# ARRIVAL: Recursive Framework & $\ell_1$-Contraction

**Sebastian Haslebacher**
ETH Zurich, Switzerland

## Abstract

ARRIVAL is the problem of deciding which out of two possible destinations will be reached first by a token that moves deterministically along the edges of a directed graph, according to so-called switching rules. It is known to lie in $\mathsf{NP} \cap \mathsf{CoNP}$, but not known to lie in $\mathsf{P}$. The state-of-the-art algorithm due to Gärtner et al. (ICALP '21) runs in time $2^{\mathcal{O}(\sqrt{n}\log n)}$ on an $n$-vertex graph.

We prove that ARRIVAL can be solved in time $2^{\mathcal{O}(k \log^2 n)}$ on $n$-vertex graphs of treewidth $k$. Our algorithm is derived by adapting a simple recursive algorithm for a generalization of ARRIVAL called G-ARRIVAL. This simple recursive algorithm acts as a framework from which we can also rederive the subexponential upper bound of Gärtner et al.

Our second result is a reduction from G-ARRIVAL to the problem of finding an approximate fixed point of an $\ell_1$-contracting function $f : [0,1]^n \to [0,1]^n$. Finding such fixed points is a well-studied problem in the case of the $\ell_2$-metric and the $\ell_\infty$-metric, but little is known about the $\ell_1$-case.

Both of our results highlight parallels between ARRIVAL and the Simple Stochastic Games (SSG) problem. Concretely, Chatterjee et al. (SODA '23) gave an algorithm for SSG parameterized by treewidth that achieves a similar bound as we do for ARRIVAL, and SSG is known to reduce to $\ell_\infty$-contraction.

**2012 ACM Subject Classification** Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Problems, reductions and completeness

**Keywords and phrases** ARRIVAL, G-ARRIVAL, Deterministic Random Walk, Rotor-Routing, $\ell_1$-Contraction, Banach Fixed Point

**Digital Object Identifier** 10.4230/LIPIcs.ICALP.2025.95

**Category** Track A: Algorithms, Complexity and Games

**Related Version** *Full Version*: `https://doi.org/10.48550/arXiv.2502.06477`

**Acknowledgements** I want to thank Bernd Gärtner and Simon Weber for valuable discussions and feedback, and anonymous reviewers for their comments and suggestions.

## 1 Introduction

ARRIVAL is a computational problem first introduced by Dohrau et al. [11]. It can be described as a deterministic process (or zero-player game) on a directed graph with a designated origin $o$ and two designated destinations $d$ and $\overline{d}$. Every vertex in ARRIVAL has out-degree two, and exactly one outgoing edge at every vertex is marked (we also call it the *even* edge). We additionally assume that both destinations are reachable from every vertex in the graph. A token is placed on $o$ and moved along the edges of the graph according to the following rule: At every vertex, the token continues along the outgoing edge that was used least so far. In case of a tie, the token uses the even edge. This effectively means that the token will alternate between the two outgoing edges at every vertex, starting with the even edge. The task is to decide which of the two destinations $d$ or $\overline{d}$ will be visited first by the token.

Dohrau et al. [11] proved that ARRIVAL is contained in $\mathsf{NP} \cap \mathsf{CoNP}$. Naturally, they then asked whether it is also in $\mathsf{P}$. This open problem has received some attention in recent years and the best algorithm to date, due to Gärtner et al. [16], runs in time $2^{\mathcal{O}(\sqrt{n}\log n)}$ on a graph with $n$ vertices.


We present two new results for ARRIVAL: Our first result is an algorithm for ARRIVAL that runs in time $2^{\mathcal{O}\left(k \log^2 n\right)}$ on graphs with $n$ vertices and treewidth $k$. Note that this bound is quasi-polynomial for graphs with bounded treewidth. Our algorithm is obtained by adapting a simple recursive algorithm for G-ARRIVAL, a generalization of ARRIVAL that allows arbitrarily many origins, destinations, and tokens: The recursive algorithm solves an instance with $\ell$ destinations and origins by making recursive calls on instances with $\ell + 1$ destinations and origins. In other words, in each recursive call, a new vertex is made into a destination and origin vertex. By choosing this pivot vertex carefully, we can exploit the underlying graph structure.

It turns out that this simple recursive algorithm for G-ARRIVAL can also be seen as a framework for other algorithms for G-ARRIVAL. Concretely, we explain how the state-of-the-art upper bound $2^{\mathcal{O}(\sqrt{n} \log n)}$ due to Gärtner et al. [16] can be derived in this framework as well.

Our second result is a reduction from G-ARRIVAL to the problem of finding an approximate fixed point of a $\ell_1$-contracting function $f : [0,1]^n \to [0,1]^n$. Concretely, we say that a function $f : [0,1]^n \to [0,1]^n$ is contracting with parameter $\lambda \in [0,1)$ if we have $\|f(x) - f(y)\|_1 \leq \lambda \|x - y\|_1$ for all $x, y \in [0,1]^n$. Such a function is guaranteed to have a unique fixed point by Banach's fixed point theorem [4]. An $\varepsilon$-approximate fixed point $x \in [0,1]^n$ has to satisfy $\|f(x) - x\|_1 \leq \varepsilon$. Given our reduction, any algorithm that can find an $\varepsilon$-approximate fixed point of $f$ in time poly $\left(\log \frac{1}{\varepsilon}, \log \frac{1}{1-\lambda}, n\right)$ would imply a polynomial-time algorithm for G-ARRIVAL.

Finding approximate fixed points is a well-studied problem in the case of the $\ell_2$-metric and the $\ell_\infty$-metric (see e.g. [24, 8]). In particular, efficient algorithms in the case of the $\ell_2$-metric have been known since 1993 [23], and Chen et al. [8] only recently gave the first polynomial query upper bound for the $\ell_\infty$-metric. Even more recently, a generalization of the result by Chen et al. to $\ell_p$-contractions for every $p \in [1, \infty]$ was announced [18]. Concretely, this means that an approximate fixed point of an $\ell_1$-contraction can be found with polynomially many queries to the contraction map.

Unfortunately, the algorithm for $\ell_1$-contractions in [18] is only query-efficient (and not time-efficient). Thus, our reduction currently does not imply better algorithms for G-ARRIVAL. We still think that our reduction is interesting: To the best of our knowledge, it provides the first concrete application for the $\ell_1$-contraction problem. Moreover, the polynomial query upper bound for $\ell_1$-contractions [18] suggests that better (maybe even time-efficient) algorithms could be obtained for $\ell_1$-contraction and hence G-ARRIVAL in the future. For this, it might also be interesting to observe that the $\ell_1$-contraction map obtained through our reduction from G-ARRIVAL has the additional property of being monotone with respect to the coordinate-wise partial order. Given recent work of Batziou et al. [5] on monotone $\ell_\infty$-contractions, it seems plausible that $\ell_1$-contraction and monotonicity could maybe be exploited simultaneously as well.

## 1.1 Related Work

Since ARRIVAL is contained in $\mathsf{NP} \cap \mathsf{CoNP}$, it also naturally fits into the complexity class $\mathsf{TFNP}$, which contains total problems with efficiently verifiable solutions. In fact, after a series of results for containment in subclasses of $\mathsf{TFNP}$ [20, 15], we now know that ARRIVAL is contained in $\mathsf{UEOPL}$ [14].

Going into a slightly different direction, Gärtner et al. [15] proved that ARRIVAL is also contained in UP and CoUP, the analogues of NP and CoNP with unique solutions. In fact, it would not be hard to rederive this using our reduction to $\ell_1$-contraction: The idea is that the fixed point acts as an efficient certificate for both YES- and NO-instances and it must be unique due to the contraction property.

It is also known that ARRIVAL can be solved in polynomial time on some restricted graph classes. For example, a result due to Priezzhev et al. [22] implies that ARRIVAL can be solved by simulation in polynomial time on Eulerian graphs. Other results include polynomial-time algorithms on tree-like multigraphs [1] and path-like multigraphs with many tokens [2]. More recently, algorithms running in quasi-polynomial-time were given for the case of tree-like multigraphs with many tokens [17]. Our algorithm effectively generalizes this result to all graphs of bounded treewidth.

