Routing Games over Time with FIFO policy

Anisse Ismaili

Ito Takayuki Laboratory, Nagoya Institute of Technology, Japan
anisse.ismaili@gmail.com

Abstract. We study atomic routing games where every agent travels both along its decided edges and through time. The agents arriving on an edge are first lined up in a *first-in-first-out* queue and may wait: an edge is associated with a capacity, which defines how many agents-per-time-step can pop from the queue's head and enter the edge, to transit for a fixed delay. We show that the best-response optimization problem is not approximable, and that deciding the existence of a Nash equilibrium is complete for the second level of the polynomial hierarchy. Then, we drop the rationality assumption, introduce a behavioral concept based on GPS navigation, and study its worst-case efficiency ratio to coordination.

Keywords: Routing Games over Time, Complexity, Price of Anarchy

1 Introduction

Numerous selfish agents use a routing network to take shortest paths that may however congest the paths of others. *Routing games* model such conflictual systems by a graph of vertices and edges, and every agent decides a path from a source to a sink, path which cost is congestion-dependent. Routing games find applications in road traffic [War52], as well as in routing packets of data via Internet Protocol [KP99]. Founding results[(a)] have been obtained on *static* routing games [RT02, Rou05, CK05, AAE05, NRTV07, Rou09], where each individual path instantaneously occurs everywhere over its decided edges. Such instantaneousness does not reflect that an agent on one edge of its path is not elsewhere, and cannot congest other edges. Routing games *over time*, where every agent travels along its route as well as *through time*, were introduced more recently [KS09, AU09]. Introducing time makes games more complicated: pure-strategy Nash equilibria are often not guaranteed; problems such as computing a best-response or an equilibrium are hard; the Price of Anarchy (PoA) can be large.

We study asymmetric atomic routing games over integer time-steps that model congestion with a very natural *first-in-first-out* (FIFO) queuing policy on the edges [WHK14]. Every edge e has an integer *fixed delay* d_e and an integer *capacity* c_e. On an edge, every arriving agent is lined up in the edge's FIFO queue (a discrete list); the capacity defines how many agents-per-time-step can pop from the queue's head and transit through the edge, while the others wait the next time-step. Every agent aims at minimizing the time from source to sink.

[(a)] Static routing games were a crucial testbed for the Price of Anarchy, a concept that bounds a game's loss of efficiency due to selfish behaviors.

Related Work. Only pure Nash equilibria (PNE) are considered here. It is the same (resp. different) source/sink in the symmetric (resp. asymmetric) case.

[HHP06, HHP09] studies multicommodity routing problems, where asymmetric commodities are routed sequentially and the cost of edges is load dependent. With affine costs, while in the splittable case the PoA is almost 4, in the unsplittable case, computing a best-response is NP-hard, and the PoA is $3 + 2\sqrt{2}$.

[FOV08] observes that "*a car traversing a road can only cause congestion delays to those cars that use the road at a later time*" and proposes an asymmetric model where every edge has a priority on agents, agents that are congested only by those with a higher priority on the edge. While a global priority (same fixed priority for every edge) guarantees the existence of a PNE, local priorities do not. Several (matching) bounds are derived on the PoA.

[KS09, KS11] introduces competitive flows over time, by building a non-atomic symmetric model upon the literature about deterministic queuing. Every edge has a fixed transit delay, and a capacity that bounds above the edge's outflow. It is shown that a sequence of ε-Nash flows converges (as $\varepsilon \to 0$) to a Nash flow; and an iterative algorithm is proposed. While the evacuation-PoA can be arbitrarily large, the time-PoA is in $O(1)$.

[AU09] proposes a dynamic selfish routing model with non-atomic asymmetric agents. A very general delay function $d_e(x, H_e^t)$ of the demand x and the historic H_e^t is introduced, along with a generalized notion of FIFO, which just states that there are no crossovers. Concurrently to [KS09], it is shown that in the symmetric case, a PNE always exists and can be computed efficiently. However, in the asymmetric case and under a specification of FIFO where an entering agent waits the previous one's end of transit, it is shown that an equilibrium may not exist, and the PoA is bounded below by the number of vertices. Flow independent delays can be reduced to static flows, providing a PoA bound.

[HMRT09, HMRT11] proposes temporal (asymmetric and atomic) network congestion games. Every edge has a speed $a_e \in \mathbb{R}_{>0}$ (latency equals speed times weight of agents being processed), and different local policies are studied. Under FIFO, an edge processes a unitary agent in time a_e, while the other agents wait. A guaranteed PNE can be computed efficiently for the unweighted symmetric case, despite the NP-hardness of computing a best-response. In the weighted or asymmetric cases, an equilibrium may not exist. One could reduce one of our edges e to $c_e \times d_e$ speedy-edges having $a_e = 1$, but it is *pseudo*-polynomial. Conversely, it is also unclear how we could reduce this model to the present one.

Our model is the same as in [WHK14], an atomic variation over integer time-steps of [KS09], where every edge has a free-flow delay and a capacity that bounds above the inflow-per-time-step. In [WHK14], the emphasis is rather on *bottleneck* individual objectives, but also on the *sum* on the edges in the path. A PNE may not exist; Computing a best-response is NP-complete; Verifying a PNE is coNP-complete; Deciding PNE existence is at least NP-hard. Also, a bound is provided on the PoA. [WBK15] studies games where agents are robust bottleneck optimizers that only know an interval about the cost of edges and learn the actual cost later.

[HPSVK16] studies a model similar to [WHK14] and ours, but instead of FIFO, studies local and overall priorities on the edges. (Crossovers may occur.) Some bounds on the price of stability and PoA are derived. Computing optimal priority lists is shown APX-hard. Under local priorities, computing best-responses is NP-hard, as well as computing a PNE.

Furthermore, [MLS10, MLS13] generalizes Braess's Paradox to the model in [KS09], and Braess's ratio can be much more severe. [BFA15] considers a model in the fashion of [KS09] and shows that under a Stackelberg strategy, the time-PoA is $(1-1/e)^{-1}$, and the total-delay-PoA is $2(1-1/e)^{-1}$. [CCCW17a, CCCW17b] propose an extensive form model where agents take new decisions on each vertex.

Results. In this paper's model [WHK14, *sum* objective], the *new* results are marked here with a *star**:

Th.1 A pure-strategy Nash equilibrium may not exist [(b)].
Th.2 The payoffs are well defined and calculable in polynomial-time [(c)].
Th.3 * The best-response decision problem is NP-complete [(d)] [(e)] [(f)].
Th.4 * The best-response optimization problem is APX-hard,
Th.5 * and it is NP-hard to approximate within $|V|^{\frac{1}{6}-\varepsilon}$, and within $n^{\frac{1}{7}-\varepsilon}$.
Th.6 Verification of equilibria is coNP-complete[(g)].
Th.7 * Existence of equilibria is Σ_2^P-complete[(h)].

