ARRIVAL game

Piotr Mikołajczyk

Department of Theoretical Computer Science at Jagiellonian University

March 25, 2020

Sources

- *ARRIVAL: A zero-player graph game in NP $\cap$ coNP* (2017) – by J. Dohrau, B. Gartner, M. Kohler, J. Matoušek, E. Welzl
- *ARRIVAL: Next Stop in CLS* (2018) – by B. Gartner, T. Dueholm Hansen, P. Hubáček, K. Král, H. Mosaad, V Slívová

Introduction

Suppose that a train is running along a railway network of a special nature: every time the train traverses a switch, the switch will change its position immediately afterwards. Hence, the next time the train traverses the same switch, the other direction will be taken, so that directions alternate with each traversal of the switch.

Introduction

Suppose that a train is running along a railway network of a special nature: every time the train traverses a switch, the switch will change its position immediately afterwards. Hence, the next time the train traverses the same switch, the other direction will be taken, so that directions alternate with each traversal of the switch.

Given a network with origin and destination, will the train eventually reach the destination when starting at the origin?

Introduction

Suppose that a train is running along a railway network of a special nature: every time the train traverses a switch, the switch will change its position immediately afterwards. Hence, the next time the train traverses the same switch, the other direction will be taken, so that directions alternate with each traversal of the switch.

Given a network with origin and destination, will the train eventually reach the destination when starting at the origin?

What is the complexity of deciding so?

Switch graph

$G = (V, E, s_0, s_1)$ is a *switch graph*, where:

Switch graph

$G = (V, E, s_0, s_1)$ is a *switch graph*, where:

- V is a set of vertices

Switch graph

$G = (V, E, s_0, s_1)$ is a *switch graph*, where:

- V is a set of vertices
- $s_0, s_1 : V \mapsto V$

Switch graph

$G = (V, E, s_0, s_1)$ is a *switch graph*, where:

- V is a set of vertices
- $s_0, s_1 : V \mapsto V$
- $E = \{(v, s_0(v) : v \in V)\} \cup \{(v, s_1(v) : v \in V)\}$, with loops (v, v) allowed

Convention

Convention

- $s_0(v)$ is the even successor of v, $s_1(v)$ is the odd successor

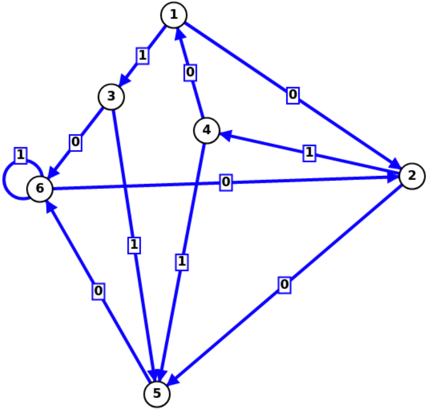
Convention

- $s_0(v)$ is the even successor of v, $s_1(v)$ is the odd successor
- let $n = |V|$

Convention

- $s_0(v)$ is the even successor of v, $s_1(v)$ is the odd successor
- let $n = |V|$
- $E^+(v)$ denotes the set of outgoing edges from v, $E^-(v)$ denotes the set of incoming edges

Example of switch graph



Running train

Given a switch graph $G = (V, E, s_0, s_1)$ with origin and destination vertices $o, d \in V$ we can define such procedure:

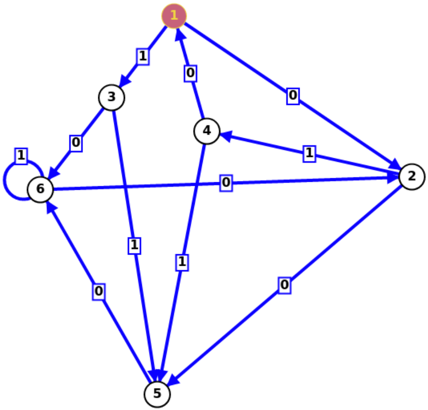
Running train

Given a switch graph $G = (V, E, s_0, s_1)$ with origin and destination vertices $o, d \in V$ we can define such procedure:

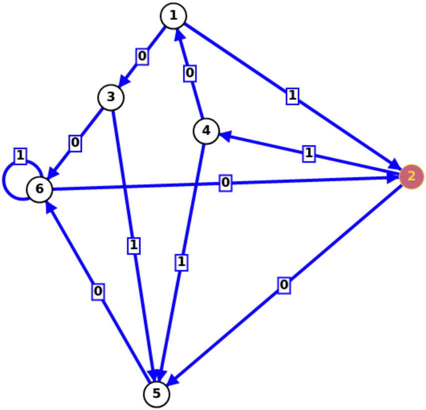
```
procedure Run(G, o, d)
    v := o
    while v ≠ d do
        w := s_curr[v]
        swap (s_curr[v], s_next[v])
        v := w
    end while
end procedure
```

where initially $\texttt{s\_curr}[v] = s_0(v), \texttt{s\_next}[v] = s_1(v)$.