Finally, further variants of ARRIVAL have been studied in the past, including a stochastic variant [25], a recursive variant [26], as well as variants with one or two players [13].

### Comparison to SSG

As mentioned before, both our results show parallels between ARRIVAL and Simple Stochastic Games (SSG). We will briefly discuss the connections between the two problems. Similarly to ARRIVAL, SSG is contained in NP∩CoNP [9] and even UP∩CoUP [6], but no polynomial-time algorithm is known. The state-of-the-art algorithm due to Ludwig [21] runs in randomized subexponential time $2^{\mathcal{O}(\sqrt{n}\log n)}$.

Dohrau et al. [11] already wondered about similarities between ARRIVAL and SSG when they first introduced ARRIVAL. Since then, both problems were shown to reduce to the problem of finding a Tarski fixed point [12, 16], and to be contained in the complexity class UEOPL [14]. Both our results for ARRIVAL further extend this list of similarities: Our upper bound of $2^{\mathcal{O}(k\log^2 n)}$ for ARRIVAL on $n$-vertex graphs of treewidth $k$ is comparable to a similar bound for SSG due to Chatterjee et al. [7]. Moreover, SSG reduces to finding an approximate fixed point of an $\ell_\infty$-contracting function [9], which is analogous to our reduction from ARRIVAL to $\ell_1$-contraction.

Both ARRIVAL and SSG also admit polynomial-time algorithms on graphs with a bounded feedback vertex set [16, 3]. In fact, in the case of ARRIVAL, we will discuss this in more detail in Section 3.2.

## 1.2 Outline

As explained above, our results are actually obtained for a generalization of ARRIVAL called G-ARRIVAL. Thus, we will start Section 2 by formally introducing G-ARRIVAL. We also use Section 2 to recall further terminology and notation from the literature that will be useful for our arguments.

Note that instead of formally introducing the notion of treewidth and tree decompositions, we will directly work with so-called balanced separators instead. The reason for this choice of exposition is that our parameterized algorithm actually exploits the existence of small balanced separators (and not tree decompositions themselves), which are guaranteed to exist in graphs of small treewidth (see Section 2.1 for more details).

We describe our parameterized algorithm in Section 3. We start the exposition with our simple recursive algorithm for G-ARRIVAL (Section 3.1), which provides a framework from which we will derive the parameterized algorithm in Section 3.3. In Section 3.2, we

additionally explain how the subexponential upper bound due to Gärtner et al. [16] and polynomial-time upper bounds on graphs with a bounded feedback vertex set [16] can be rederived from our framework.

Finally, Section 4 contains our reduction to the problem of finding an approximate fixed point of a $\ell_1$-contraction map. Crucially, given an instance of G-ARRIVAL, we define a function $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}^n_{\geq 0}$ that is contracting and thus has a unique fixed point. We then prove that a reasonably good approximation of this fixed point will give away the solution to the G-ARRIVAL-instance. The function can be restricted to a compact subset of $\mathbb{R}^n_{\geq 0}$ and scaled to fit into $[0, 1]^n$, if desired.

## 2 Preliminaries

We start by recalling G-ARRIVAL, which was first formally defined by Hoang [19]. Note that our formulation slightly deviates from the one by Hoang, but is easily seen to be equivalent.

A switch graph is a directed graph $G = (V, E, s_0, s_1)$ with $s_0, s_1 : V \to V$ and $E = \{(v, s_0(v)) \mid v \in V\} \cup \{(v, s_1(v)) \mid v \in V\}$. We consider $E$ to be a multiset, and two edges $(v, s_0(v)), (v, s_1(v)) \in E$ to be distinct objects even if we have $s_0(v) = s_1(v)$. Given a switch graph, we call $s_0(v)$ and $s_1(v)$ the even and odd successor of $v \in V$, respectively. Similarly, we refer to edges induced by $s_0$ as even edges, and to edges induced by $s_1$ as odd edges.

In G-ARRIVAL, many tokens traverse the directed graph simultaneously, starting and ending at special vertices that we call terminals or terminal vertices. Concretely, a G-ARRIVAL-instance consists of a switch graph as well as a non-empty subset $T \subseteq V$ of terminals. For each terminal $v \in T$, we also get a natural number $t^+_v$ of tokens starting at $v$. We always assume that at least one terminal is reachable from each non-terminal vertex $v \in V \setminus T$ (otherwise, tokens could loop indefinitely without ever reaching a terminal). We will also always denote the total number of tokens by $t^+ := \sum_{v \in T} t^+_v$ and assume $t^+ \geq 1$.

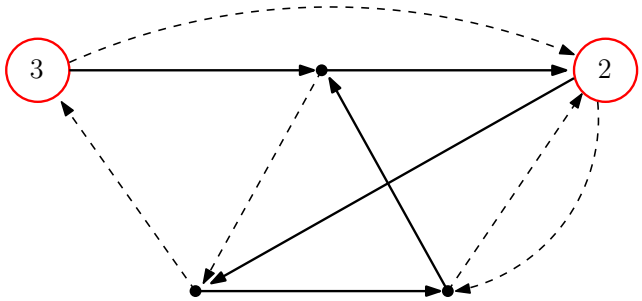


**Figure 1** An instance of G-ARRIVAL. Even edges are bold while odd edges are dashed. Terminals are marked in red. Three tokens start at the left terminal, and two tokens start at the right terminal.

Now consider the following non-deterministic procedure. Initially, for every terminal $v \in T$, move $\lceil \frac{t^+_v}{2} \rceil$ tokens from $v$ to $s_0(v)$, and $\lfloor \frac{t^+_v}{2} \rfloor$ tokens from $v$ to $s_1(v)$ (we do this for every terminal simultaneously). Then, while there exists a token on a non-terminal $v \in V \setminus T$, non-deterministically choose one such token and move it along the out-edge of $v$ that was used fewer times so far. In case of a tie, the token must use the even out-edge $(v, s_0(v)) \in E$. In other words, the even and odd out-edges at $v$ will be used in an alternating fashion, starting with the even out-edge. This procedure stops once all tokens have reached a terminal. The goal of G-ARRIVAL is to predict the number of tokens $t^-_v$ arriving at each terminal $v \in T$.

In order for G-ARRIVAL to be well-defined, we need to make sure that the above procedure terminates and that it always produces the same values $t_v^-$ for all $v \in T$ (independently of the non-deterministic choices). Indeed, both of these properties hold, and this follows by generalizing the arguments of Dohrau et al. [11] to work with multiple tokens, multiple destinations, and multiple origins, as explained by Gärtner et al. [16] and Hoang [19]. To explain this, we first need to recall the concept of switchings flows from the literature.

Given a switch graph $G = (V, E, s_0, s_1)$ with terminals $T \subseteq V$ and starting tokens $(t_v^+)_{v \in T}$, a function $x : E \to \mathbb{R}_{\geq 0}$ satisfying the three constraints

$$x(v, s_0(v)) - x(v, s_1(v)) \in \{0, 1\} \qquad (\forall v \in V)$$

$$\underbrace{\sum_{u:(v,u)\in E} x(v,u)}_{=:x^+(v)} - \underbrace{\sum_{u:(u,v)\in E} x(u,v)}_{:=x^-(v)} = 0 \qquad (\forall v \in V \setminus T)$$

$$\underbrace{\sum_{u:(v,u)\in E} x(v,u)}_{=:x^+(v)} = t_v^+ \qquad (\forall v \in T)$$

is called a switching flow. We will refer to the first set of constraints above as switching behavior, and to the second set of constraints as flow conservation.

Dohrau et al. [11] proved the following theorem in the case of ARRIVAL, but we directly state its generalization for G-ARRIVAL (see also [16, 19]).