That best-responses are not approximable, deeply questions the rationality assumption of PNE. We then introduce a behavioral model for vehicles taking decisions by GPS, inspired by how navigation assistants work: by retrieving information on the current traffic and recomputing shortest paths in real-time. On the worst-case efficiency ratio of GPS navigation, to coordination, we found:

Th.8 * Allowing walks[(i)] as strategies, GPS-agents may cycle infinitely.
Th.9 * The Price of GPS Navigation is in $\Omega(|V|+n)$ as the number of vertices $|V|$ and the number of agents n grow.

Model Discussion. The positioning of waiting queues on the edges' tails, and of fixed-delays inside edges, is without much loss of generality. Indeed, this choice reduces in polynomial time from/to models where the queue occurs after the fixed delay, where queues are on the nodes and fixed delays on the edges, where edges are unoriented, etc. Furthermore, one can model starting times by

[(b)] [WHK14, App. B.1, Fig. B.11] contains a similar result. For culture and the pleasure of the eyes, we include a close didactic counter-example in our Preliminaries.
[(c)] A close Dijkstra-style algorithm for local priorities lies in [HPSVK16, Prop2.2].
[(d)] Theorem 3 differs from [HMRT09, Sec. 3.1] where edges instead only have a speed.
[(e)] Not addressed in [WHK14, Appendix B], but for *bottleneck* [WHK14, Cor. 4].
[(f)] Our reduction from problem MinVertexCover is the base of Th. 4, 6 and 7.
[(g)] Theorem 6 differs from [WHK14, Cor. 4], where the objectives are bottlenecks. [WHK14, Sec. 7] claims that one can derive NP-*hardness* for sum-objectives. Theorems 2 and 3 enable to obtain coNP-*completeness*, which is then partly novel.
[(h)] As a Σ_2^P-*completeness* result, Th. 7 clearly improves on the NP-*hardness* shown in [WHK14, App. B.3, Th. 19].
[(i)] A walk is an alternating sequence of vertices and edges, consistent with the given (di)graph, and that allows repetitions and infiniteness.

adding edges, and bottlenecks by delay $d_e = 0$ edges. One can also note that on each edge e, delay $\lfloor (|q_e| - 1)/c_e \rfloor + d_e$ is almost an affine function of congestion $|q_e|$ (the queue's length). Since the unweighted agents case that we consider is a particular case of the weighted case (and Algorithm 1 adapts), our complexity results and efficiency lower bounds extend to weighted agents.

2 Preliminaries

Definition 1. *A First-in-first-out Routing Game (*Frog*) is a non-cooperative finite game characterized by tuple* $\Gamma = (G = (V, E), (c_e, d_e)_{e \in E}, N, (s_i, s_i^*)_{i \in N}, \succ)$.

- $G = (V, E)$ *is a finite digraph with vertices* V *and edges* $E \subseteq V \times V$.
- *Given edge* $e \in E$*, positive number* $c_e \in \mathbb{N}_{\geq 1}$ *is the capacity of edge* e*, and non-negative number* $d_e \in \mathbb{N}_{\geq 0}$ *is the fixed delay on edge* e.
- *Finite set* $N = \{1, \ldots, n\}$ *is the set of agents.*
- *Given agent* $i \in N$*, vertices* $s_i, s_i^* \in V$ *are its source vertex and sink vertex.*
- *Strict order* $\succ$ *on set* N *is a tie-breaking priority on agents.*

For a given Frog, we introduce the following notations. For every agent i, its *strategy-set* $\mathcal{P}_i$ consists of every simple path π_i from source vertex s_i to sink vertex s_i^*. A *strategy-profile* $(\pi_1, \ldots, \pi_n) \in \mathcal{P}_1 \times \cdots \times \mathcal{P}_n$, which for short we denote in bold by $\boldsymbol{\pi} \in \boldsymbol{\mathcal{P}}$, defines a strategy for every agent. For a given strategy-profile $\boldsymbol{\pi} \in \boldsymbol{\mathcal{P}}$; strategy π_i is the strategy of agent i therein (a simple path from s_i to s_i^*); *adversary strategy-profile* $\boldsymbol{\pi}_{-i} \in \prod_{j \neq i} \mathcal{P}_j$ consists of all strategies in $\boldsymbol{\pi}$ but agent i's; and given strategy π_i', strategy-profile $(\pi_i', \boldsymbol{\pi}_{-i}) \in \boldsymbol{\mathcal{P}}$ is obtained from strategy-profile $\boldsymbol{\pi}$ by changing strategy π_i into π_i'.

Agents travel both along edges and through *time*. For an agent i, *total delay* $C_i : \boldsymbol{\mathcal{P}} \to \mathbb{N}_{\geq 0}$ is a function of the strategy-profile, defined as follows. As depicted in Fig. 1, when agent i arrives on edge $e \in \pi_i$, it lines up in a first-in-first-out (FIFO) queue specific to edge e. At each time-step, edge e lets the c_e first agents in the queue enter the edge to transit for d_e time steps. Let function $w_{i,e} : \boldsymbol{\mathcal{P}} \to \mathbb{N}_{\geq 0}$ be the time spent waiting by agent i in the queue of edge e. It follows that agent i's total delay is defined by equality

$$C_i(\boldsymbol{\pi}) \quad = \quad \sum_{e \in \pi_i} w_{i,e}(\boldsymbol{\pi}) + d_e.$$

If, on one edge, some agents arrive at the same exact time step, then these synchronous agents are ordered in the edge's queue by tie-breaking priority $\succ$. Section 3 shows how to compute, given a strategy profile, the total delays.

A rational agent, given an adversary strategy-profile, individually optimizes its total delay. This rationality assumption induces standard concepts:

Definition 2. *Given agent* i *and adversary strategy-profile* $\boldsymbol{\pi}_{-i}$*, strategy* π_i *is a best-response if and only if:* $C_i(\pi_i, \boldsymbol{\pi}_{-i}) \quad = \quad \min_{\pi_i' \in \mathcal{P}_i} \{C_i(\pi_i', \boldsymbol{\pi}_{-i})\}$.

Definition 3. *A pure Nash equilibrium (PNE) is a strategy-profile* $\boldsymbol{\pi} \in \boldsymbol{\mathcal{P}}$ *where*

$$\forall i \in N, \quad \forall \pi_i' \in \mathcal{P}_i, \quad C_i(\boldsymbol{\pi}) \leq C_i(\pi_i', \boldsymbol{\pi}_{-i}).$$

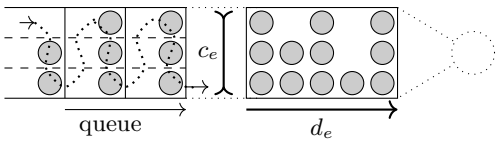


Fig. 1. On a given edge e, agents (gray rounds) are first lined up in a FIFO queue. The edge lets the c_e first agents enter at each time-step, to travel for d_e time steps.

