Journal of Computational Science 26 (2018) 39–54

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

A cellular automaton that solves distributed spanning tree problem

Souvik Roy*, Arunima Ray, Sukanta Das

Department of Information Technology, Indian Institute of Engineering Science and Technology, Shibpur, Howrah, West Bengal 711103, India

ARTICLE INFO

Article history:
Received 14 August 2017
Received in revised form 27 January 2018
Accepted 6 March 2018
Available online 10 March 2018

Keywords:
Cellular automata (CAs)
Anonymous systems
Concurrent initiator distributed spanning tree problem
Grid graph

ABSTRACT

This work introduces the distributed spanning tree problem in the domain of cellular automata. We present a cellular automaton that computes a spanning tree of a given (grid) graph. The time required for this computation is $\mathcal{O}(n \log n)$, where n is the number of nodes of the graph.

1. Introduction

A cellular automaton (CA) is defined over a regular grid, each cell of which consists of a finite automaton that interacts with its neighbours to go to its next state [1]. One of the exciting aspects of CA is to solve computational problems, such as leader election problem [2–7], mutual exclusion problem [8], shortest path problem [9], generation of circles and parabola [10], density classification problem [11–13] and firing squad synchronization problem [14]. In this paper we attempt to solve another computational problem, named distributed spanning tree problem [15] using a CA.

1.1. Distributed spanning tree problem

A spanning of a graph is a tree that covers all the vertices of the graph. Finding of spanning tree of a given graph is a well known computational problem. Under distributed stetting of the same problem, it is considered that the nodes of the graph are themselves computing elements and they compute locally. The problem is to find a spanning tree of the graph by means of independent local computation [15].

In this present work, we consider the anonymous instance of the distributed spanning tree problem. In distributed systems, non-anonymous models assume that the system's components (i.e. nodes) are distinguishable, which requires a central entity to assign unique IDs to the components (i.e. nodes). It violates the very essence of a distributed control. However, a distributed system is anonymous if unique identities are not available to distinguish the nodes.

1.2. History of the problem and motivation

There has been a great interest among the distributed system researchers to solve this problem under different assumptions. Many of those contributions are concerned with non-anonymous instance of spanning trees, see for example [15–19], this problem is also addressed under parallel settings [20–27]. In anonymous instance of the problem, all the nodes can initiate the construction of spanning tree concurrently. For anonymous instance of spanning tree construction, Augluin [28] showed that it is impossible to deterministically construct a spanning tree in uniform networks. Therefore, the research effort shifted towards single initiator spanning tree algorithm, see for example [15–19]. In this case, the algorithm requires a central entity to assign a single initiator. Though a single initiator must register first, which again violates the very essence of a distributed control.

One of the properties of CA is anonymous cells, which do not have any pre-defined identification number. In fact, this property makes the problem interesting in the domain of cellular automata (CAs), because a CA based solution to this problem could imply the existence of solutions of the same problem in anonymous systems. This is one of the motivations of undertaking this research. In the literature of cellular automata, few researchers have explored dif-

* Corresponding author.
E-mail addresses: svkr89@gmail.com (S. Roy), rayarunima55@gmail.com (A. Ray), sukanta@it.iiests.ac.in (S. Das).

https://doi.org/10.1016/j.jocs.2018.03.005
1877-7503/

ferent new type of cellular automata-based, *Physarum*-inspired,[1] network design models which can also solve the distributed spanning tree problem [29–32].

In this context, it is worth mentioning that the problem is closely related to the *leader election problem*. Electing leader in the context of spanning tree problem indicates finding of a single initiator for the spanning tree problem. The literature of leader election problem as a cellular automaton computational task is remarkably rich; see for example [2–6]. Note that, tree computation and leader election is also widely used in algorithms of reconfiguration of array processors [33,34].

In this scenario, this paper addresses the distributed spanning tree problem as a *new* cellular automaton computational problem. In the proposed solution, all the nodes of input graph initiate the computation.

2. Overview of the computation

2.1. The CA model

Classically, cellular automata (CAs) are dynamical systems that are homogeneous and discrete in both time and space [1]. A CA can be defined as a two-dimensional grid $\mathcal{L} \subseteq \mathbb{Z}^2$ of identical automata (cells). Formally, CA is a triple $(\mathcal{S}, \mathcal{N}, f)$, where $\mathcal{S}$ is the finite set of states, $\mathcal{N}$ is the self and non-orthogonal four neighbouring cells which is the standard *von Neumann* neighbourhood,[2] $f : \mathcal{S}^5 \longrightarrow \mathcal{S}$ is a local rule or transition function of the automaton. Here, $\Sigma \subseteq \mathcal{S}$ denotes the set of *input alphabet*.

At any given time, the configuration of the automaton is a mapping $m : \mathcal{L} \longrightarrow \mathcal{S}$ that specifies the states of all cells. The set $\mathcal{S}^{\mathcal{L}}$ of all configurations is denoted by $\mathcal{C}$. After one time stamp, the next configuration of c becomes $e = G(c)$; $G : \mathcal{C} \longrightarrow \mathcal{C}$ is the *global transition function* of the CA. During the evolution of a CA, it generates a sequence of configurations, and sometime approaches to fixed point. A fixed point is a CA configuration, next configuration of which is the configuration itself.