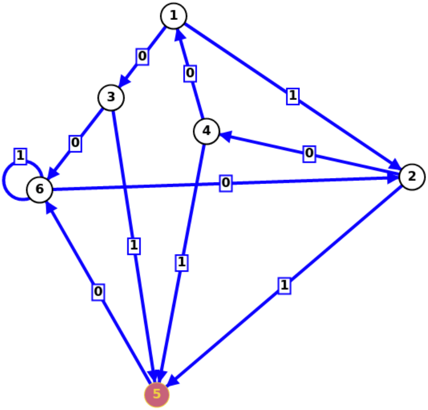
Example run



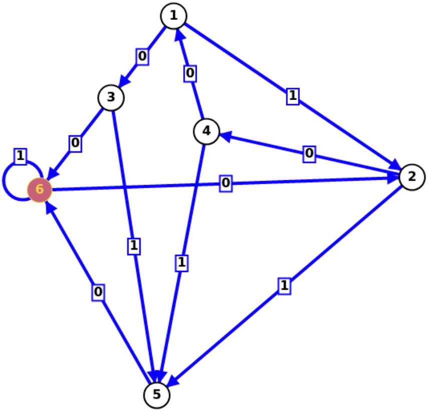
Example run



Example run



Example run



ARRIVAL

Problem *ARRIVAL* is to decide whether procedure $\texttt{Run}(G, o, d)$ terminates for a given switch graph $G = (V, E, s_0, s_1)$ and $o, d \in V$.

Existing research

Existing research

- Most of the existing research is focused on actively controlling switches.

Existing research

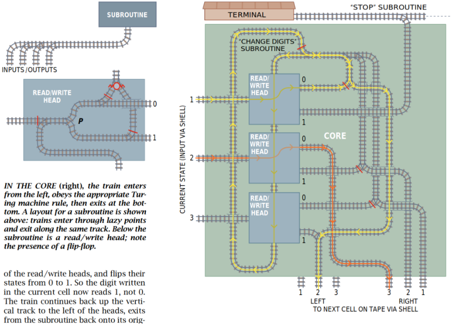
- Most of the existing research is focused on actively controlling switches.
- It was shown (here and here) that if we enrich our network with two other natural switch types, we could simulate Turing machines

Existing research

- Most of the existing research is focused on actively controlling switches.
- It was shown (here and here) that if we enrich our network with two other natural switch types, we could simulate Turing machines
- So *ARRIVAL* becomes *NPC* problem in the richer setting

Existing research

- Most of the existing research is focused on actively controlling switches.
- It was shown (here and here) that if we enrich our network with two other natural switch types, we could simulate Turing machines
- So *ARRIVAL* becomes *NPC* problem in the richer setting



IN THE CORE (right), *the train enters from the left, obeys the appropriate Turing machine rule, then exits at the bottom. A layout for a subroutine is shown above: trains enter through lazy points and exit along the same track. Below the subroutine is a read/write head; note the presence of a flip-flop.*

of the read/write heads, and flips their states from 0 to 1. So the digit written in the current cell now reads 1, not 0. The train continues back up the vertical track to the left of the heads, exits from the subroutine back onto its orig-

Decidability

Theorem 1.

Problem ARRIVAL is decidable.

Decidability

Theorem 1.

Problem ARRIVAL is decidable.

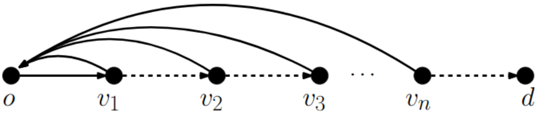


Figure: Switch graph, on which *RUN* procedure takes exponential number of steps. Solid edges point to the even successors, dashed point to the odd.

Theorem 2.

Problem ARRIVAL is in NP.

NP

Theorem 2.

Problem ARRIVAL is in NP.

- natural witness – *run profile*

NP

Theorem 2.

Problem ARRIVAL is in NP.