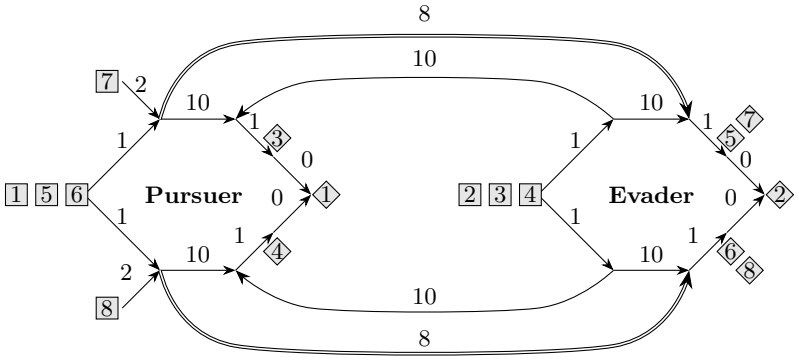


Fig. 2. An example of a FROG where there is no PNE. Single edges have capacity one. Double edges have capacity two. The fixed delays of edges are the numbers displayed above. The sources (resp. sinks) of agents are indicated by squares (resp. diamonds). Choose any tie breaking priority compatible with $2 \succ 3 \sim 4 \succ 1 \succ 5 \sim 6 \succ 7 \sim 8$. Agent one *Pursuer* and two *Evader* have two strategies; the others have only one.

In plain words, strategy-profile $\boldsymbol{\pi} \in \boldsymbol{\mathcal{P}}$ is a PNE if no agent has an individual incentive to deviate from his current strategy, hence if every agent plays a best-response. To illustrate the definitions above we recall (Fig. 2) a didactic variation of a known counter-example [WHK14, Fig. B.11], which implies Theorem 1.

Theorem 1. *In a* FROG*, there may not exist any PNE [WHK14].*

Proof (Theorem 1). Recall that in a pursuer-evader game, the two agents have two corresponding strategies; the pursuer prefers to decide the same; the evader prefers to decide differently; and consequently there is no PNE. In Fig. 2, all agents are degenerate[(j)], but agent one *Pursuer* and agent two *Evader*, who can decide between two paths: *up* or *down*. Agents three and four transmit from Evader to Pursuer a positive externality for deciding the same, and agents five to eight, from Pursuer to Evader, a negative externality.

If both agents decide the same, without loss of generality path *up*, then Evader makes agent three wait one time-step, who in turn arrives one step after Pursuer where they could have collided; hence the total delay of Pursuer is 12. Also, Pursuer makes agent five arrive at time step 10 instead of 9 on the possible

[(j)] A degenerate agent only has one strategy, but can still incur and cause externalities.

collision point with Evader. Moreover, agent seven also arrives there at time 10. Consequently, this queue is congested by five and seven and Evader waits one time-step. So Evader's total delay is 13. Similarly, one can show that when they decide different strategies, Pursuer's total delay is 13, and Evader's is 12. To conclude, Fig. 2 is a pursuer-evader game, and so has no PNE. □

Definition 4. *We study this sequence of computational problems.*

- FROG/DELAYS: *Given a* FROG Γ *and a strategy-profile* $\boldsymbol{\pi}$*, compute the total delays* $(C_1(\boldsymbol{\pi}), \ldots, C_n(\boldsymbol{\pi}))$ *of every agent.*
- FROG/BR/OPT: *Given a* FROG Γ*, an agent* i*, and an adversary strategy-profile* $\boldsymbol{\pi}_{-i}$*, compute a best-response* π_i *for agent* i*.*
- FROG/BR/DEC: *Decision version of* FROG/BR/OPT*. Given a* FROG Γ*, an agent* i*, an adversary strategy-profile* $\boldsymbol{\pi}_{-i}$*, and an integer threshold* $\kappa \in \mathbb{N}_{\geq 0}$*, decide whether there exists a strategy* π_i *with cost* $C_i(\pi_i, \boldsymbol{\pi}_{-i}) \leq \kappa$*.*
- FROG/NE/VERIF: *Given a* FROG Γ *and a strategy-profile* $\boldsymbol{\pi}$*, decide whether strategy-profile* $\boldsymbol{\pi}$ *is a PNE.*
- FROG/NE/EXIST: *Given a* FROG*, decide whether it admits a PNE.*

The representation size of FROGs is a polynomial of numbers $|V|$ and n. We assume that the following concepts are common knowledge: decision problem, length function, complexity classes P, ZPP, NP, coNP, Σ_2^P, Π_2^P, PH, NPO and APX, polynomial-time reduction, L-reduction, hardness and completeness.

3 Mapping Strategy-Profiles to Total-Delays

Strikingly, the mapping from strategy-profiles to payoffs is not well defined under local priorities, when there are directed cycles of length zero [HPSVK16]. Under FIFO priorities with a tie-breaking order, we show the following.

Theorem 2. *The mapping from strategy-profiles to total-delays is well defined, and there is* $^{(c)}$ *a polynomial-time algorithm to compute it:* FROG/DELAYS $\in$ P.

Proof. This proof relies on Alg. 1 that is made deterministic by order $\succ$. Algorithm 1 sequentially performs events, Dijkstra-like, along time. Type θ_{queue} events are when the agent is lined-up at the queue's tail. Type θ_{pop} events are when the agent pops from the queue's head and starts the fixed transit delay on edge e. An *event* is a quadruplet (t, θ, i, e) that occurs on edge $e \in E$ when agent $i \in N$ performs an event of type $\theta \in \{\theta_{\text{queue}}, \theta_{\text{pop}}\}$ at time $t \in \mathbb{N}_{\geq 0}$. As Alg. 1 iterates, time goes forward. Heap $\mathcal{E}$ is the set of next events for the agents that did not finish their path. The events in heap $\mathcal{E}$ are ordered according to time t (lowest first), then type θ (θ_{queue} before θ_{pop}), and then agent i's priority (high priority first). Then, the overall next event is the top of the heap $\textbf{top}(\mathcal{E})$, and can be removed by $\textbf{pop}(\mathcal{E})$. For every edge e, queue q_e is maintained as agents line-up using $\textbf{push-back}(q_e, i)$ and pop using $\textbf{pop}(q_e)$. For every path π_i, we can access

Algorithm 1 Algorithm for mapping strategy-profiles to total-delays.