CAs are the natural model of computation which can compute a function $\mathcal{T}$. We call that a CA computes a function $\mathcal{T}$ if an input to $\mathcal{T}$ is an initial configuration $\mathcal{I} \subset \mathcal{C}$ of the CA, and the final configuration $\mathcal{F} \subset \mathcal{C}$ is the evaluated value of $\mathcal{T}$. Let us consider two configurations x and z, where $x \in \mathcal{I}$, $z \in \mathcal{F}$, and $z = \mathcal{T}(x)$. For a CA and a finite time $T \in \mathbb{N}$, if $z = G^T(x)$, then the CA computes $\mathcal{T}$ in time T.

A CA solving spanning tree problem, transforms an initial configuration, representing a connected graph, to a final configuration, which represents the corresponding spanning tree of the graph. Hence, the rule for the CA represents the corresponding (distributed) algorithm for the problem. In this work, we consider that a fixed point is final configuration of any computation. The proposed CA, however, considers only *grid graphs*.

2.2. Grid graph and configurations

Definition 1. A graph $G_k(n) = (V, E)$ is a $k \times n$ $(k, n \in \mathbb{Z})$ grid graph if the vertex and edges are as follows

$$V \subseteq \{V_{i,j} | 1 \le i \le k, 1 \le j \le n\}$$
$$E \subseteq \{(V_{i,j}, V_{a,b}) \,|\, |i-a| + |j-b| = 1\}$$

where, any two vertex $V_{i,j}, V_{a,b} \in V$, there is a path from $V_{i,j}$ to $V_{a,b}$.

[1] *Physarum polycephalum* (*P. polycephalum*), which is a large single cell visible by an unaided eye, has been proven as a reliable living substrate for implementing biological computing devices for computational geometry, graph theoretical problems etc. [29–32].

[2] $\mathcal{N} = \{(0,0),(-1,0),(0,1),(1,0),(0,-1)\}$

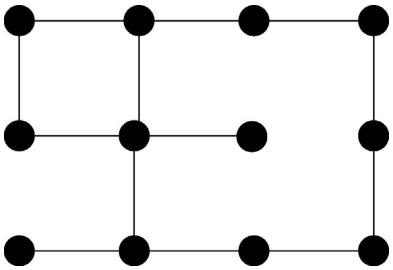

Fig. 1. A grid graph.

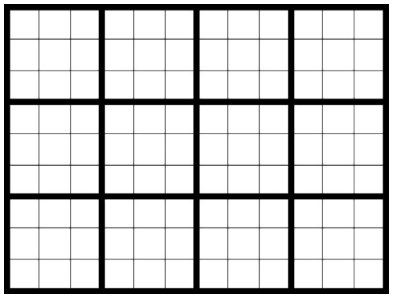

Fig. 2. 2-D grid structure: the blocks are surrounded by thicker lines.

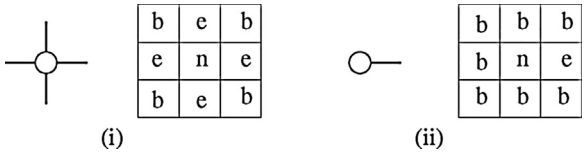

Fig. 3. (i) A node with 4 edges and corresponding grid representation; (ii) a node with 1 edge and corresponding grid representation.

Fig. 1 shows an example gird graph where $k = 4$, $n = 3$. Hence, the arrangement of CA cells $\mathcal{L}$ forms a grid graph, and any other grid graphs are subgraph of $\mathcal{L}$. It is, therefore, natural to emulate any grid graph by a CA grid. In this paper, we consider two-dimensional grid with five neighbourhood dependency to find the spanning tree of grid graphs.

To find out spanning tree of a graph, a CA takes the graph as its initial configuration. So, our first task is to represent a graph as a configuration. Let us divide the CA grid as blocks of 3×3 cells (see Fig. 2). These blocks also form a grid in meta-level. A block is used to represent a node, and the edges connected to it. We name of block as $B_{i,j}$ which represents a vertex $V_{i,j}$ of a grid graph, which $i \in \mathbb{Z}$; $j \in \mathbb{Z}$.

Let us consider 3 states (of cells) to represent a graph – n (node), e (edge) and b (blank). That is, $\Sigma = \{n, e, b\}$. To represent a node, the state of the center cell of a block is to be n. In this work, maximum degree of a node is 4. So, the states of corner cells are set as b, and the rest 4 cells can be in state e. Fig. 3 shows two example cases.

Let us consider the graph of Fig. 1. The configuration corresponding to this graph is shown in Fig. 4. Obviously, a configuration that represents an input graph is in Σ^*

2.3. The computation

Let us consider $G = (V, E)$ be the input grid graph, and $T = (V_T, E_T)$ be a tree where $V_T \subseteq V$ and $E_T \subseteq E$. T is spanning tree of G if $V_T = V$. In the proposed scheme, the CA develops a forest of trees, and the trees grow in the forest by joining the individuals; and finally the forest contains a single tree which is the spanning tree.

Let F be the set of all trees, that is, the forest. Initially, F is set of null trees formed by all the nodes of G. During the evolution of CA,