- natural witness – *run profile*
- fake run profiles may fool the verifier

Switching flow

Switching flow

- let $G = (V, E, s_0, s_1)$ be the switch graph

Switching flow

- let $G = (V, E, s_0, s_1)$ be the switch graph
- let $o, d \in V$ be origin and destination

Switching flow

- let $G = (V, E, s_0, s_1)$ be the switch graph
- let $o, d \in V$ be origin and destination
- $x : E \mapsto \mathbb{N}$ is a *switching flow* if:

Switching flow

- let $G = (V, E, s_0, s_1)$ be the switch graph
- let $o, d \in V$ be origin and destination
- $x : E \mapsto \mathbb{N}$ is a *switching flow* if:

$$\forall_{v \in V} \sum_{e \in E^+(v)} x(e) - \sum_{e \in E^-(v)} x(e) = \begin{cases} 1, & v = o \\ -1, & v = d \\ 0, & otherwise \end{cases}$$

$$\forall_{v \in V}\ 0 \leq x((v, s_1(v))) \leq x((v, s_0(v))) \leq x((v, s_1(v))) + 1$$

Run profile vs switching flow

Observation 1.

Let $G = (V, E, s_0, s_1)$ *be a switch graph, and let* $o, d \in V, o \neq d$*, such that* $\mathit{Run}(G, o, d)$ *terminates. Let* $x(G, o, d) : E \mapsto \mathbb{N}$ *(the run profile) be the function that assigns to each edge the number of times it has been traversed during* $\mathit{Run}(G, o, d)$*. Then* $x(G, o, d)$ *is a switching flow.*

Fake flows

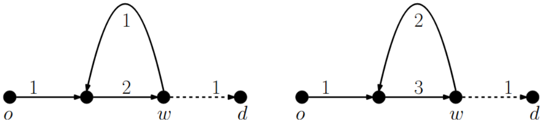


Figure 2: Run profile (left) and fake run profile (right); both are switching flows. Solid edges point to even or unique successors, dashed edges to odd successors.

Switching flow is enough

Lemma 1.

Let $G = (V, E, s_0, s_1)$ be a switch graph, and let $o, d \in V, o \neq d$. If there exists a switching flow x, then $\mathcal{Run}(G, o, d)$ terminates, and $x(G, o, d) \leq x$ (componentwise).

Switching flow is enough

Lemma 1.

Let $G = (V, E, s_0, s_1)$ *be a switch graph, and let* $o, d \in V, o \neq d$. *If there exists a switching flow* x, *then* $\mathcal{Run}(G, o, d)$ *terminates, and* $x(G, o, d) \leq x$ *(componentwise).*

- during the run, flow conservation (w.r.t. to the remaining pebbles) always holds, except at d, and at the current vertex which has one more pebble on its outgoing edges

Switching flow is enough

Lemma 1.

Let $G = (V, E, s_0, s_1)$ *be a switch graph, and let* $o, d \in V, o \neq d$*. If there exists a switching flow* x*, then* $\mathcal{Run}(G, o, d)$ *terminates, and* $x(G, o, d) \leq x$ *(componentwise).*

- during the run, flow conservation (w.r.t. to the remaining pebbles) always holds, except at d, and at the current vertex which has one more pebble on its outgoing edges
- by alternation, starting with the even successor, the numbers of pebbles on $(v, s_0(v))$ and $(v, s_1(v))$ always differ by at most one, for every vertex v

So *NP*

Theorem 2.

Problem ARRIVAL is in NP.

coNP

Theorem 3.

Problem ARRIVAL is in coNP.

coNP

- dead vertices (dead ends)

coNP

- dead vertices (dead ends)
- dead edges

coNP

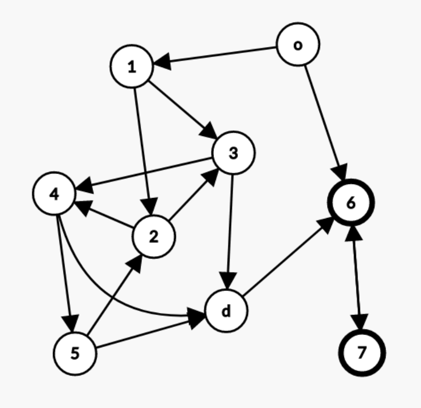
- dead vertices (dead ends)
- dead edges
- hopeful edges

coNP

- dead vertices (dead ends)
- dead edges
- hopeful edges
- edge desperation

coNP

- dead vertices (dead ends)
- dead edges
- hopeful edges
- edge desperation



Lemmas

Lemma 2.

Let $G = (V, E, s_0, s_1)$ *be a switch graph,* $o, d \in V, o \neq d$*, and let* $e = (v, w) \in E$ *be a hopeful edge of desperation* k*. Then* $\mathtt{Run}(G, o, d)$ *will traverse* e *at most* $2^k + 1 - 1$ *times.*

Lemmas

Lemma 2.

