Information Processing Letters 121 (2017) 17–21

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Did the train reach its destination: The complexity of finding a witness

Karthik C. S. [1]

Department of Applied Mathematics and Computer Science, Weizmann Institute of Science, Israel

ARTICLE INFO

Article history:
Received 13 September 2016
Received in revised form 6 January 2017
Accepted 8 January 2017
Available online 16 January 2017
Communicated by B. Doerr

Keywords:
Switch graphs
PLS
Cellular automaton
Combinatorial problems

ABSTRACT

Recently, Dohrau et al. studied a zero-player game on switch graphs and proved that deciding the termination of the game is in **NP** ∩ **coNP**. In this short paper, we show that the search version of this game on switch graphs, i.e., the task of finding a witness of termination (or of non-termination) is in **PLS**.

1. Introduction

Over the years, switch graphs have been a natural model for studying many combinatorial problems (see [4] and references therein). Dohrau et al. [1] study a problem on switch graphs, which as they suggest fits well in the theory of cellular automata. Informally, they describe their problem in the following way.

> "Suppose that a train is running along a railway network, starting from a designated origin, with the goal of reaching a designated destination. The network, however, is 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, what is the complexity of deciding whether the train, starting at the origin, will eventually reach the destination?

They showed that deciding the above problem lies in **NP** ∩ **coNP**.

In this paper, we address the complexity of the search version of the above problem. From a result of Megiddo and Papadimitriou [5], we have that $F(\mathbf{NP} \cap \mathbf{coNP}) \subseteq \mathbf{TFNP}$, i.e., the search version of any decision problem in **NP** ∩ **coNP** is in **TFNP**. We show that the search version of the problem considered by Dohrau et al. is in **PLS**, a complexity class inside **TFNP** that captures the difficulty of finding a locally optimal solution in optimization problems.

2. Preliminaries

We use the following notation $[n] = \{1, \dots, n\}$ and $[\![n]\!] = \{0, \dots, n\}$. We recapitulate here the definition of the complexity class **PLS**, introduced by Johnson et al. [3]. There are many equivalent ways to define the class **PLS** and below we define it through its complete problem LocalOpt similar to [2].

Definition 1 *(LocalOpt).* Given circuits $S : \{0,1\}^n \to \{0,1\}^n$, and $V : \{0,1\}^n \to [2^n]$, find a string $x \in \{0,1\}^n$ satisfying $V(x) \geq V(S(x))$.

E-mail address: karthik.srikanta@weizmann.ac.il.

[1] This work was partially supported by ISF-UGC 1399/14 grant.

http://dx.doi.org/10.1016/j.ipl.2017.01.004

Definition 2. PLS is the class of all search problems which are polynomial time reducible to LocalOpt.

Below, we recollect some definitions introduced in [1].

2.1. Arrival problem

We start with the object of study: switch graphs. We use the exact same notations as in [1].

Definition 3 *(Switch graph).* A switch graph is a 4-tuple $G = (V, E, s_0, s_1)$, where $s_0, s_1 : V \to V$, $E = \{(v, s_0(v)) \mid v \in V\} \cup \{(v, s_1(v)) \mid v \in V\}$, with self-loops allowed. For every vertex $v \in V$, we refer to $s_0(v)$ as the even successor of v, and we refer to $s_1(v)$ as the odd successor of v. For every $v \in V$, $E^+(v)$ denotes the set of outgoing edges from v, and $E^-(v)$ denotes the set of incoming edges to v.

Next, consider a formal definition of the procedure Run, which captures the run of the train described in the introduction.

Definition 4 *(Run procedure given in [1]).* Given a switch graph $G = (V, E, s_0, s_1)$ with origin and destination $o, d \in V$, the procedure Run is described below. For the procedure, we assume arrays `s_curr` and `s_next`, indexed by V, such that initially `s_curr`$[v] = s_0(v)$ and `s_next`$[v] = s_1(v)$ for all $v \in V$.

procedure Run(G,o,d)
  $v := o$
  while $v \neq d$ **do**
    $w :=$ `s_curr`$[v]$
    swap (`s_curr`$[v]$, `s_next`$[v]$)
    $v := w$ ▷ traverse edge (v, w)
  end while
end procedure

The problem Arrival, considered in [1] is the following.

