2\gamma^j$
(for all $i \in \{1,\ldots,n\}$	$\text{Trigger}\text{Done}Y_{j'}$(for all $1 \le j' < j$)	$0/M^3$
and $b \in \{0,1\}$)	$\text{Block}X_{i,1-b}$	$0/M^3$
	$\text{Trigger}Y_j$	$0/5\alpha^5\gamma^j$
	$\text{Block}S^{j'}_{i',b'}$ (for all vectors (i',j',b') except (i,j,b) with $i' \in \{1,\ldots,n\}$, $j' \in \{1,\ldots,m\}$ and $b' \in \{0,1\}$)	$0/M^2$
	TriggerController	$1/\alpha^2$
	$\text{Bit0}a_k$ of $G(S)$ (if gate g_k of circuit S has y_j as input a)	$0/\alpha^{2k}$
	$\text{Bit0}b_k$ of $G(S)$ (if gate g_k of circuit S has y_j as input b)	$0/\alpha^{2k}$
	$\text{Lock0}a_k$ of $G(S)$ (if gate g_k of circuit S has y_j as input a)	$0/0/M^2$
	$\text{Lock0}b_k$ of $G(S)$ (if gate g_k of circuit S has y_j as input b)	$0/0/M^2$
Zero	$\text{Trigger}Y_j$	$0/5\alpha^5\gamma^j$
	$\text{Trigger}\text{Done}Y_j$	$0/M^3$
	$\text{Block}Y_j$	$0/M^2$
	$\text{Bit0}a_k$ of $G(S)$ (if gate g_k of circuit S has y_j as input a)	$0/\alpha^{2k}$
	$\text{Bit0}b_k$ of $G(S)$ (if gate g_k of circuit S has y_j as input b)	$0/\alpha^{2k}$
	$\text{Lock0}a_k$ of $G(S)$ (if gate g_k of circuit S has y_j as input a)	$0/0/M^2$
	$\text{Lock0}b_k$ of $G(S)$ (if gate g_k of circuit S has y_j as input b)	$0/0/M^2$

Figure 2: Definition of the strategies of the players Y_j with $1 \le j \le m$.

Strategies of X_i	Resources	Delays
One	Trigger$X_{i,0}$	$0/\alpha\beta$
	Block$X_{i,1}$	$0/M^3$
	Bit1$_k$ of $G(S)$ (if gate g_k of circuit S has x_i as input)	$0/0/\alpha^{2k}$
	Lock1$_k$ of $G(S)$ (if gate g_k of circuit S has x_i as input)	$0/0/0/M^2$
Zero	Trigger$X_{i,1}$	$0/\alpha\beta$
	Block$X_{i,0}$	$0/M^3$
	Bit0a_k of $G(S)$ (if gate g_k of circuit S has x_i as input a)	$0/\alpha^{2k}$
	Bit0b_k of $G(S)$ (if gate g_k of circuit S has x_i as input b)	$0/\alpha^{2k}$
	Lock0a_k of $G(S)$ (if gate g_k of circuit S has x_i as input a)	$0/0/M^2$
	Lock0b_k of $G(S)$ (if gate g_k of circuit S has x_i as input b)	$0/0/M^2$

Figure 3: Definition of the strategies of the players X_i with $1 \leq i \leq n$.

Strategies of the controller	Resources	Delays
LockS_0	Lock$_0$	α
	BlockS_0	$0/M^2$
	Lock1$_k$ of $G(S_0)$ for all gates g_k of S_0	$0/0/0/M^2$
	Lock0a_k of $G(S_0)$ for all gates g_k of S_0	$0/0/M^2$
	Lock0b_k of $G(S_0)$ for all gates g_k of S_0	$0/0/M^2$
Lock$S^j_{i,b}$	TriggerController	$1/\alpha^2$
	Block$S^j_{i,b}$	$0/M^2$
	BlockY_j	$0/M^2$
	Lock1$_k$ of $G(S^j_{i,b})$ for all gates g_k of $S^j_{i,b}$	$0/0/0/M^2$
	Lock0a_k of $G(S^j_{i,b})$ for all gates g_k of $S^j_{i,b}$	$0/0/M^2$
	Lock0b_k of $G(S^j_{i,b})$ for all gates g_k of $S^j_{i,b}$	$0/0/M^2$
reset	Reset	M
	TriggerY_j (for all $j \in \{1, \ldots, m\}$)	$0/5\alpha^5\gamma^j$

Figure 4: Definition of the strategies of the controller

C Proof of Lemma 4

We prove this lemma by dividing the set of states in several disjoint sets $Z_0, \ldots, Z_6$, where the set Z_0 is the set of base states. We show that the sets $Z_1, \ldots, Z_6$ contain no Nash equilibrium.

First, we study the inexpensive states, i.e., the states in which all players have a delay of less than M. These states are partitioned into the sets $Z_0, \ldots, Z_4$ as follows.

