Simulation, Visualization and Experimental analysis for Population Protocols and Network Constructor in 2-Dimensional Case

Author:	Haoxuan Wang
Supervisor:	Dr. Othon Michail
	Dr. Vitaliy Kurlin

May 1, 2018

Declaration of academic integrity

I confirm that I have read and understood the University's Academic Integrity Policy.

I confirm that I have acted honestly, ethically and professionally in conduct leading to assessment for the programme of study.

I confirm that I have not copied material from another source nor committed plagiarism nor fabricated, falsified or embellished data when completing the attached piece of work.

I confirm that I have not copied material from another source, nor colluded with any other student in the preparation and production of this work.

Date: May 1, 2018

Abstract

The abstract[1] [2] [3]

Contents

1 Introduction

1.1 Aims and Objectives

The project aimed to study general population protocols [1] and its two derived model, network constructor [2] and terminating grid network constructor [3] (specifically for grid network construction). It also attempted to experimentally simulate, visualise and compare these protocols via building the simulator and visualizer.

1.2 The challenges in the project

1.2.1 Heterogeneous for different types of Models

The theoretical models involved in three main different models initially originated in population protocols. These three models share inherently common points but there are also some conceptual differences in between them. For instance, the network constructor [2] and terminating grid network constructor [3] involves state of connections in between two nodes while the original population protocol does not. The node of terminating grid network constructor has its complexity structurally compared with the other two types of model.

1.2.2 Heterogeneous for different types of Protocols

The protocols discussed in the related papers [1, 2, 3] involves many different protocols. The protocols may totally different on many characteristics, such as their different computational ability, different ending in either or termination, computation target. These differences between protocol to protocol may lead the simulator and visualizer hardly to be developed and fully tested.

1.2.3 Human factor: Lacking Experience for model visualization

Prior to this project, the author has totally no experiences on model simulation and also no knowledge on what the related code library will be involved. Learning may take more time than its expected.

1.2.4 The Programme

The final programme contains an UI with an fix-sized area to illustrate the process of a particular protocol. The state of elements* In addition, it contains a information panel contains some related information with regard of the population itself, including:

- Number of nodes
- Number of nodes distinguished in different status
- Number of selections for pairs of nodes† that scheduler had took
- Number of effective interactions the population executed

*"Elements" refers nodes in general population protocol, but also includes edge if the protocol involves edge states.

†may also include pair of ports for terminating grid network constructor

Additionally, it provides a set of parameters' settings regarding the initial configuration for the protocol and the population to be simulated, which includes:

- The number of nodes included in the simulation
- The initial state for the nodes[‡]
- The protocol type (and also different sets of transition rules for the protocol)
- Option on whether to use fast-forward simulation method for initially n times selection from scheduler, and the value of n if the option is enabled. A fast-forward simulation executed in the model but does not present in viewer so it will faster than normal the case that does not enable this option.

1.2.5 Evoluation of the project

In general, the simulator successfully implemented a series of different protocols for the three model mentioned above.

UI The UI functions of simulator is verified through a large number of different population simulations. This ensures the UI functions work as it expected in design stage. These experiments on theoretical model may also be asserted the correctness of model through the output configuration of these simulations.

Model The model partition of the simulator developed through Testing driven development method, which indicates the testing suits according to the specification had been written before any model code starts.

2 Background

2.1 Brief Introduction to Population Protocols [1, 4]

Population protocols are theoretical models for distributed computation. The model contains a collection of indistinguishable agents. They (i.e. the agents) carry out computation tasks through directly pair-wised interactions. The interaction pattern of agents is unpredictable from perspective of agents themselves but is controlled through an adversary scheduler with fairness constraints.

Formal Definition A protocol can be formally defined as:

- Q, a finite set of passible states for an agent,
- Σ, a finite set of input alphabet,
- Y, a finite set of output range,

[‡]The state of edge for network constructors (i.e. network constructor and terminating grid network constructor) should be always "0" (i.e. inactivated) at initial, so it is omitted here.