Input: FROG Γ, strategy-profile $\boldsymbol{\pi}$. **Output:** Total delays $C_1(\boldsymbol{\pi}), \ldots, C_n(\boldsymbol{\pi})$.
Variables: Heap of events $\mathcal{E}$, queues q_e on every edge e, integer results $C_1, \ldots, C_n$.

1: $\mathcal{E} \leftarrow \bigcup_{i \in N} \{(0, \theta_{\text{queue}}, i, \textbf{first}(\pi_i))\}$
2: **while** $\mathcal{E} \neq \emptyset$ **do**
3: $\quad (t, \theta, i, e) \leftarrow \textbf{top}(\mathcal{E})$ **and** **pop**$(\mathcal{E})$
4: $\quad$ **if** $\theta = \theta_{\text{queue}}$ **then**
5: $\quad\quad$ **push-back**(q_e, i)
6: $\quad\quad \mathcal{E} \xleftarrow{\text{insert}} \left(t + \lfloor \frac{|q_e|-1}{c_e} \rfloor, \theta_{\text{pop}}, i, e\right)$
7: $\quad$ **else if** $\theta = \theta_{\text{pop}}$ **then**
8: $\quad\quad$ **pop**(q_e)
9: $\quad\quad$ **if** $e = \textbf{last}(\pi_i)$ **then**
10: $\quad\quad\quad C_i \leftarrow t + d_e$
11: $\quad\quad$ **else**
12: $\quad\quad\quad \mathcal{E} \xleftarrow{\text{insert}} (t + d_e, \theta_{\text{queue}}, i, \textbf{next}(\pi_i, e))$
13: $\quad\quad$ **end if**
14: $\quad$ **end if**
15: **end while**
16: **return** $C_1, \ldots, C_n$.

first edge $\textbf{first}(\pi_i)$, last edge $\textbf{last}(\pi_i)$, and given an edge $e \in \pi_i, e \neq \textbf{last}(\pi_i)$, we can access the next edge $\textbf{next}(\pi_i, e)$.

A queuing event $(t, \theta_{\text{queue}}, i, e)$ develops into the popping event where agent i leaves queue q_e to start the fixed-delay. Crucially, when this queuing event is performed, we know that no further agents can be lined-up prior to agent i in queue q_e, since it's the heap's top, which optimizes time and priority. Also, type θ_{queue} goes before type θ_{pop}, so that current congestion is counted. Hence, we know that agent i will spend time $w_{i,e}(\boldsymbol{\pi}) = \lfloor \frac{|q_e|-1}{c_e} \rfloor$ in the queue. A θ_{pop} event generates either the queuing event after delay d_e, or total delay C_i.

There are at most $2|V|$ events per-agent. Moreover $|\mathcal{E}| \leq n$. Consequently, Alg. 1 is in polynomial-time $O(n|V| \log_2(n))$. It was tested in detail under C++. It adapts to weighted agents by calculation of the weighted length of queues. □

4 Inapproximability of Best-Responses

Theorem 2 implies that problem FROG/BR/OPT is somewhere inside class NPO, and problem FROG/BR/DEC in class NP. In this section, we show that computing a best-response is hard, and provide two inapproximability results.

Theorem 3. *Decision problem* FROG/BR/DEC *is NP-complete.*

So, a polynomial-time algorithm addressing FROG/BR/DEC is unlikely to exist.

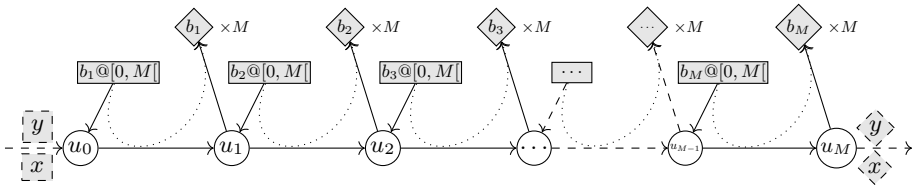


Fig. 3. *Filter: Agent y is not delayed, but agent x is delayed by at least $M \in \mathbb{N}_{\geq 1}$ steps.* Circles, rectangles and diamonds are resp. vertices, sources and sinks. The idea is that agent x waits one step on every edge (u_i, u_{i+1}), but y does not wait. After symbol @ is the source's starting time. An agent b_i starts for every $i \in [1, M]$ and $t \in [0, M[$, hence the number of agents is in $\Theta(M^2)$. The edges have capacity one and fixed-delay zero. Priority $\succ$ satisfies $y \succ b_i \succ x$, for any i (and time) defined in the Figure.

Theorem 4. *Optimization problem* Frog/Br/Opt *is APX-hard.*

Hence, a PTAS for Frog/Br/Opt would imply a PTAS for every NPO problem that admits a poly.-time constant factor approx. algorithm, which is unlikely.

Theorem 5. *For any $\varepsilon \in \mathbb{R}_{>0}$, approximating problem* Frog/Br/Opt *within factor $|V|^{\frac{1}{6}-\varepsilon}$, and within factor $n^{\frac{1}{7}-\varepsilon}$, are NP-hard.*

In plain words, it would take an intractable amount of time for an agent to find a path within factor $|V|^{\frac{1}{7}}$ or $n^{\frac{1}{8}}$ of the shortest delay. A more realistic model may rather drop rationality, and be better based on agents using heuristics.

Before the proofs, a good rule of thumb to distinguish between easy and hard path problems, is whether Bellman's Principle of Optimality is satisfied, or if preference inversions violate the principle[(k)]. We introduce a gadget game[(l)].

Definition 5. *An (M, t)-Backfire is a piece of* Frog*, defined as in Fig. 4.*

Lemma 1. *In an (M, t)-Backfire, if agent x arrives on the t-trigger at time t, then on the bomb, M agents arrive at time $t+1$, and massively delay agent x. Otherwise, if x arrives at a different time, then this Backfire does not delay x. Furthermore, the backfire contains $\Theta(M)$ vertices and $\Theta(M^2)$ agents.*

Proof (Lemma 1). If agent x does not trigger on time t, then agent r_1 makes every agent b_i wait one step. Hence, every agent b_i collides on u_i at $t+2$ with M agents m_i who have priority. Agents b_i finally arrive on w_3 at time $t+1+M$, way too late to delay anyone (assuming large M). If agent x triggers on time t, then he gets queued after agent r_0, and agent r_1 has to wait one step. Then agent r_1 arrives too late on vertices u_i to delay any agent b_i. Consequently, agents b_i arrive on u_i one step before m_i, don't get delayed, and arrive on w_3 at $t+1$. □

(k) This assertion is only a rule of thumb, since no state-space is actually precised.