▶ **Theorem 1** (Integral Switching Flows are Certificates [11]). *Given a switch graph $G = (V, E, s_0, s_1)$ with terminals $\varnothing \neq T \subseteq V$ and starting tokens $(t_v^+)_{v \in T}$ with $t^+ \geq 1$, the number of tokens $t_v^-$ arriving at terminal $v$ is well-defined and any integral switching flow $x : E \to \mathbb{N}_0$ satisfies $x^-(v) = t_v^-$, for all $v \in T$.*

Concretely, Theorem 1 states that G-ARRIVAL is well-defined and that it can be solved by finding any integral switching flow. Observe that by simulating the non-deterministic procedure outlined before and recording the number of times that each edge is traversed by a token, one can obtain a special integral switching flow that we call the run profile. While the run profile is unique (i.e. it does not depend on non-deterministic choices) [16], Dohrau et al. [11] already observed that in general, integral switching flows are not unique. Concretely, there may be integral switching flows other than the run profile, but Theorem 1 tells us that they still predict the values $(t_v^-)_{v \in T}$ correctly.

Finally, we will need an upper bound on the total flow in any integral switching flow. Similar upper bounds were used in previous work as well (see e.g. [11, 15, 16, 19]). We sketch a short proof.

▶ **Lemma 2** (Upper Bound on Integral Switching Flow [11]). *Let $x : E \to \mathbb{N}_0$ be an arbitrary integral switching flow for the G-ARRIVAL-instance on a switch graph $G = (V, E, s_0, s_1)$ with terminals $\varnothing \neq T \subseteq V$ and starting tokens $(t_v^+)_{v \in T}$ with $t^+ \geq 1$. Then we must have $t^+ = \sum_{v \in T} t_v^+ = \sum_{v \in T} t_v^- = \sum_{v \in T} x^-(v)$ and $x(e) < 2^{|V|} t^+$ for all $e \in E$.*

**Proof.** The equation $\sum_{v \in T} t_v^+ = \sum_{v \in T} t_v^- = \sum_{v \in T} x^-(v)$ follows from flow conservation of switching flows and Theorem 1. We will now prove the upper bound on the flow values.

Let $e = (u, v) \in E$ be arbitrary. Observe that there must exist a simple path $P = (v_0, v_1, v_2, \ldots, v_k)$ with $v_0 = v$ of length $0 \leq k < |V|$ from $v$ to some $v_k \in T$ (recall that we assume that at least one terminal is reachable from every non-terminal in the graph). Observe that by the switching behavior of switching flows, $x(e) \geq 2^{|V|} t^+$ would imply $x^-(v_i) \geq 2^{|V|-i} t^+$ for all $i \in [k]$. In particular, we would have $t_{v_k}^- = x^-(v_k) \geq 2t^+$, contradicting the equation $t^+ = \sum_{v \in T} t_v^-$ above. ◀

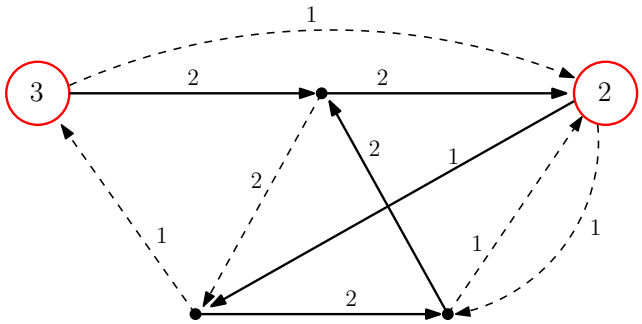


**Figure 2** The number on the edges indicate an integral switching flow. Note that this is not the run profile, but it still certifies (by Theorem 1) that four tokens arrive at the right terminal while only one token arrives at the left terminal. To get the run profile, one would have to decrease the flow on the edges of the directed triangle formed by the three non-terminal vertices by one each.

## 2.1 Treewidth and Balanced Separators

Treewidth is a well-established graph parameter that plays an important role in parameterized algorithms, and it is intimately related to the notion of balanced separators (see e.g. [10, Chapter 7]). As is the case with many applications on graphs of bounded treewidth, our algorithm actually exploits the existence of balanced separators and can be formulated without computations of tree decompositions. Hence, we will refrain from formally introducing tree decompositions and instead focus on balanced separators.

Given an undirected graph $G = (V, E)$ and a subset $S \subseteq V$ of its vertices, we use $G - S$ to denote the graph resulting from deleting the vertices in $S$ and their incident edges from $G$. We call $S$ a balanced separator if each connected component in $G - S$ contains at most $\frac{1}{2}|V|$ vertices. Note that this does not necessarily imply that $G - S$ must have more than one connected component and it could even be an empty graph (despite what the term separator may suggest): Concretely, any set $S$ of size at least $\frac{1}{2}|V|$ is a balanced separator, and thus there always exists a balanced separator. Using a brute-force approach, we can find a smallest balanced separator $S$ in $G$ in time $|V|^{\mathcal{O}(|S|)}$.

The following connection between treewidth and balanced separators is crucial for us.

▶ **Lemma 3** (Balanced Separators and Treewidth [10, Lemma 7.19]). *If $G$ has treewidth at most $k$, then for every $S \subseteq V$, the subgraph $G - S$ has a balanced separator of size at most $k + 1$.*

This lemma allows us to use the treewidth of $G$ to infer the existence of small balanced separators in all induced subgraphs of $G$.

All of the previous concepts are defined on undirected graphs. Since we will be working exclusively with directed graphs, we will adopt the following convention: When used on a directed graph, the terms treewidth and balanced separators are to be interpreted with respect to the underlying simple undirected graph.

## 2.2 Non-Expansive, Contracting, and Monotone Functions

We will be interested in the Manhattan distance, which is induced by the $\ell_1$-norm. Concretely, we use $\|x\| := \sum_{i=1}^{n} |x_i|$ to denote the $\ell_1$-norm of a vector $x \in \mathbb{R}^n$. The Manhattan distance of $x, y \in \mathbb{R}^n$ is then given by $\|x - y\|$.

We are mainly concerned with the non-negative orthant $\mathbb{R}^n_{\geq 0} \subseteq \mathbb{R}^n$. A function $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}^n_{\geq 0}$ is called a $\lambda$-contraction (or is $\lambda$-contracting) for some $\lambda \in [0, 1)$ if and only if $\|f(x) - f(y)\| \leq \lambda \|x - y\|$ for all $x, y \in \mathbb{R}^n_{\geq 0}$. Banach's fixed point theorem [4] implies that such a contracting function admits a unique fixed point. If $f$ only satisfies the weaker property $\|f(x) - f(y)\| \leq \|x - y\|$ for all $x, y \in \mathbb{R}^n_{\geq 0}$, we call it non-expansive instead.

In Section 4, we reduce G-ARRIVAL to the following computational problem: Given access to a $\lambda$-contraction $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}^n_{\geq 0}$, find an $\varepsilon$-approximate fixed point of $f$, i.e. a point $x \in \mathbb{R}^n_{\geq 0}$ such that $\|f(x) - x\| \leq \varepsilon$. We will also point out how the domain of our function $f$ can be restricted to $[0, 1]^n$, if desired.

Another property that we need in some of our proofs is monotonicity with respect to the coordinate-wise partial order. Concretely, we call a function $f : X \subseteq \mathbb{R}^n_{\geq 0} \to \mathbb{R}^n_{\geq 0}$ monotone if and only if $x \leq y$ implies $f(x) \leq f(y)$ for all $x, y \in X$, where $\leq$ denotes coordinate-wise comparison.

▶ **Lemma 4** (Monotonicity in G-ARRIVAL [16]). *Consider a switch graph $G = (V, E, s_0, s_1)$ with terminals $\varnothing \neq T \subseteq V$. Consider the function $f : \mathbb{N}_0^{|T|} \to \mathbb{N}_0^{|T|}$ that maps the vector $(t_v^+)_{v \in T}$ of starting tokens to the vector $(t_v^-)_{v \in T}$ of tokens ending at terminals. The function $f$ is monotone with respect to the coordinate-wise partial order.*

## 3 A Family of Recursive Algorithms

The main goal of this section is to give a parameterized algorithm for G-ARRIVAL that runs in time $2^{\mathcal{O}\left(k \log n \log(n + \log t^+)\right)}$ on graphs with $n$ vertices, treewidth $k$, and a total of $t^+$ starting tokens. Note that for the sake of simplicity, our algorithm recurses by doing a binary search over one vertex at a time. One could instead use algorithms for the so-called Tarski-problem (see e.g. [12]) to recurse by searching over multiple vertices (e.g. all vertices of a balanced separator) at a time. However, with the current best algorithms for Tarski, this would not yield any asymptotic improvements.