- $\iota = \Sigma \to Q$, is an input map from $\sum$ to Q, hence $\iota(\sigma)$ represents the initial state whose input is $\sigma \in \Sigma$
- $\omega = Q \to Y$, is an output map from Q to Y, and
- $\delta \subseteq Q^4$, a transition relation describes how pairs of agents can interact.

A *configuration* for a population is a vector of all the agents' states. Because agents with a same state are indistinguishable with each other, each configuration could also an unordered multiset of states. It can be represented as C.

At any point of the discrete time, the interaction is unpredictable from the perspective of the agents in the population and the population itself. The interactions at any time are decided by a adversary scheduler necessarily enforced with *fairness* condition. The fairness condition means that the scheduler cannot avoid a possible step forever. Formally, it means if a infinitely often configuration C and $C \to C^{'}$, the $C^{'}$ must also appear infinitely of in the execution.

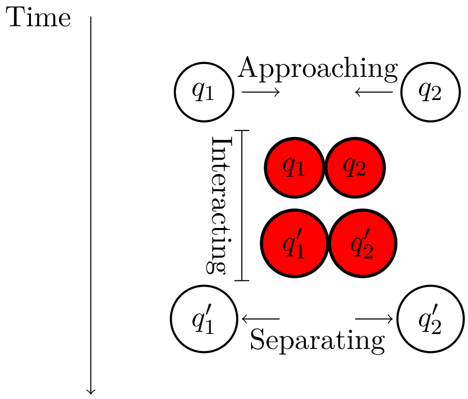

Figure 1: A typical (simple) interaction in population protocol

The current implementation of this work assumes a more strict scheduler called random scheduler, which resulting uniform random interactions (i.e. at each step, it presents equally possibility for every pair of agents to interact.) Essentially, a random scheduler is a "fair" scheduler, but a "fair" scheduler is not necessarily a random scheduler. The assumption actually increases the power of the model because it allows a leader agent detects the absence of agents in a particular state after a long enough waiting. Note this is only for simplifying the current implementation and possibly extends to other fair schedulers that is not a random scheduler in the future.

2.2 From Population Protocols to Network Constructor [2]

A variance of population protocol is called "network constructor" [2]. It involves the further elements for population protocol called "edge", which is the connection in between any two agents (or "processes"). In general, the connection is similar to the agents, for the characteristic that they all only have a finite number of states. For instance, if a connection have $k + 1$ states, then state 0 represents the connection does not exist and for state $i \in \{1, 2, 3, ..., k\}$ shows the strength for a particular connection.

In the following paragraph and the currently implemented simulator, we solely consider the simplest case containing only *on* and off two states for edges. A connection is said to be in *on* or *active* state, if at any discrete time, the connection exists in between two particular nodes; otherwise if the connection is stated as in off or *inactive* state. At initial, all edge are in off state, which means that there is no connection in a given population at discrete time 0. The state for an edge may or may not change through an interaction in between two nodes.

The network constructor also follows an adversary scheduler with "fairness" constraint. This is exactly same as the paragraphs mentioned the previous section 2.1.

Formal Definition Formally, a network constructor can be defined as:

- Q, a finite set of passible states for an agent,
- Q_{out}, a finite set of output range,
- $q_0 \in Q$, is initial state of node,
- $\delta : Q \times Q \times \{0,1\} \to Q \times Q \times \{0,1\}$, a transition function, where the $\{0,1\}$ is the states for edges with initial value 0.

The main target of this model concentrates on network construction rather than specific function computation. Notice that a transition can be either *effective* or *ineffective*. Define $\delta(a,b,c) = (a^{'},b^{'},c^{'})$ as a transition function (Here, a,b are "node-state" whereas c is state for edge.) as in the formal definition, $\delta_1(a,b,c) = a^{'}$, $\delta_2(a,b,c) = b^{'}$ (representing two nodes states' change) and $\delta_3(a,b,c) = c^{'}$ (representing edge state change), the transition $(a,b,c) \to (a^{'},b^{'},c^{'})$ is called effective if at least one $x \in \{a,b,c\} \neq x^{'}$; otherwise it is called ineffective.