(l) A busy reader can, after a quick look to Fig. 4, jump straight to Lem. 2.

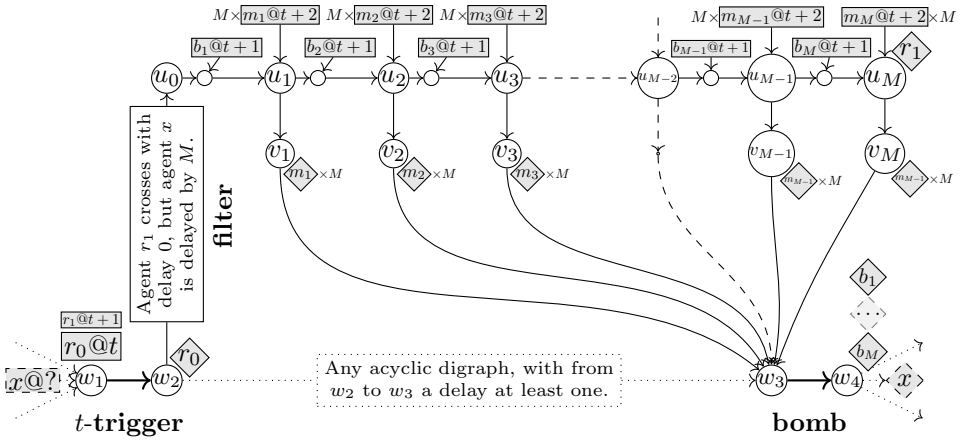


Fig. 4. An (M, t)-Backfire, where $M \in \mathbb{N}_{\geq 1}$ is some large number and $t \in \mathbb{N}_{\geq 0}$ is a time-step. Circles, rectangles and diamonds are resp. vertices, sources and sinks. After symbol @ is the source's starting time. The edges that are plainly depicted (or dashed) have capacity one and fixed-delay zero. The filter is depicted in Fig. 3. Let priority $\succ$ satisfy $r_i \succ m_j \succ b_k \succ x$, for any i, j, k defined in the figure. One can connect to any digraph from the trigger to the bomb, if the minimum delay from w_2 to w_3 is one. *Agent x gets heavily delayed on the bomb if and only if he uses the trigger on time t.*

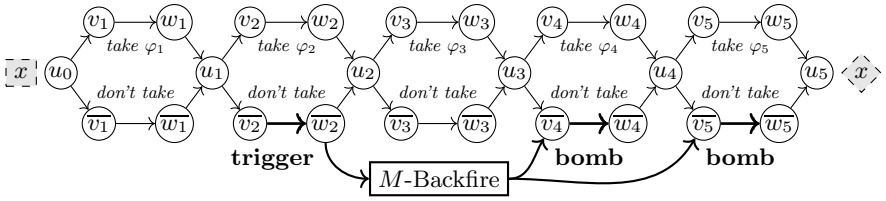
Definition 6. *An M-Backfire is a sequence of (M, t)-Backfires, for $0 \leq t \leq M$, that share the same trigger-edge and bomb-edge. Agent r_0 is removed everywhere but for $t = 0$, because for $t \geq 1$, its role in the (M, t)-Backfire is played by r_1 from the $(M, t-1)$-Backfire. (There is one r_1 per-time-step in $[0, M]$.)*

Lemma 2. *With an M-Backfire, agent x is massively delayed on the bomb-edge (assuming large M), if and only if he crosses the trigger-edge (anytime in $[0, M]$). Furthermore, the backfire contains $\Theta(M^2)$ vertices and $\Theta(M^3)$ agents.*

Proof (Lemma 2). Assume that agent x triggers on time t; all subsequent agents r_1 get delayed by one: an (M, t')-Backfire gets triggered for every $t' \geq t$. □

Proof (Theorem 3). Membership in class NP follows from Th. 2. We show NP-hardness by starting the reduction from *decision* problem MinVertexCover [Kar72, GJ79] that asks, given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and threshold $\kappa \in \mathbb{N}_{\geq 0}$, whether there is a subset $\mathcal{W} \subseteq \mathcal{V}$ such that $\forall \{\varphi_1, \varphi_2\} \in \mathcal{E},\ \varphi_1 \in \mathcal{W}$ or $\varphi_2 \in \mathcal{W}$, and $|\mathcal{W}| \leq \kappa$. Recall that problem MinVertexCover is NP-complete even for degrees bounded above by three [GJS74], which we assume here. We build the Frog depicted in Fig. 5. The reduction's validity is by construction (see the figure's caption). Taking $M = 6\eta$ is sufficient. Since there are $\Theta(\eta)$ edges in $\mathcal{V}$, the reduction makes $\Theta(\eta^3)$ vertices and $\Theta(\eta^4)$ agents, which is polynomial. □

Proof (Theorem 4). Starting from the *optimization* version of problem MinVertexCover where one must find $\mathcal{W}^* \in \arg\min_{\mathcal{W} \subseteq \mathcal{V}} \{|\mathcal{W}| \mid \forall \epsilon \in \mathcal{E},\ \mathcal{W} \cap \epsilon \neq \emptyset\}$



"Edges $\{\varphi_2, \varphi_4\}$ and $\{\varphi_2, \varphi_5\}$ shall be covered, or a backfire will heavily delay agent x."

Fig. 5. From MinVertexCover (degrees bounded above by 3) to Frog/Br/Dec. Circles, squares and diamonds depict respectively vertices, sources and sinks for Frog. Let $\eta = |\mathcal{V}|$ and observe that the starting size is in $\Theta(\eta)$. In the depiction, $\eta = 5$. The idea is a correspondence between $\mathcal{W} \in 2^{\mathcal{V}}$ and path π_x decided by agent x: *taking edge* (v_i, w_i) *in path* π_x *amounts to take vertex* φ_i *in subset* $\mathcal{W}$. Every edge is associated with $(c_e, d_e) = (1, 1)$, but edges (v_i, w_i) with $(1, 2)$, and edges $(\overline{v_i}, \overline{w_i})$ when it's a trigger with $(1, 0)$ (because agent r_1 already makes x wait one step). Consequently going up always takes two steps, and going down one step if it's not a backfired edge.. Hence a vertex cover $\mathcal{W}$ of size k corresponds to a path π_x with length $3\eta + k$. So, threshold κ in MinVertexCover is reduced to $\kappa' = 3\eta + k$ in Frog/Br/Dec. For every edge $\{\varphi_i, \varphi_j\}$ $(i < j)$ in MinVertexCover, we introduce an M-Backfire with trigger $(\overline{v_i}, \overline{w_i})$ and bomb $(\overline{v_j}, \overline{w_j})$, in order to heavily punish x for not taking φ_i *and* φ_j. The backfire splits the provided punishement between up to three neighbors.