- Z_0 contains the base states, i.e., the states in which every clock player Y_j plays a zero- or one-strategy, and the controller plays LockS_0.
- Z_1 contains the states in which the controller plays Lock$S^j_{i,b}$, for some $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$, $b \in \{0, 1\}$, and all clock players $Y_{j'}$ play a zero- or one-strategy.
- Z_2 contains the states in which the controller plays Lock$S^j_{i,b}$, for some $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$, $b \in \{0, 1\}$, and player Y_j plays strategy change$^j_{i,b}$. Every player $Y_{j'}$ with $j' \neq j$, plays its zero- or one-strategy.

- Z_3 contains the states in which the controller plays Lock$S^j_{i,b}$, for some $i \in \{1,\dots,n\}$, $j \in \{1,\dots,m\}$, $b \in \{0,1\}$, and player Y_j plays check$^j_{i,b}$. All players $Y_{j'}$ with $1 \le j' < j$ play their one-strategy. Every player $Y_{j''}$ with $j < j'' \le m$ plays its one or zero-strategy. Player X_i chooses its one or zero-strategy if b is 1 or 0, respectively.

- Z_4 contains all states in which the controller plays strategy LockS_0, and, for some $i \in \{1,\dots,n\}$, $j \in \{1,\dots,m\}$, $b \in \{0,1\}$, player Y_j plays strategy check$^j_{i,b}$. Every player $Y_{j'}$ with $j' \neq j$ plays its one or zero-strategy.

Lemma 13. *The set of inexpensive states is equal to* $Z_0 \dot\cup \dots \dot\cup Z_4$.

Proof. The five sets are obviously disjoint. We need to show that every state is either contained in one of these sets or there is a player with a delay of at least M. Fix any state s. We distinguish the following three cases according to the strategy selected by the controller.

- Suppose the controller plays LockS_0. If all Y_j play their zero- or one-strategy, s is contained in Z_0. If there is a $j \in \{1,\dots,m\}$ with player Y_j playing a change strategy, Y_j has delay of more then M^2 due to the resource BlockS_0 that is allocated by the controller. If there is exactly one player Y_j on a check strategy, then the state is in Z_4. Assume two players Y_j and $Y_{j'}$ are on a check strategy. Let W.l.o.g. $j < j'$. Then both players use the resource TriggerDoneY_j with a delay of M^2.

- Suppose the controller plays Lock$S^j_{i,b}$. If all clock players are on their zero- or one-strategy, s is contained in Z_1. If player Y_j is on a change or check strategy and no other clock player is on a change or check strategy, s is contained in Z_2 or Z_3, respectively. If additionally a player $Y_{j'}$ with $j' \neq j$ is on a change or check strategy, then this player has a delay of at least M^2 due to the resource Block$S^j_{i,b}$.

- If the controller plays *reset* then its delay is at least M. □

Now we prove that none of the states in $Z_1,\dots,Z_4$ is a Nash equilibrium by considering these sets one after the other and showing that in each of these cases there is a player that can decrease its delay by changing its strategy.

Consider any state $s \in Z_1$. Let Lock$S^j_{i,b}$ be the strategy chosen by the controller. Observe that player Y_j cannot be on its zero-strategy because, otherwise, it and the controller would have a delay of at least M^2 due to the resource BlockY_j. Thus, Y_j is on its one-strategy and has delay of at least $4\alpha^4\gamma^j$ so that Y_j can decrease its delay to at most $4\alpha^3\gamma^j$ by switching to the strategy change$^j_{i,b}$.

Consider any state $s \in Z_2$. Let Lock$S^j_{i,b}$ be the strategy chosen by the controller. Assume s is a Nash equilibrium. This implies that every player $Y_{j'}$ with $j' < j$ plays its one-strategy which delay is by a factor of at least α smaller than the delay of all other strategies because player Y_j allocates the resources Trigger$Y_1,\dots,$Trigger$Y_{j'}$. Furthermore, as Y_j allocates resource Trigger$X_{i,b}$, player X_i plays its one-strategy if $b = 1$ and its zero-strategy if $b = 0$. Now observe that player Y_j can decrease its delay by changing to strategy check$^j_{i,b}$ which, under these conditions, has a delay of at most $3\alpha^2\gamma^j$.

Consider any state $s \in Z_3$. Let Lock$S^j_{i,b}$ be the strategy chosen by the controller. Assume s is a Nash equilibrium. By Lemma 11 circuit $S^j_{i,b}$ outputs a 1 and by the definition of the circuits, circuit S_0 also outputs a 1 in an equilibrium state under the conditions in Z_3. Hence, the controller has an incentive to lock S_0 since strategy LockS_0 has a delay of α whereas strategy Lock$S^j_{i,b}$ has a delay of α^2.

Consider any state $s \in Z_4$. In this case, there is a player Y_j using his check strategy with a delay of at least $2\alpha^2\gamma^j$. Y_j has an incentive to switch to its zero-strategy having a delay of at most $2\alpha\gamma^j$.

It remains to investigate the expensive states, i.e., the states in which at least one player has a delay of M or larger. We partition these states into the sets Z_5 and Z_6.