### 3.1 A Simple Recursive Algorithm

We start by explaining a simple recursive algorithm for G-ARRIVAL that is inspired by the approach of Gärtner et al. [16]: The algorithm chooses an arbitrary non-terminal $p \in V \setminus T$ that we call the pivot, and makes a guess $a \in \mathbb{N}_0$ for the outflow of $p$ in an integral switching flow. In order to verify the guess, $p$ is converted to a terminal and $t_p^+ := a$ tokens are assigned to start at $p$. In this way, we obtain again an instance of G-ARRIVAL with one more terminal. After solving this subinstance, we can check our guess by looking at the number of tokens $t_p^-$ that arrive at $p$ in the subinstance. If we find that $t_p^- = a = t_p^+$, the integral switching flow obtained for the subinstance is also an integral switching flow for the original instance where $p$ is not a terminal. Otherwise, we have $t_p^- < a$ or $t_p^- > a$ and we use this information to adjust our guess in a binary search fashion. We make this precise in Algorithm 1 and use the remainder of this section to prove that Algorithm 1 is correct.

▶ **Lemma 5** (Analysis of Algorithm 1). *Given an arbitrary G-ARRIVAL-instance consisting of $G = (V, E, s_0, s_1)$ with terminals $\varnothing \neq T \subseteq V$ and starting tokens $(t_v^+)_{v \in T}$ with $t^+ \geq 1$, Algorithm 1 correctly returns an integral switching flow in time $2^{\mathcal{O}\left(|V \setminus T| \log(|V| + \log t^+)\right)}$.*

**Proof.** We start by proving correctness by induction over the size of $V \setminus T$. As a base case, observe that for $T = V$, the algorithm clearly terminates and produces an integral switching flow $x$. Thus, assume now $T \neq V$ and let $p \in V \setminus T$ be the pivot that is chosen by the algorithm. By the induction hypothesis, we can assume that all recursive calls correctly

**Algorithm 1** A simple recursive algorithm.

```
Find-Switching-Flow-1(G = (V, E, s_0, s_1), T ⊆ V, (t_v^+)_{v∈T})
 2 if T = V                                                    // Base Case
 3  then
 4     x(v, s_0) ← ⌈t_v^+ / 2⌉ for all v ∈ V = T
 5     x(v, s_1) ← ⌊t_v^+ / 2⌋ for all v ∈ V = T
 6     return x
 7 choose arbitrary p ∈ V \ T                                  // Binary Search Case
 8 T' ← T ∪ {p}
 9 ℓ ← 0
10 r ← 2^{|V|} t^+
11 while ℓ < r do
12     t_p^+ ← ⌈(ℓ + r) / 2⌉
13     x ← Find-Switching-Flow-1(G, T', (t_v^+)_{v∈T'})
14     if x^-(p) < t_p^+ then
15         r ← t_p^+ − 1
16     if x^-(p) = t_p^+ then
17         return x
18     if x^-(p) > t_p^+ then
19         ℓ ← t_p^+ + 1
```

return an integral switching flow. Consider now the function $f : \{0, 1, \ldots, 2^{|V|}t^+\} \to \mathbb{N}_0$ that maps the guessed number of tokens $t_p^+$ to the value $x^-(p)$ returned by the recursive call. We claim that $f$ is monotone and maps $\{0, 1, \ldots, 2^{|V|}t^+\}$ to itself. Indeed, monotonicity follows directly from Lemma 4. Moreover, we have $f(0) \geq 0$ since by definition, we must have $x^-(p) \geq 0$. For the upper bound, we recycle the argument from Lemma 2: There must exist a path $P = (p, v_1, \ldots, v_k)$ of length $k < |V|$ from $p$ to some $v_k \in T$. Thus, starting $t_p^+ = 2^{|V|}t^+$ tokens at $p$ would imply $x^-(v_k) \geq t^+$ by switching behavior, and hence

$$x^-(p) = t^+ + t_p^+ - \sum_{v \in T} x^-(v) \leq t^+ + t_p^+ - x^-(v_k) \leq t_p^+$$

using Lemma 2. We conclude that binary search will successfully find a correct guess for $t_p^+$ in the given set $\{0, 1, \ldots, 2^{|V|}t^+\}$.

Having proved correctness, we move on to the bound on the overall runtime. Observe that the recursion depth of the algorithm is at most $|V \setminus T|$. Thus, the total number of tokens starting at terminals in any of the recursive call is always bounded from above by $N \coloneqq (1+2^{|V|})^{|V\setminus T|}t^+$. Let now $\mathcal{T}(\ell)$ denote an upper bound on the runtime of the algorithm on subinstances with $\ell$ terminals. We get that

$$\begin{aligned}
\mathcal{T}(|T|) &\leq (c \log N)\mathcal{T}(|T|+1) \\
&\leq (c \log N)^2 \mathcal{T}(|T|+2) \\
&\;\;\vdots \\
&\leq (c \log N)^{|V\setminus T|}\mathcal{T}(|V|)
\end{aligned}$$

for some constant $c$. Using the definition of $N$ and $\mathcal{T}(|V|) \leq \mathcal{O}(|V| \log N)$, this yields an overall runtime of $2^{\mathcal{O}\left(|V\setminus T| \log(|V|+\log t^+)\right)}$, as desired. ◀

## 3.2 Subexponential Upper Bound

Picking the pivot $p \in V \setminus T$ in Algorithm 1 arbitrarily seems quite naive. In this section, we explain how applying the ideas of Gärtner et al. [16] yields a subexponential upper bound.

▶ **Lemma 6** (Algorithm with Diameter-Like Bound [16]). *Consider an arbitrary G-ARRIVAL-instance consisting of $G = (V, E, s_0, s_1)$ with non-empty $T \subseteq V$ and starting tokens $(t_v^+)_{v \in T}$ with $t^+ \geq 1$. Let $\ell := \max_{v \in V \setminus T} dist(v, T)$, where $dist(v, T)$ denotes the shortest path distance from $v$ to any vertex in $T$. There is an algorithm that solves G-ARRIVAL in time $2^\ell \operatorname{poly}(|V|, \log t^+)$.*

▶ **Lemma 7** (Decomposition Lemma [16]). *Let $G = (V, E, s_0, s_1)$ be an arbitrary switch graph with a non-empty set $T \subseteq V$ of terminals. There is an algorithm that finds a set $S \subseteq V \setminus T$ of size $\mathcal{O}(\sqrt{|V|})$ satisfying $\max_{v \in V \setminus (T \cup S)} dist(v, T \cup S) \leq \mathcal{O}(\sqrt{|V|} \log |V|)$ in time $\mathcal{O}(|V|)$.*

Given those two observations, it is now not hard to adapt Algorithm 1 to run in subexponential-time: We first precompute the set $S$ from Lemma 7 in linear time. Then, we run Algorithm 1 with two changes: In the recursive step, we make sure to always pick a pivot $p$ from $S$ instead of all of $V \setminus T$. Further, we add a new base case that applies the algorithm from Lemma 6 as soon as the parameter $\ell$ from Lemma 6 has shrunk to $\mathcal{O}(\sqrt{|V|} \log |V|)$, which is guaranteed to happen at the latest once all of the vertices in $S$ have been turned into terminals. With these two changes, the recursion depth will become $\mathcal{O}(\sqrt{|V|})$, and since the base case also runs in time exponential only in $\mathcal{O}(\sqrt{|V|} \log |V|)$, we get an overall subexponential runtime.