(forget about κ and κ'), the same reduction as for Th. 3 is also an L-reduction[(m)] [PY91, Cre97], which we show by exhibiting functions f, g and constants α, β.

Recall that *optimization* problem MinVertexCover is APX-complete even for degrees bounded above by three [PY91, AK97], which we still assume. The correspondences f and g are depicted in the caption of Fig. 5. Given a MinVertexCover instance $\mathcal{I}$, one has $\mathrm{OPT}_{\mathrm{Frog}}(f(\mathcal{I})) \leq \alpha \mathrm{OPT}_{\mathrm{VC}}(\mathcal{I})$ for $\alpha = 10$ and $|\mathcal{W}| \leq \beta C_i(\pi_x)$ where $\mathcal{W} = g(\mathcal{I}, \pi_x)$ for $\beta = 1$. Indeed, for the former, observe that a vertex can cover at most three edges; hence $\frac{\eta}{3} \leq \mathrm{OPT}_{\mathrm{VC}}(\mathcal{I})$. Correspondence $3\eta + \mathrm{OPT}_{\mathrm{VC}} = \mathrm{OPT}_{\mathrm{Frog}}$ then yields $\alpha = 10$. The later comes from $k \leq 3\eta + k$. Consequently, this is an L-reduction and then, optimization problem Frog/Br/Opt is APX-hard. □

Proof (Sketch, Th. 5). Problem MinColoring, given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, asks a coloring of $\mathcal{G}$, i.e. a partition of $\mathcal{V}$ into disjoint sets $V_1, V_2, \ldots, V_k$ such that each V_i is an independent set of $\mathcal{G}$ (no edges in $\mathcal{G}[V_i]$), with *minimum chromatic number* $k = \chi(\mathcal{G})$. Let $\eta = |\mathcal{V}|$. It is known that whatever $\varepsilon > 0$, approximating $\chi(\mathcal{G})$ within $\eta^{1-\varepsilon}$ is NP-hard [FK96, Zuc06]. The idea of the reduction is in Fig. 6, and (with $M = 3\eta$) involves $\Theta(\eta^6)$ vertices and $\Theta(\eta^7)$ agents. Consequently, better approximation ratios than $|V|^{\frac{1}{6}-\varepsilon}$ or $n^{\frac{1}{7}-\varepsilon}$ contradict intractable ratio $\eta^{1-\varepsilon}$ from [FK96]. □

[(m)] An L-reduction is a poly.-time reduction in NPO, which conserves approximations.

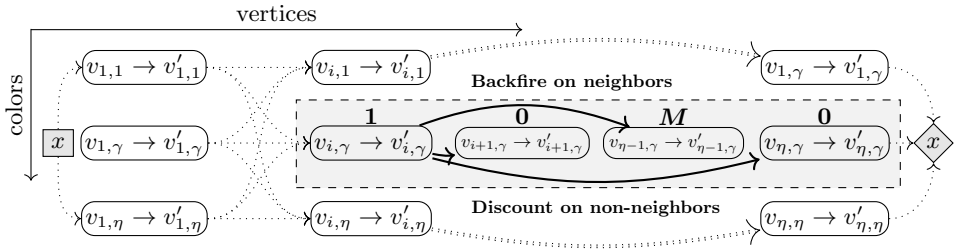


Fig. 6. Sketch of reduction from MinColoring to Frog/Br/Opt. Agent x's paths correspond to deciding a color γ for each vertex i. Depicted in the gray rectangle, the first time x chooses color γ is on vertex i (first time on the line). Then (1) x waits one step on edge $(v_{i,\gamma}, v'_{i,\gamma})$ (because of an agent r_1). Then on the same line, (2) neighbors in $\mathcal{G}$ are Backfired (can't put the same color on a neighbor) and non-neighbors are discounted to delay zero (by heavily delaying agents r_1 and disarming their eventual backfires). Transit edges (dotted) have delay one to allow for backfires to work. Hence, a valid coloring of size k would correspond to a path π_x of length $\eta + k$ which does not enable to find β for an L-reduction. Therefore, to bring the correspondence back from affine to linear, we technically multiply all the costs by M, with M times more agents and vertices, but not on the dotted edges.

5 The Complexity of Pure Nash Equilibria

In this section, we first observe that the verification problem Frog/NE/Verif is coNP-complete. Then, we completely characterize the complexity of the existence problem as *complete* for the second level of $\mathrm{PH}^{(\mathrm{n})}$.

Theorem 6. *Problem* Frog/NE/Verif *is coNP-complete [WHK14, Almost]*$^{(g)}$.

Proof (Theorem 6). A deviation is a no-certificate verifiable in polynomial-time by Alg. 1, hence this problem is inside class coNP. A proof with bottleneck objectives lies in [WHK14, Cor. 4], and the authors claim [WHK14, Sec. 7] that one can obtain NP-hardness for sum-objectives in the same way. We confirm that claim since the same reduction as for Th. 3 holds here. □

Theorem 7. *Problem* Frog/NE/Exist *is* Σ_2^P*-complete.*

Proof (Theorem 7). This problem is in class Σ_2^P. Indeed, yes-instances admit a certificate verifiable by an NP-oracle: by guessing the right strategy-profile, according to Th. 6, one can use an NP-oracle to verify that it is a PNE. The Σ_2^P-hardness proof below generalizes the reduction introduced for Th. 3.

In Fig. 7, we reduce decision problem MaxMinVertexCover to the complement of Frog/NE/Exist. Given set of indices I, the vertices of graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ partition into $\mathcal{V} = \bigcup_{i \in I} \mathcal{V}_{i,0} \cup \mathcal{V}_{i,1}$. Given function $\theta : I \to \{0, 1\}$ (i.e. $2^{|I|}$

(n) Class Σ_2^P are the problems that nest a coNP problem inside an NP problem. Only very small sizes ($\lessapprox 10$) of such problems can usually be practically addressed.