- Z_5 is the set of states with a player having delay greater or equal M^2.
- Z_6 is the set of states with the controller playing the reset strategy and no other player having a delay greater than or equal to M^2.

Lemma 14. *The set of expensive states is equal to* $Z_5 \dot\cup Z_6$.

Proof. The lemma holds as the controller is the only strategy with a delay of at least M and less than M^2. □

We have to show that these sets do not contain a Nash equilibrium.

In every state $s \in Z_5$, there is at least one player having a delay of at least M^2. If the controller has delay of at least M^2, then changing to its reset strategy decreases its delay at least by a factor of α. Now we show that any gate, input or clock player with a delay of at least M^2 can either improve its delay or the controller has a delay of M^2 as well. If an input player has a delay of at least M^3 due to a blocking resource, then it can always decrease its delay by a strategy change. If it has a delay of at least M^2 but less than M^3, the controller has a delay of M^2, too. If a clock player has a delay of at least M^3, then it can decrease its delay by changing to its zero-strategy. If it has delay of at least M^2 but less than M^3, the controller has delay M^2, too. If a gate player has a delay of at least M^2, then the controller has a delay of at least M^2 as well.

Finally, assume there is a state $s \in Z_6$ that is a Nash equilibrium. In s every player Y_j plays its one-strategy since the delay of all other strategies is by a factor of at least α larger than its zero-strategy. In this case, the upper bound condition obviously holds and, thus, the result of circuit S_0 is 1. Consequently, the controller can decrease its delay by changing to LockS_0 which is a contradiction to the assumption.

This completes the proof of Lemma 4.

D Details of the players Main_i and Block_i^j

The congestion game $G(n)$ consists of the gadgets $G_1, \ldots, G_n$. Each gadget G_i consists of a player Main_i and the players $\mathrm{Block}_i^1, \ldots, \mathrm{Block}_i^8$. The nine strategies of a player Main_i are given in Figure 5. The two strategies of a player Block_i^j are given in Figure 6. $\delta = 10\alpha^9$ is a scaling factor for the delay functions.

Strategy	Resources	Delays
(1)	e_i^1	$\delta^i/9\alpha^9\delta^i$
(2)	e_i^2	$8\alpha^8\delta^i$
	$c_{i-1}^1, \ldots, c_{i-1}^9$	$2\alpha\delta^{i-2}/\delta^{i+1}$
	t_i^1	$\delta^{i-1}/2\alpha^2\delta^{i-1}$
(3)	e_i^3	$7\alpha^7\delta^i$
	e_{i-1}^1	$\delta^{i-1}/9\alpha^9\delta^{i-1}$
	t_i^2	$\delta^{i-1}/2\alpha^2\delta^{i-1}$
	b_i^1	$\delta^{i-1}/\delta^{i+2}$
(4)	e_i^4	$6\alpha^6\delta^i$
	b_{i-1}^8	$\delta^{i-2}/\delta^{i+1}$
	t_i^3	$\delta^{i-1}/2\alpha^2\delta^{i-1}$
	b_i^2	$\delta^{i-1}/\delta^{i+2}$
(5)	e_i^5	$5\alpha^5\delta^i$
	t_i^4	$\delta^{i-1}/2\alpha^2\delta^{i-1}$
	b_i^3	$\delta^{i-1}/\delta^{i+2}$

Strategy	Resources	Delays
(6)	e_i^6	$4\alpha^4\delta^i$
	$c_{i-1}^1, \ldots, c_{i-1}^9$	$2\alpha\delta^{i-2}/\delta^{i+1}$
	t_i^5	$\delta^{i-1}/2\alpha^2\delta^{i-1}$
	b_i^4	$\delta^{i-1}/\delta^{i+2}$
(7)	e_i^7	$3\alpha^3\delta^i$
	e_{i-1}^1	$\delta^{i-1}/9\alpha^9\delta^{i-1}$
	t_i^6	$\delta^{i-1}/2\alpha^2\delta^{i-1}$
	b_i^5	$\delta^{i-1}/\delta^{i+2}$
(8)	e_i^8	$2\alpha^2\delta^i$
	b_{i-1}^8	$\alpha\delta^{i-2}/\delta^{i+1}$
	t_i^7	$\delta^{i-1}/2\alpha^2\delta^{i-1}$
	b_i^6	$\delta^{i-1}/\delta^{i+2}$
(9)	e^9	$\alpha\delta^i$
	t_i^8	$\delta^{i-1}/2\alpha^2\delta^{i-1}$
	b_i^7	$\delta^{i-1}/\delta^{i+2}$

Figure 5: Definition of the strategies of the players Main_i. The delay of resource e_n^1 is constantly $9\alpha^9\delta^n$.

Strategies of Block_i^j	Resources	Delays
(1)	t_i^j	$\delta^{i-1}/2\alpha^2\delta^{i-1}$
	b_i^j	$\delta^{i-1}/\delta^{i+2}$
(2)	c_i^1	$2\alpha\delta^{i-1}/\delta^{i+2}$

Figure 6: Definition of the strategies of the players Block_i^j