As observed by Gärtner et al. [16], one can do better on graphs with a small feedback vertex set: The crucial ingredient is that G-ARRIVAL can be solved efficiently on acyclic graphs by greedy simulation of the tokens. Using this as a base case and choosing pivots from a feedback vertex set yields yet another instantiation of the framework provided by the recursive algorithm. Concretely, this yields a polynomial-time algorithm on graphs with a bounded feedback vertex set.

## 3.3 Exploiting Balanced Separators

We now describe an adaptation of Algorithm 1 that works well on graphs of small treewidth. The general idea is again to pick the pivot $p \in V \setminus T$ appropriately. Concretely, as mentioned in Section 2.1, small treewidth ensures small balanced separators in all induced subgraphs of our input graph. Thus, it seems intuitive that in each step, we should pick the pivot $p$ from a balanced separator of the graph induced by the remaining non-terminals. Eventually, this should disconnect the induced graph into independent subinstances, each with a significantly smaller number of non-terminals. Recursing on all subinstances independently yields the desired speedup. We make this precise in Algorithm 2 and analyse the algorithm in the remainder of this section.

▶ **Lemma 8** (Correctness of Algorithm 2). *Given an arbitrary G-ARRIVAL-instance consisting of $G = (V, E, s_0, s_1)$ with non-empty $T \subseteq V$, starting tokens $(t_v^+)_{v \in T}$ with $t^+ \geq 1$, and a balanced separator $S$ for $G - T$, Algorithm 2 correctly returns an integral switching flow.*

**Proof.** Correctness of the base case and the binary search case follows from the same arguments as in Lemma 5 (correctness of Algorithm 1). The only thing we changed is that our pivot is chosen from $S$ instead of all of $V \setminus T$.

It remains to argue correctness of the new splitting case. For this, assume that $S = \varnothing$ and $V \neq T$. In particular, $G - T$ has at least one non-empty connected component. For each connected component $C \subseteq V \setminus T$ of $G - T$, the algorithm computes the switch graph $G_C$

**Algorithm 2** The main difference to Algorithm 1 is that we choose our pivot from a smallest balanced separator $S$ that is passed through the recursive calls. As soon as all vertices in the separator have been turned into terminals, we can split the instance into independent subinstances and proceed from there.

```
Find-Switching-Flow-2(G = (V, E, s_0, s_1), T ⊆ V, (t_v^+)_{v∈T}, S ⊆ V \ T)
 2 if T = V                                                        // Base Case
 3  then
 4     x(v, s_0) ← ⌈t_v^+ / 2⌉ for all v ∈ V = T
 5     x(v, s_1) ← ⌊t_v^+ / 2⌋ for all v ∈ V = T
 6     return x
 7 if |S| > 0                                                      // Binary Search Case
 8  then
 9     choose arbitrary p ∈ S
10     T' ← T ∪ {p}
11     S' ← S \ {p}
12     ℓ ← 0
13     r ← 2^{|V|} t^+
14     while ℓ < r do
15        t_p^+ ← ⌈(ℓ+r)/2⌉
16        x ← Find-Switching-Flow-2(G, T', (t_v^+)_{v∈T'}, S')
17        if x^-(p) < t_p^+ then
18           r ← t_p^+ − 1
19        if x^-(p) = t_p^+ then
20           return x
21        if x^-(p) > t_p^+ then
22           ℓ ← t_p^+ + 1
23 else                                                            // Splitting Case
24     for every connected component C ⊆ V \ T of G − T (undirected) do
25        let G_C be the graph (directed) obtained from G by removing all vertices not
           in C ∪ T, their outgoing edges, and replacing edges leaving the set C ∪ T by
           self-loops.
26        find a smallest balanced separator S_C ⊆ C of G_C − T (undirected)
27        x^(C) ← Find-Switching-Flow-2(G_C, T, (t_v^+)_{v∈T}, S_C)
28     combine solutions of each connected component to x (see Lemma 8 for details)
29     return x
```

obtained from $G$ by deleting all vertices $V \setminus (C \cup T)$, their outgoing edges, and replacing edges leaving $C \cup T$ by self-loops. Moreover, assume that we are given an integral switching flow $x^{(C)}$ for each of those G-ARRIVAL-subinstances. Observe that every edge $e = (u, v)$ in $G$ with $u \in V \setminus T$ appears in exactly one subinstance $G_C$: Indeed, we must have $u \in C$ for some connected component $C$, and thus $e$ can only appear in $G_C$. Hence, we can uniquely assign $x(e) \coloneqq x^{(C)}(e)$ for each edge $e$ with corresponding connected component $C$. If we instead have $u \in T$, then $e$ appears in at least one subinstance $G_C$. However, since $u \in T$, the value $x^{(C)}(e)$ must be the same for all subinstances $G_C$ that $e$ appears in. Therefore,

we can safely assign $x(e) \coloneqq x^{(C)}(e)$ for any of those connecteced components $C$. This fully defines $x$, and it remains to prove that it is a switching flow. This is not hard to see, again by distinguishing between vertices from $T$ and $V \setminus T$. For any vertex $u \in T$, switching behavior holds at $u$ because it holds in all subinstances (where the outgoing edges of $u$ have the exact same values even if they were converted into self-loops). Similarly, if we instead have $u \in C$ for some connected component $C$, then $x(u, s_0(u))$ and $x(u, s_1(u))$ are taken from $x^{(C)}$, where switching behavior must hold by the assumption that $x^{(C)}$ is a switching flow. Flow conservation only has to hold for non-terminals, and it holds due to the fact that all incoming edges of $u \in C$ in $G$ must be present in $G_C$ as well, implying that the flow conservation from the subinstance carries over. We conclude that $x$ is indeed an integral switching flow. ◀

▶ **Lemma 9.** *Assume that we run Algorithm 2 on a G-ARRIVAL-instance consisting of $G = (V, E, s_0, s_1)$ with non-empty $T \subseteq V$ and starting tokens $(t_v^+)_{v \in T}$ with $t^+ \geq 1$. Further assume that we input a smallest balanced separator $S$ of the subgraph $G - T$, and assume that $G$ has treewidth at most $k$. Then the algorithm cannot reach recursion depth $k+2$ without at least once recursing in a splitting case. Moreover, if a splitting case is reached, then each connected component $C \subseteq V \setminus (T \cup S)$ satisfies $|C| \leq \frac{|V \setminus T|}{2}$.*

**Proof.** By Lemma 3, $S$ has size at most $k+1$. Thus, by only using the binary search case, the algorithm can reach a recursion depth of at most $k+1$. This implies that to reach depth $k+2$, it must at least once have recursed in a splitting case. This happens once all vertices in $S$ have been turned into terminals, and the remaining non-terminals are $V \setminus (T \cup S)$. Since $S$ was chosen as a smallest balanced separator of $G - T$, each connected component in the graph $G - T - S$ consists of at most $\frac{|V \setminus T|}{2}$ vertices. ◀

▶ **Theorem 10.** *Given a G-ARRIVAL-instance consisting of $G = (V, E, s_0, s_1)$ with non-empty $T \subseteq V$, starting tokens $(t_v^+)_{v \in T}$ with $t^+ \geq 1$, and a smallest balanced separator $S$ of $G - T$, Algorithm 2 computes an integral switching flow $x$ in time $2^{\mathcal{O}\left(k \log(|V \setminus T|) \log(|V| + \log t^+)\right)}$, where $k$ is the treewidth of $G$.*