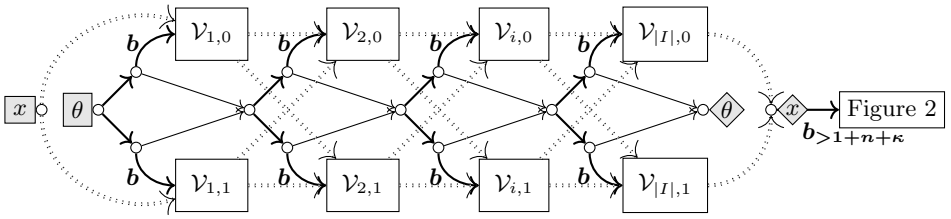


Fig. 7. Reduction from MaxMinVertexCover to the complement of Frog/NE/Exist. Generalizes the reduction in Fig. 5. Each box $\mathcal{V}_{i,j}$ is made as in Fig. 5. We create a universally indifferent agent $\overline{\theta}$ who can early decide between two paths for every $i \in I$, : one backfires $\mathcal{V}_{i,0}$'s entry and the other backfires $\mathcal{V}_{i,1}$'s entry. Agent $\overline{\theta}$ models function θ by blocking early the entries to $\mathcal{V}_{i,0}$ xor to $\mathcal{V}_{i,1}$ with the backfires $\boldsymbol{b}$. Plain edges have capacity and cost $(c_e, d_e) = (1, 0)$. Dotted edges have $(c_e, d_e) = (1, 1)$ but the two first ones $(1, 2)$, to let $\overline{\theta}$ run in front of x. Then agent x decides a path through what corresponds to subgraph $\mathcal{G}^{(\theta)}$. Agent x sends backfires to Fig. 2 if and only if he reaches his sink after time $1 + n + \kappa$.

possibilities), let $\mathcal{G}^{(\theta)}$ denote the graph restricted to vertices $\mathcal{V}^{(\theta)} = \bigcup_{i \in I} \mathcal{V}_{i,t(i)}$. Problem MaxMinVertexCover, given threshold $\kappa \in \mathbb{N}_{\geq} 0$, asks whether:

$$\forall \theta : I \to \{0, 1\}, \quad \exists \mathcal{W} \subseteq \mathcal{V}^{(\theta)}, \quad \mathcal{W} \text{ vertex-covers } \mathcal{G}^{(\theta)} \text{ and } |\mathcal{W}| \leq \kappa, \qquad (1)$$

and is Π_2^P-complete (i.e. co-Σ_2^P-complete); co-Frog/NE/Exist asks whether:

$$\forall \boldsymbol{\pi} \in \mathcal{P}, \quad \text{There exists an individual deviation from } \boldsymbol{\pi}. \qquad (2)$$

[Eq. (1) $\Rightarrow$ Eq. (2)] Whatever the choices of agent $\overline{\theta}$, if the strategy of agent x costs more than $C_i > 1 + n + \kappa$, then he can deviate and improve, because of Eq. (1); otherwise, now assuming that x's strategy is a best-response, then he reaches his sink before time $1 + n + k$ (because Eq. (1)) and does not disable the example from Fig. 2, which remains unstable: there is a deviation.

[not Eq. (1) $\Rightarrow$ not Eq. (2)] If there exists a function θ, then we position agent $\overline{\theta}$ as such. Then the best-response of agent x makes him reach his sink after time $1 + n + \kappa$. Consequently, Fig. 2 is disabled: we have a PNE. □

6 The Price of GPS

Previous sections show how strong an assumption rationality is. Instead, we propose a model inspired by GPS personal navigation assistants: agents retrieve instantaneous traffic data to recompute shortest paths at each crossroad.

We introduce a *GPS-agent* as an agent who at each vertex (between two time steps) recalculates a shortest path according to the fixed delays d_e plus congestion $\lfloor \frac{|q_e|}{c_e} \rfloor$ of the past step. In place of PNE, let $\mathcal{O} \subseteq \mathcal{P}$ be the set of strategy-profiles that can be obtained by GPS-agents. We study the worst-case ratio to coordination, defined for one Frog as the *Price-of-GPS (navigation)*:

$$\text{PoGPS} = \frac{\max_{\boldsymbol{\pi}' \in \mathcal{O}} \{\ C(\boldsymbol{\pi}')\ \}}{\min_{\boldsymbol{\pi} \in \mathcal{P}} \{\ C(\boldsymbol{\pi})\ \}},$$

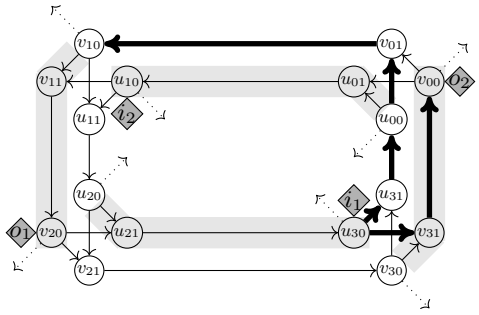


Fig. 8. *Double cycle of infinite procrastination.* The idea is that there is an inner-cycle and an equivalent outer-cycle. Agents from a cycle have to go through the other cycle to reach their sink, but the information that they get from the other cycles does not discourage procrastination. Circles are nodes. Every edge e has capacity $c_e = 1$. The four edges in every corner have fixed-delay $d_e = 0$, and the two from each corner to the next one, fixed-delay $d_e = 1$. There are two *inner-agents* i_1 and i_2, with resp. sources u_{11} and u_{31}, and a sink reachable instantly by the dotted edges from the outer cycle's vertices $v_{00}, v_{10}, v_{20}, v_{30}$. However, they can decide to stay on the inner-cycle $u_{0-}, u_{1-}, u_{2-}, u_{3-}$. There are two *outer-agents* o_1 and o_2, with resp. sources v_{21} and v_{01}, and sinks reachable by the dotted edges from inner vertices $u_{00}, u_{10}, u_{20}, u_{30}$. However, they can decide to cycle on the outer-cycle $v_{0-}, v_{1-}, v_{2-}, v_{3-}$. On the figure, we show w.l.o.g. current positions of the agents and the congestion from the last step in gray rectangles. The current choice faced by agent i_1 is depicted with double edges.

where $C(\boldsymbol{\pi}) = \sum_{i \in N} C_i(\boldsymbol{\pi})$. For a family of FROGs, PoGPS is the supremum of every PoGPS therein. As shown in Fig. 8, a first negative result follows:

Theorem 8. *Allowing walks$^{(i)}$, GPS-agents may cycle infinitely (Fig.8).*

Proof. As depicted in Fig.8, consider w.l.o.g. the end of a given time step, and the current choice faced by agent i_1. Straight outside shows congestion and is not better than taking the later exit at the next node. Since every agent faces the same choice and the game is symmetric, it is possible to loop endlessly. □

Following Th. 8, we now focus on simple-paths and study the order of PoGPS.

Theorem 9. *The Price of GPS Navigation is in $\Omega(|V| + n)$ as the number of vertices $|V|$ and the number of agents n grow$^{(o)}$.*

Proof. It suffices to generalize the double cycle of Fig. 8 from 4 corners and agents, to a similar double cycle with more corners and agents. Then for every agent, while the shortest path has total-delay in $\Theta(1)$, the decided path can have total-delay in $\Theta(|V|)$ and $\Theta(n)$. □

(o) For two variables x, y, Landau notation $f(x, y) \in \Omega(g(x, y))$ is defined as: $\exists K \in \mathbb{R}_{>0},\ \exists n_0 \in \mathbb{N}_{\geq 0},\ \forall x, y \geq n_0,\ f(x, y) \geq K g(x, y)$.