Let $G = (V, E, s_0, s_1)$ *be a switch graph,* $o, d \in V, o \neq d$*, and let* $e = (v, w) \in E$ *be a hopeful edge of desperation* k*. Then* $\mathtt{Run}(G, o, d)$ *will traverse* e *at most* $2^k + 1 - 1$ *times.*

Proof by induction on k.

Lemmas

Lemma 2.

Let $G = (V, E, s_0, s_1)$ *be a switch graph,* $o, d \in V, o \neq d$*, and let* $e = (v, w) \in E$ *be a hopeful edge of desperation* k*. Then* $\mathit{Run}(G, o, d)$ *will traverse* e *at most* $2^k + 1 - 1$ *times.*

Proof by induction on k.

Lemma 3.

Let $G = (V, E, s_0, s_1)$ *be a switch graph, and let* $o, d \in V, o \neq d$*. If* $\mathit{Run}(G, o, d)$ *does not terminate, it will reach a dead end.*

Lemmas

Lemma 2.

Let $G = (V, E, s_0, s_1)$ *be a switch graph,* $o, d \in V, o \neq d$*, and let* $e = (v, w) \in E$ *be a hopeful edge of desperation* k*. Then* $\mathcal{Run}(G, o, d)$ *will traverse* e *at most* $2^k + 1 - 1$ *times.*

Proof by induction on k.

Lemma 3.

Let $G = (V, E, s_0, s_1)$ *be a switch graph, and let* $o, d \in V, o \neq d$*. If* $\mathcal{Run}(G, o, d)$ *does not terminate, it will reach a dead end.*

Proof based on *Lemma 1*.

coNP

Theorem 3.

Problem ARRIVAL is in coNP.

Given instance (G, o, d) we will construct (in polynomial time) another instance $(\bar{G}, o, \bar{d})$ such that `Run` on the first one terminates iff it does not terminate on the second one.

coNP

- $\bar{V} = V \cup \{\bar{d}\}$

coNP

- $\bar{V} = V \cup \{\bar{d}\}$
- if v was a dead end: $\bar{s}_0(v) = \bar{s}_1(v) = \bar{d}$

coNP

- $\bar{V} = V \cup \{\bar{d}\}$
- if v was a dead end: $\bar{s_0}(v) = \bar{s_1}(v) = \bar{d}$
- $\bar{s_0}(d) = \bar{s_1}(d) = d$

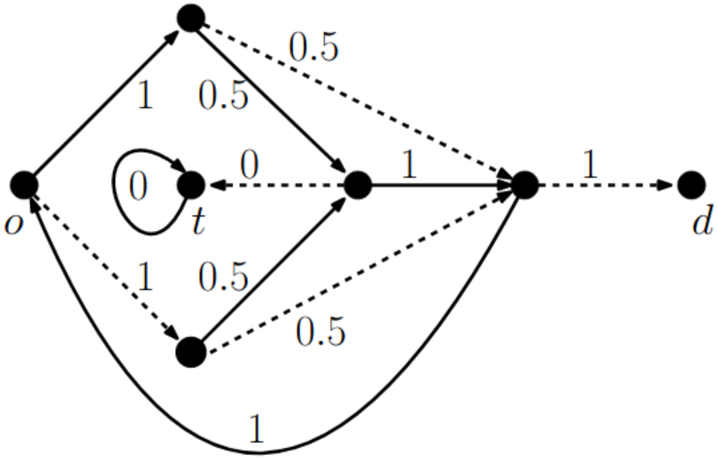
coNP

- $\bar{V} = V \cup \{\bar{d}\}$
- if v was a dead end: $\bar{s_0}(v) = \bar{s_1}(v) = \bar{d}$
- $\bar{s_0}(d) = \bar{s_1}(d) = d$
- for the rest: $\bar{s_0}(v) = s_0(v), \bar{s_1}(v) = s_1(v)$

Theorem 4.

Let $G = (V, E, s_0, s_1)$ be a switch graph, and let $o, d \in V, o \neq d$. $\mathcal{Run}(G, o, d)$ terminates if and only if there exists an integer solution satisfying the constraints (1) and (2). In this case, the run profile $x(G, o, d)$ is the unique integer solution that minimizes the linear objective function $\sum x = \sum_{e \in E} x(e)$ subject to the constraints (1) and (2).

No integer solution



Further results

- $ARRIVAL \in UP \cap coUP$
- $S - ARRIVAL \in PLS$
- $S - ARRIVAL \in CLS$