**Proof.** As in the analysis of Lemma 5, $N \coloneqq (1 + 2^{|V|})^{|V \setminus T|} t^+$ is an upper bound on the total number of starting tokens in any recursive call (since the recursion depth is still certainly at most $|V \setminus T|$). Compared to the analysis in Lemma 5, we now additionally have to include the splitting case, which also includes finding small balanced separators in time at most $|V|^{\mathcal{O}(k)}$. Let $\mathcal{T}(\ell, q)$ denote an upper bound on the runtime of subinstances with $\ell$ non-terminals and a set $S$ of size $q$. Observe that by Lemma 9, we get the upper bound

$$\begin{aligned}
\mathcal{T}(|V \setminus T|, k+1) &\leq (c \log N) \mathcal{T}(|V \setminus T|, k) \\
&\vdots \\
&\leq (c \log N)^{k+1} \mathcal{T}(|V \setminus T|, 0) \leq |V|^{c'k} (c \log N)^{k+1} \mathcal{T}\left(\frac{|V \setminus T|}{2}, k+1\right)
\end{aligned}$$

where the last inequality comes from the splitting case and accounts for the search of new balanced separators. Repeating this, we then get

$$\begin{aligned}
\mathcal{T}(|V \setminus T|, k+1) &\leq \cdots \leq |V|^{c'k} (c \log N)^{k+1} \mathcal{T}\left(\frac{|V \setminus T|}{2}, k+1\right) \\
&\vdots \\
&\leq \cdots \leq |V|^{c'k \log |V \setminus T|} (c \log N)^{(k+1) \log |V \setminus T|} \mathcal{T}(0, 0)
\end{aligned}$$

for constants $c, c'$. Using the definition of $N$ and $\mathcal{T}(0, 0) \leq \text{poly}(\log N, n)$, this implies an overall upper bound of $2^{\mathcal{O}\left(k \log(|V \setminus T|) \log(|V| + \log t^+)\right)}$, as desired. ◀

# 4 Reduction to $\ell_1$-Contraction

The goal of this section is to prove that G-ARRIVAL reduces to finding an approximate fixed point of an $\ell_1$-contracting function. For this, we will frequently use the functions $h_0, h_1 : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ defined as

$$h_0(x) \coloneqq \min\left\{x - \left\lfloor \frac{x}{2} \right\rfloor, \left\lceil \frac{x}{2} \right\rceil\right\} \quad \text{and} \quad h_1(x) \coloneqq \max\left\{\left\lfloor \frac{x}{2} \right\rfloor, x - \left\lceil \frac{x}{2} \right\rceil\right\}$$

for all $x \in \mathbb{R}_{\geq 0}$. Observe that both $h_0$ and $h_1$ are continuous and monotone, and that we have $h_0(x) + h_1(x) = x$ as well as $h_1(x) \leq h_0(x) \leq h_1(x) + 1$ for all $x \in \mathbb{R}_{\geq 0}$.

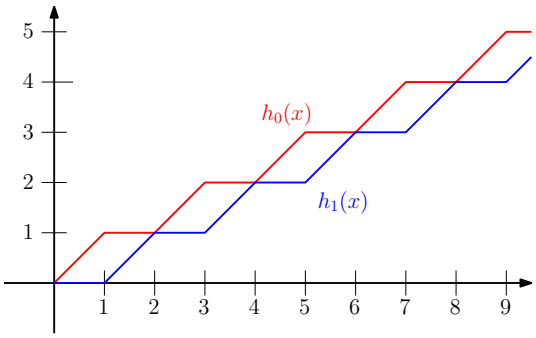


**Figure 3** The functions $h_0$ and $h_1$.

## 4.1 One-Step Update

Consider a vector $x \in \mathbb{R}^n_{\geq 0}$ and think of it as token mass that is distributed among the vertices of a switch graph, with $x_v$ token mass currently occupying vertex $v$. We want to define a notion of moving all (possibly fractional) tokens by one step each while respecting the switching rules. The following one-step update function captures this idea, with the intuition that $f(x)_v$ is the amount token mass that $v \in V$ receives from its predecessors.

▶ **Definition 11** (One-Step Update). *Let $G = (V, E, s_0, s_1)$ be a switch graph. The one-step update function $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}^n_{\geq 0}$ associated with this switch graph is defined as*

$$f(x)_v \coloneqq \sum_{u : s_0(u) = v} h_0(x_u) + \sum_{u : s_1(u) = v} h_1(x_u)$$

*for all $v \in V$.*

▶ **Lemma 12** (Non-Expansiveness and Monotonicity). *The one-step update $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}^n_{\geq 0}$ associated with the switch graph $G = (V, E, s_0, s_1)$ is monotone and non-expansive.*

**Proof.** Monotonicity of $f$ follows directly from monotonicity of $h_0$ and $h_1$. Thus, it remains to prove that $f$ is non-expansive. Let $x, y \in \mathbb{R}^n_{\geq 0}$ be arbitrary. By the previously discussed properties of $h_0$ and $h_1$, we have

$$|x_v - y_v| = |h_0(x_v) - h_0(y_v)| + |h_1(x_v) - h_1(y_v)|$$

for all $v \in V$. With this, we calculate

$$\begin{aligned}
||f(x) - f(y)|| &= \sum_{v \in V} |f(x)_v - f(y)_v| \\
&= \sum_{v \in V} \left| \sum_{u : s_0(u) = v} (h_0(x_u) - h_0(y_u)) + \sum_{u : s_1(u) = v} (h_1(x_u) - h_1(y_u)) \right| \\
&\leq \sum_{u \in V} |h_0(x_u) - h_0(y_u)| + \sum_{u \in V} |h_1(x_u) - h_1(y_u)| \\
&= \sum_{u \in V} |x_u - y_u| \\
&= ||x - y||,
\end{aligned}$$

where we went from summing over every edge $(u, v)$ by its target $v$ to summing over every edge $(u, v)$ by its source $u$ in the inequality-step. ◀

Next, we consider what happens if we reintroduce terminals. Concretely, we want to fix $x_v = t_v^+$ for all $v \in T$ and consider the resulting function on the non-terminal vertices.

▶ **Definition 13** (Extension and Projection). *Let $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}^n_{\geq 0}$ be the one-step update associated with the switch graph $G = (V, E, s_0, s_1)$. Assume that we are given terminals $T \subseteq V$ with starting tokens $(t_v^+)_{v \in T}$. For arbitrary $x \in \mathbb{R}^{|V \setminus T|}_{\geq 0}$, let the extension $x' \in \mathbb{R}^n_{\geq 0}$ of $x$ denote the vector*

$$x'_v = \begin{cases} t_v^+ & \text{if } v \in T \\ x_v & \text{otherwise} \end{cases}$$

*obtained by filling in the values $t_v^+$ for terminals $v \in T$. Moreover, we define the projection $g : \mathbb{R}^{|V \setminus T|}_{\geq 0} \to \mathbb{R}^{|V \setminus T|}_{\geq 0}$ of $f$ to non-terminals as $g(x)_v := f(x')_v$ for all $v \in V \setminus T$ and $x \in \mathbb{R}^{|V \setminus T|}_{\geq 0}$.*

Observe that the projected one-step update $g$ is still non-expansive and monotone: In particular, non-expansiveness can be obtained by

$$\|g(x) - g(y)\| \leq \|f(x') - f(y')\| \leq \|x' - y'\| = \|x - y\|$$

for all $x, y \in \mathbb{R}^{|V \setminus T|}_{\geq 0}$.

The next lemma says that the fixed points of the projected one-step update reveal the solution to the given G-ARRIVAL-instance.

▶ **Lemma 14** (Interpretation of One-Step Update). *Let $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}^n_{\geq 0}$ be the one-step update associated with the switch graph $G = (V, E, s_0, s_1)$. Assume that we are given terminals $T \subseteq V$ with starting tokens $(t_v^+)_{v \in T}$, and let $g : \mathbb{R}^{|V \setminus T|}_{\geq 0} \to \mathbb{R}^{|V \setminus T|}_{\geq 0}$ be the projection of $f$ to non-terminals. Let $x \in \mathbb{R}^{|V \setminus T|}_{\geq 0}$ be arbitrary and consider its extension $x' \in \mathbb{R}^n_{\geq 0}$. Finally, define*

$$y(v, s_0(v)) := h_0(x'_v) \quad y(v, s_1(v)) := h_1(x'_v)$$

*for all $v \in V$. Then the following two statements are true:*

- *If $x$ is a fixed point of $g$ and $y(e)$ is fractional for some edge $e \in E$, then there exists a directed fractional cycle $C$ containing $e$.*
- *$x$ is an integral fixed point of $g$ if and only if $y$ is an integral switching flow.*

**Proof.** Observe first that for every $v \in T$, both $y(v, s_0(v))$ and $y(v, s_1(v))$ are integral (since $t_v^+ \in \mathbb{N}_0$ by assumption). Similarly, for every $v \in V \setminus T$, either $y(v, s_0(v))$ or $y(v, s_1(v))$ is integral.