Problem 1 *(Arrival).* Given a switch graph $G = (V, E, s_0, s_1)$, an origin $o \in V$, and a destination $d \in V$, the problem Arrival is to decide if the procedure Run terminates or not.

Theorem 1 *([1]). Arrival is in* **NP** $\cap$ **coNP**.

In order to prove the above result, the authors consider the run profile as a witness. Elaborating, the run profile is a function which assigns to each edge the number of times it has been traversed during the procedure Run. It is easy to note that a run profile has to be a switching flow.

Definition 5 *(Switching flow, as defined in [1]).* Let $G = (V, E, s_0, s_1)$ be a switch graph, and let $o, d \in V$, $o \neq d$. A switching flow is a function $\boldsymbol{x} : E \to \mathbb{N}_0$ (where $\boldsymbol{x}(e)$ is denoted as x_e) such that the following two conditions hold for all $v \in V$.

$$\sum_{e \in E^+(v)} x_e - \sum_{e \in E^-(v)} x_e = \begin{cases} 1, & v = o, \\ -1, & v = d, \\ 0, & \text{otherwise}. \end{cases} \quad (1)$$

$$0 \le x_{(v,s_1(v))} \le x_{(v,s_0(v))} \le x_{(v,s_1(v))} + 1. \quad (2)$$

Note that while every run profile is a switching flow, the converse is not always true as the balancing condition (2) fails to capture the strict alternation between even and odd successors. Nonetheless, the existence of a switching flow implies the termination of the Run procedure (Lemma 1 of [1]).

3. S-Arrival problem

Now, we describe a reduction from an instance of Arrival to two instances of Arrival (this is an implicit step in the proof of Theorem 1). Given an instance (G, o, d) of Arrival, we build two new instances of Arrival, $(H, \overline{o}, d)$ and $(H, \overline{o}, \overline{d})$, where $H = (V \cup \{\overline{o}, \overline{d}\}, E', s'_0, s'_1)$ is a switch graph specified below. Let X_d be the following subset of the vertex set of G:

$$X_d = \{v \mid \text{There is no directed path in } G \text{ from } v \text{ to } d\}.$$

The vertex set of H is the vertex set of G with the addition of two new vertices $\overline{o}$ and $\overline{d}$. We define $s_0(\overline{o}) = s_1(\overline{o}) = o$. For $i \in \{0, 1\}$ and $v \in V \cup \{\overline{d}\}$, we have that s'_i of H is obtained from s_i of G as follows.

$$s'_i(v) = \begin{cases} v, & v \in \{d, \overline{d}\}, \\ \overline{d}, & v \in X_d, \\ s_i(v), & \text{otherwise}. \end{cases}$$

This reduction has the following property.

Claim 1. *If (G, o, d) is an YES instance of Arrival then, $(H, \overline{o}, d)$ is an YES instance of Arrival and $(H, \overline{o}, \overline{d})$ is a NO instance of Arrival. On the other hand, if (G, o, d) is a NO instance of Arrival then, $(H, \overline{o}, d)$ is a NO instance of Arrival and $(H, \overline{o}, \overline{d})$ is an YES instance of Arrival.*

The proof of the above claim follows from the proof of Theorem 3 in [1]. We are now ready to describe a search version of the Arrival problem.

Problem 2 *(S-Arrival).* Given a switch graph $G = (V, E, s_0, s_1)$, an origin $o \in V$, and a destination $d \in V$, the problem S-Arrival is to either find a switching flow of $(H, \overline{o}, d)$ or a switching flow of $(H, \overline{o}, \overline{d})$.

We have that from Lemma 1 of [1], a switching flow of $(H, \overline{o}, d)$ is an **NP**-witness for the existence of a run profile of (G, o, d), and that a switching flow of $(H, \overline{o}, \overline{d})$ is a **coNP**-witness for the non-existence of a run profile of (G, o, d). Thus, S-Arrival is the appropriate search version problem of the Arrival problem. From Claim 1, S-Arrival is clearly in **TFNP**. In the next section, we show that S-Arrival is in **PLS**, a subclass of **TFNP**.

Below, we essentially show that switching flows are bounded, and this is a critical result in order to establish the reduction in Section 4. Note that this is a strengthening of Lemma 2 in [1] which provided a bound on the run profile.