7 Prospects

The symmetric case seems usually well behaved [HMRT09, Th.1] and would be worth investigating. Time expanded graphs, where one does the cross product of vertices and time or positions may yield an other beautiful approach. A study on tie-breaking under FIFO is motivated by its importance in the proofs. Studying less extreme, average, or sub-cases would be appealing. Extensive forms with decisions on each node [CCCW17b] are a promising model. Finally, one could study the impact of new technologies of IP, like video streaming.

Acknowledgments I am grateful to the anonymous reviewers for their work. Following recent practices, the reviews will be appended to a preprint.

References

[AAE05] Baruch Awerbuch, Yossi Azar, and Amir Epstein. The Price of Routing Unsplittable Flow. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 57–66. ACM, 2005.

[AK97] Paola Alimonti and Viggo Kann. Hardness of Approximating Problems on Cubic Graphs. *Algorithms and Complexity*, pages 288–298, 1997.

[AU09] Elliot Anshelevich and Satish Ukkusuri. Equilibria in Dynamic Selfish Routing. In *International Symposium on Algorithmic Game Theory*, pages 171–182. Springer, 2009.

[BFA15] Umang Bhaskar, Lisa Fleischer, Elliot Anshelevich. A Stackelberg Strategy for Routing Flow over Time. *Games and Economic Behavior*, 2015.

[CCCW17a] Zhigang Cao, Bo Chen, Xujin Chen, and Changjun Wang. A Network Game of Dynamic Traffic. *CoRR*, abs/1705.01784, 2017.

[CCCW17b] Zhigang Cao, Bo Chen, Xujin Chen, and Changjun Wang. A Network Game of Dynamic Traffic. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, EC '17, pages 695–696. ACM, 2017.

[CK05] George Christodoulou and Elias Koutsoupias. The Price of Anarchy of Finite Congestion Games. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 67–73. ACM, 2005.

[Cre97] Pierluigi Crescenzi. A Short Guide to Approximation Preserving Reductions. In *Computational Complexity, 1997. Proceedings., Twelfth Annual IEEE Conference on (Formerly: Structure in Complexity Theory Conference)*, pages 262–273. IEEE, 1997.

[FK96] Uriel Feige and Joe Kilian. Zero Knowledge and the Chromatic Number. In *Computational Complexity, 1996. Proceedings., Eleventh Annual IEEE Conference on*, pages 278–287. IEEE, 1996.

[FOV08] Babak Farzad, Neil Olver, and Adrian Vetta. A Priority-Based Model of Routing. *Chicago Journal of Theoretical Computer Science*, 2008.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability*, volume 174. Freeman San Francisco, CA, 1979.

[GJS74] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some Simplified NP-complete Problems. In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, STOC '74, pages 47–63. ACM, 1974.

[HHP06] Tobias Harks, Stefan Heinz, and Marc E Pfetsch. Competitive Online Multicommodity Routing. In *International Workshop on Approximation and Online Algorithms*, pages 240–252. Springer, 2006.

[HHP09] Tobias Harks, Stefan Heinz, Marc E Pfetsch. Competitive Online Multicommodity Routing. *Theory of Computing Systems*, 45(3):533–554, 2009.

[HMRT09] Martin Hoefer, Vahab Mirrokni, Heiko Röglin, and Shang-Hua Teng. Competitive Routing over Time. *Internet and Network Economics*, pages 18–29, 2009.

[HMRT11] Martin Hoefer, Vahab S Mirrokni, Heiko Röglin, and Shang-Hua Teng. Competitive Routing over Time. *Theoretical Computer Science*, 412(39):5420–5432, 2011.

[HPSVK16] Tobias Harks, Britta Peis, Daniel Schmand, and Laura Vargas Koch. Competitive Packet Routing with Priority Lists. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 58. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[Kar72] Richard M Karp. Reducibility among Combinatorial Problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[Koc12] Ronald Koch. Routing Games over Time. *PhD Thesis*, 2012.

[KP99] Elias Koutsoupias and Christos Papadimitriou. Worst-Case Equilibria. In *STACS*, volume 99, pages 404–413. Springer, 1999.

[KS09] Ronald Koch and Martin Skutella. Nash Equilibria and the Price of Anarchy for Flows over Time. In *International Symposium on Algorithmic Game Theory*, pages 323–334. Springer, 2009.

[KS11] Ronald Koch and Martin Skutella. Nash Equilibria and the Price of Anarchy for Flows over Time. *Theory of Computing Systems*, 49(1):71–97, 2011.

[MLS10] Martin Macko, Kate Larson, and L'ubos Steskal. Braess's Paradox for Flows over Time. In *SAGT*, pages 262–275. Springer, 2010.

[MLS13] Martin Macko, Kate Larson, and L'uboš Steskal. Braess's Paradox for Flows over Time. *Theory of Computing Systems*, 53(1):86–106, 2013.

[NRTV07] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic Game Theory*, volume 1. Cambridge University Press Cambridge, 2007.

[PY91] Christos H Papadimitriou and Mihalis Yannakakis. Optimization, Approximation, and Complexity Classes. *Journal of computer and system sciences*, 43(3):425–440, 1991.

[Rou05] Tim Roughgarden. *Selfish Routing and the Price of Anarchy*, volume 174. Cambridge MIT press, 2005.

[Rou09] Tim Roughgarden. Intrinsic Robustness of the Price of Anarchy. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 513–522. ACM, 2009.

[RT02] Tim Roughgarden and Éva Tardos. How Bad is Selfish Routing? *Journal of the ACM (JACM)*, 49(2):236–259, 2002.

[SST14] Marco Scarsini, Marc Schröder, and Tristan Tomala. Atomic Dynamic Network Games. 2014.

[SST16] Marco Scarsini, Marc Schröder, and Tristan Tomala. Dynamic Atomic Congestion Games with Seasonal Flows. 2016.

[War52] John Glen Wardrop. Some Theoretical Aspects of Road Traffic Research. *Proceedings of the institution of civil engineers*, 1(3):325–362, 1952.

[WBK15] Thomas L Werth, Sabine Büttner, and Sven O Krumke. Robust Bottleneck Routing Games. *Networks*, 66(1):57–66, 2015.

[WHK14] TL Werth, M Holzhauser, SO Krumke. Atomic Routing in a Deterministic Queuing Model. *Operations Research Perspectives*, 1(1):18–41, 2014.

[Zuc06] David Zuckerman. Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 681–690. ACM, 2006.