We are now ready to prove the first statement: Assume that $x$ is a fixed point of $g$ and that $y(e_1)$ is fractional for some edge $e_1 = (u, v) \in E$. By our previous observation, we know that $x_u = y(u, s_0(u)) + y(u, s_1(u))$ must be fractional and that $u \notin T$. Since $x$ is a fixed point, this implies that $g(x)_u$ is fractional as well. By definition of $g$, this means that there must exist some edge $e_2 = (w, u) \in E$ with $y(e_2)$ fractional. We can now repeat this argument until we find a fractional cycle $C$. Observe that $C$ cannot pass through any terminals and that it must come back to $v$ and thus include $e_1$ (because at most one of the two outgoing edges at every non-terminal vertex is fractional).

For the second statement, observe that $y$ satisfies the flow conservation constraints if and only if $x$ is a fixed point of $g$: Indeed, this follows from

$$y^-(v) = \sum_{u:s_0(u)=v} h_0(x'_u) + \sum_{u:s_1(u)=v} h_1(x'_u) = f(x')_v$$

and $y^+(v) = h_0(x'_v) + h_1(x'_v) = x'_v$ for all $v \in V$.

Finally, integrality of $y$ immediately implies integrality of $x$ and vice versa. The definition of $y$ also implies that its values on two out-edges satisfy switching behavior. Hence, if $y$ is integral, then it must be an integral switching flow (since it automatically satisfies the switching constraints). ◀

## 4.2 Discounted One-Step Update

As we have seen, the one-step update function and its projection are non-expansive with respect to the Manhattan distance. In this section, we make them contracting by artificially introducing a contraction factor.

▶ **Definition 15** (Discounted One-Step Update). *Let $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}^n_{\geq 0}$ be the one-step update function and $g : \mathbb{R}^{|V\setminus T|}_{\geq 0} \to \mathbb{R}^{|V\setminus T|}_{\geq 0}$ its projection associated with the switch graph $G = (V, E, s_0, s_1)$ with terminals $T \subseteq V$ and token numbers $(t_v^+)_{v\in T}$. For $\lambda \in [0, 1)$, $f^{(\lambda)} := \lambda f$ is the $\lambda$-discounted one-step update function and $g^{(\lambda)} := \lambda g$ its $\lambda$-discounted projection.*

▶ **Corollary 16** (Contracting and Monotone Discounted One-Step Update). *Let $\lambda \in [0, 1)$ be arbitrary. The $\lambda$-discounted one-step update function $f^{(\lambda)}$ and its $\lambda$-discounted projection $g^{(\lambda)}$ associated with the switch graph $G = (V, E, s_0, s_1)$ are monotone and contracting with parameter $\lambda$.*

**Proof.** Follows from Lemma 12. ◀

▶ **Lemma 17.** *Let $\lambda \in [0, 1)$ be arbitrary. Assume that $g : \mathbb{R}^{|V\setminus T|}_{\geq 0} \to \mathbb{R}^{|V\setminus T|}_{\geq 0}$ is the projected one-step update and $g^{(\lambda)} : \mathbb{R}^{|V\setminus T|}_{\geq 0} \to \mathbb{R}^{|V\setminus T|}_{\geq 0}$ its discounted version associated with the switch graph $G = (V, E, s_0, s_1)$ with terminals $T \subseteq V$ and token numbers $(t_v^+)_{v\in T}$. Assume that $x^* \in \mathbb{R}^{|V\setminus T|}_{\geq 0}$ is the unique fixed point of $g^{(\lambda)}$. Every fixed point $x \in \mathbb{R}^{|V\setminus T|}_{\geq 0}$ of $g$ satisfies $x^* \leq x$.*

**Proof.** Let $x \in \mathbb{R}^{|V\setminus T|}_{\geq 0}$ be an arbitrary fixed point of $g$. We have

$$g^{(\lambda)}(x) = \lambda g(x) = \lambda x \leq x$$

which implies that the function $g^{(\lambda)}$ maps the box $B := [0, x_1] \times \cdots \times [0, x_{|V\setminus T|}]$ to itself. In particular, the unique fixed point $x^*$ of $g^{(\lambda)}$ must lie inside $B$, and we get $x^* \leq x$. ◀

▶ **Lemma 18.** *Let $x^\star \in \mathbb{R}_{\geq 0}^{|V\setminus T|}$ be the unique fixed point of the $\lambda$-discounted projected one-step update function $g^{(\lambda)} : \mathbb{R}_{\geq 0}^{|V\setminus T|} \to \mathbb{R}_{\geq 0}^{|V\setminus T|}$ associated with the switch graph $G = (V, E, s_0, s_1)$ with terminals $T$ and token numbers $(t_v^+)_{v\in T}$. Let $\lambda \in (1 - \frac{1}{t^+ + ||x^\star||}, 1)$. With*

$$y(v, s_0(v)) := \lambda h_0(x_v^\star) \quad \text{and} \quad y(v, s_1(v)) := \lambda h_1(x_v^\star)$$

*for all $v \in V \setminus T$, as well as*

$$y(v, s_0(v)) := \lambda h_0(t_v^+) \quad \text{and} \quad y(v, s_1(v)) := \lambda h_1(t_v^+)$$

*for all $v \in T$, we must have $\sum_{v\in T} y^-(v) > t^+ - 1$.*

**Proof.** Let $z : E \to \mathbb{R}_{\geq 0}$ be an integral switching flow. Let $v \in V \setminus T$ be arbitrary and define the difference $w := z - y$. We have

$$\begin{aligned}
\underbrace{\sum_{u:(u,v)\in E} w(u,v)}_{=w^+(v)} - \underbrace{\sum_{u:(v,u)\in E} w(v,u)}_{=w^-(v)} &= z^+(v) - z^-(v) - y^+(v) + y^-(v) \\
&= 0 - y(v, s_0(v)) - y(v, s_1(v)) + \sum_{u:(u,v)\in E} y(u,v) \\
&= -\lambda x_v^\star + g^{(\lambda)}(x^\star)_v \\
&= (1-\lambda)x_v^\star
\end{aligned}$$

for all $v \in V \setminus T$ and

$$w^+(v) = z^+(v) - y^+(v) = t_v^+ - \lambda t_v^+ = (1-\lambda)t_v^+$$

for all $v \in T$. With $\sum_{v\in V} w^+(v) = \sum_{v\in V} w^-(v)$, we therefore get

$$\sum_{v\in T} w^-(v) = \sum_{v\in V\setminus T} (w^+(v) - w^-(v)) + \sum_{v\in T} w^+(v) \leq (1-\lambda)(t^+ + ||x^\star||) < 1$$

by using our bound on $\lambda$. It remains to observe that this implies

$$\sum_{v\in T} y^-(v) = \sum_{v\in T} (z^-(v) - w^-(v)) > t^+ - 1. \qquad \blacktriangleleft$$

▶ **Theorem 19.** *Deciding an instance of G-ARRIVAL on $n$ vertices with $t^+ \geq 1$ starting tokens reduces to finding a fixed point of the $\lambda$-discounted projected one-step update with $\lambda \in (1 - \frac{1}{t^+(1+n2^n)}, 1)$.*

**Proof.** By Lemma 18, we know that the unique fixed point of the $\lambda$-discounted projected one-step update must send a flow of strictly more than $t^+ - 1$ to the terminals if $\lambda > 1 - \frac{1}{t^+ + ||x^\star||}$. By Lemma 14 and Lemma 17, we know that this implies that any integral switching flow must send at least the same amount of flow to the respective terminals. Since every integral switching flow sends exactly $t^+$ flow to the terminals (Lemma 4), we can infer the values $(t_v^+)_{v\in T}$ from the unique fixed point (by rounding up). The argument in Lemma 2 yields the upper bound $\|x^\star\| \leq n2^n t^+$. $\blacktriangleleft$

Observe that the bound in Lemma 18 can be improved to $\sum_{v\in T} y^-(v) > t^+ - \delta$ by choosing $\lambda > 1 - \delta\frac{1}{t^+(1+n2^n)}$. Furthermore, any $\varepsilon$-approximate fixed point $\hat{x}$ satisfies

$$\|\hat{x} - x^\star\| \leq \|\hat{x} - g^{(\lambda)}(\hat{x})\| + \|g^{(\lambda)}(\hat{x}) - x^\star\| \leq \varepsilon + \|g^{(\lambda)}(\hat{x}) - g^{(\lambda)}(x^\star)\| \leq \varepsilon + \lambda\|\hat{x} - x^\star\|$$

and therefore $\|g^{(\lambda)}(\hat{x}) - g^{(\lambda)}(x^\star)\| \leq \|\hat{x} - x^\star\| \leq \frac{\varepsilon}{1-\lambda}$. Thus, we get

$$\sum_{v \in T} \left|g(\hat{x})_v - t_v^-\right| \leq \sum_{v \in T} |g(\hat{x})_v - g(x^\star)_v| + \sum_{v \in T} \left|g(x^\star)_v - t_v^-\right| \leq \frac{\varepsilon}{1-\lambda} + \delta,$$

which is enough to derive $(t_v^-)_{v \in T}$ from $\hat{x}$ if $\frac{\varepsilon}{1-\lambda} + \delta < \frac{1}{2}$.

Finally, capping the function $g^{(\lambda)}$ in each coordinate to make it map $[0, t^+2^n]^{|V \setminus T|}$ to itself preserves the contraction property. By Lemma 2, the unique fixed point lies inside $[0, t^+2^n]^{|V \setminus T|}$. Scaling everything by a factor of $\frac{1}{t^+2^n}$ yields a $\ell_1$-contracting function that maps $[0,1]^{|V \setminus T|}$ to itself.

## References


1 David Auger, Pierre Coucheney, and Loric Duhazé. Polynomial Time Algorithm for ARRIVAL on Tree-Like Multigraphs. *LIPIcs, Volume 241, MFCS 2022*, 241:12:1–12:14, 2022. doi:10.4230/LIPICS.MFCS.2022.12.

2 David Auger, Pierre Coucheney, Loric Duhazé, and Kossi Roland Etse. Generalized ARRIVAL Problem for Rotor Walks in Path Multigraphs. In *Reachability Problems*, pages 183–198, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-45286-4_14.

3 David Auger, Pierre Coucheney, and Yann Strozecki. Finding Optimal Strategies of Almost Acyclic Simple Stochastic Games. In *Theory and Applications of Models of Computation*, volume 8402, pages 67–85. Springer International Publishing, Cham, 2014. doi:10.1007/978-3-319-06089-7_6.

4 Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta mathematicae*, 3(1):133–181, 1922. doi:10.4064/fm-3-1-133-181.

5 Eleni Batziou, John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. Monotone Contractions, November 2024. doi:10.48550/arXiv.2411.10107.

6 Krishnendu Chatterjee and Nathanaël Fijalkow. A Reduction from Parity Games to Simple Stochastic Games. In *International Symposium on Games, Automata, Logics, and Formal Verification, GandALF*, NA, Italy, 2011. doi:10.4204/EPTCS.54.6.

7 Krishnendu Chatterjee, Tobias Meggendorfer, Raimundo Saona, and Jakub Svoboda. Faster Algorithm for Turn-based Stochastic Games with Bounded Treewidth. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 4590–4605. Society for Industrial and Applied Mathematics, January 2023. doi:10.1137/1.9781611977554.ch173.

8 Xi Chen, Yuhao Li, and Mihalis Yannakakis. Computing a Fixed Point of Contraction Maps in Polynomial Queries. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC 2024, pages 1364–1373, New York, NY, USA, June 2024. Association for Computing Machinery. doi:10.1145/3618260.3649623.

9 Anne Condon. The Complexity of Stochastic Games. *Information and Computation*, 96(2):203–224, February 1992. doi:10.1016/0890-5401(92)90048-K.

10 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer International Publishing, Cham, 2015. doi:10.1007/978-3-319-21275-3.

11 Jérôme Dohrau, Bernd Gärtner, Manuel Kohler, Jiří Matoušek, and Emo Welzl. ARRIVAL: A Zero-Player Graph Game in NP ∩ coNP. In *A Journey Through Discrete Mathematics: A Tribute to Jiří Matoušek*, pages 367–374. Springer International Publishing, Cham, 2017. doi:10.1007/978-3-319-44479-6_14.

12 Kousha Etessami, Christos Papadimitriou, Aviad Rubinstein, and Mihalis Yannakakis. Tarski's Theorem, Supermodular Games, and the Complexity of Equilibria. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:19, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2020.18.

13 John Fearnley, Martin Gairing, Matthias Mnich, and Rahul Savani. Reachability Switching Games. *Logical Methods in Computer Science*, Volume 17, Issue 2, April 2021. doi:10.23638/LMCS-17(2:10)2021.

14 John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. Unique End of Potential Line. *Journal of Computer and System Sciences*, 114:1–35, December 2020. doi:10.1016/j.jcss.2020.05.007.

15 Bernd Gärtner, Thomas Dueholm Hansen, Pavel Hubácek, Karel Král, Hagar Mosaad, and Veronika Slívová. ARRIVAL: Next Stop in CLS. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 60:1–60:13, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2018.60.

16 Bernd Gärtner, Sebastian Haslebacher, and Hung P. Hoang. A Subexponential Algorithm for ARRIVAL. In *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (Lipics)*, pages 69:1–69:14, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2021.69.

17 Ebrahim Ghorbani, Jonah Leander Hoff, and Matthias Mnich. A Quasi-Polynomial Time Algorithm for Multi-Arrival on Tree-Like Multigraphs. In *42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025)*, volume 327 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 39:1–39:19, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.STACS.2025.39.

18 Sebastian Haslebacher, Jonas Lill, Patrick Schnider, and Simon Weber. Query-Efficient Fixpoints of $\ell_p$-Contractions, March 2025. doi:10.48550/arXiv.2503.16089.

19 Hung P. Hoang. *On Two Combinatorial Reconfiguration Problems: Reachability and Hamiltonicity*. Doctoral Thesis, ETH Zurich, 2022. doi:10.3929/ethz-b-000572947.

20 Karthik C. S. Did the Train Reach its Destination: The Complexity of Finding a Witness. *Information Processing Letters*, 121:17–21, May 2017. doi:10.1016/j.ipl.2017.01.004.

21 W. Ludwig. A Subexponential Randomized Algorithm for the Simple Stochastic Game Problem. *Information and Computation*, 117(1):151–155, February 1995. doi:10.1006/inco.1995.1035.

22 V. B. Priezzhev, Deepak Dhar, Abhishek Dhar, and Supriya Krishnamurthy. Eulerian Walkers as a Model of Self-Organized Criticality. *Physical Review Letters*, 77(25):5079–5082, December 1996. doi:10.1103/PhysRevLett.77.5079.

23 K. Sikorski, C.W. Tsay, and H. Woźniakowski. An Ellipsoid Algorithm for the Computation of Fixed Points. *Journal of Complexity*, 9(1):181–200, March 1993. doi:10.1006/jcom.1993.1013.

24 Krzysztof Sikorski. Computational complexity of fixed points. *Journal of Fixed Point Theory and Applications*, 6(2):249–283, December 2009. doi:10.1007/s11784-009-0128-3.

25 Thomas Webster. The Stochastic Arrival Problem. In *Reachability Problems*, pages 93–107, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-031-19135-0_7.

26 Thomas Webster. The Recursive Arrival Problem. *Electronic Proceedings in Theoretical Computer Science*, 390:168–184, September 2023. doi:10.4204/EPTCS.390.11.