Name the set of nodes (or "distributed processes" under this context) V_I and the set of pairs of nodes as E_I. A configuration C is a mapping $V_I \cup E_I \to Q \cup \{0,1\}$ determines the states of nodes and edges in the population at any discrete time. The output of configuration C is defined as the graph $G(C) = (V,E)$ where $V = \{u \in V_I : C(u) \in Q_{out}\}$ and $E = \{uv : u,v \in V, u \neq v$, and $C(uv) = 1\}$.

2.3 Grid Terminating Network Constructor [3]

The paper [3] presents a different automata but similar to network constructor. The node in this model has a fixed number of ports with it. In the 2-dimensional case, it will have 4 distinguished ports p_y, p_x, p_{-y} and p_{-x}, may simply donated as u, r, d and l, respectively. The ports which neighbour with each other are also perpendicular to each other, forming local axes. Hence, $u \perp r$, $r \perp d$, $d \perp l$, and $l \perp u$. The coordinates are only for local purposes and do not necessarily represent the actual orientation of a node. A connection (or edge) only can be built through pairs of ports, which is different from the previous model (network constructor).

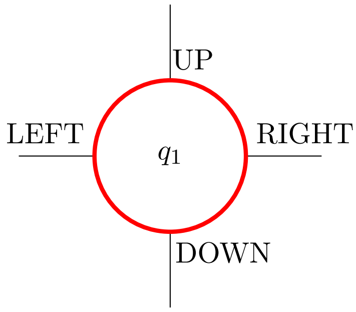

Figure 2: Structure of a node in Grid Terminating Network Constructor model

Formal Definition Formally, a terminating grid network constructor in 2-D can be defined as:

- Q, a finite set of passible states for an agent,
- Q_{out}, a finite set of output range,
- $q_0 \in Q$, is initial state of node
- $\delta : (Q \times P) \times (Q \times P) \times \{0, 1\} \to Q \times Q \times \{0, 1\}$, a transition function, where the $\{0, 1\}$ is the states for edges with initial value 0.
- When required, there will be also a special initial leader-state $L_0 \in Q$ defined.

A transition can be either *effective* or *ineffective*. Define $\delta((a, p_1), (b, p_2), c) = (a^{'}, b^{'}, c^{'})$ as a transition function as in the formal definition, it is called effective if at least one $x \in \{a, b, c\} \neq x^{'}$; otherwise it is called ineffective.

Every configuration C forms a set of shapes $G[A(C)]$, where a shape means that the network induced by active edges of C. Given the geometrical restrictions (i.e. the connection are made at unit distance and are perpendicular whenever they correspond to consecutive ports of a node), not all possible $A(C)$ are valid. $A(C)$ is valid if any connected component defined by it is a subnetwork of 2D grid network with unit distances. A valid $A(C)$ at time $t - 1$ also restricts the possible valid $A(C)$ at time $t + k$, where $k \in Integer$ and $k \geq 0$.

The paper [3] also defines a set of shapes $G_{out}(C) = \{V_s, E_s\}$ as output of a configuration where $V_s = \{u \in V : C_v(u) \in Q_{out}\}$ and $E_s = A(C) \cap \{(v_1, p_1) : v_1 \neq v_2 \in V_s; p_1, p_2 \in P\}$. Less formally, the output shapes of a configuration contains all nodes in output states and the active edges in between them. This model focuses on terminating protocols so it assumes $Q_{out} \subseteq Q_{halt} \subseteq Q$. All rules containing $q_{halt} \in Q_{halt}$ is ineffective.

2.4 Vector transformation: Handle coordinate rotations in 2-dimension

As mentioned in the previous section 2.3, Grid Terminating Network Constructor enforces perpendicular ports and geometrical restrictions. This means that handling rotation in some cases becomes necessary. This section will cover some basic but related vector mathematics. The detailed algorithms will be discussed later on, in "Realisation" section (**??**).

2.4.1 2D centred coordinate rotation, with origin point as the centre

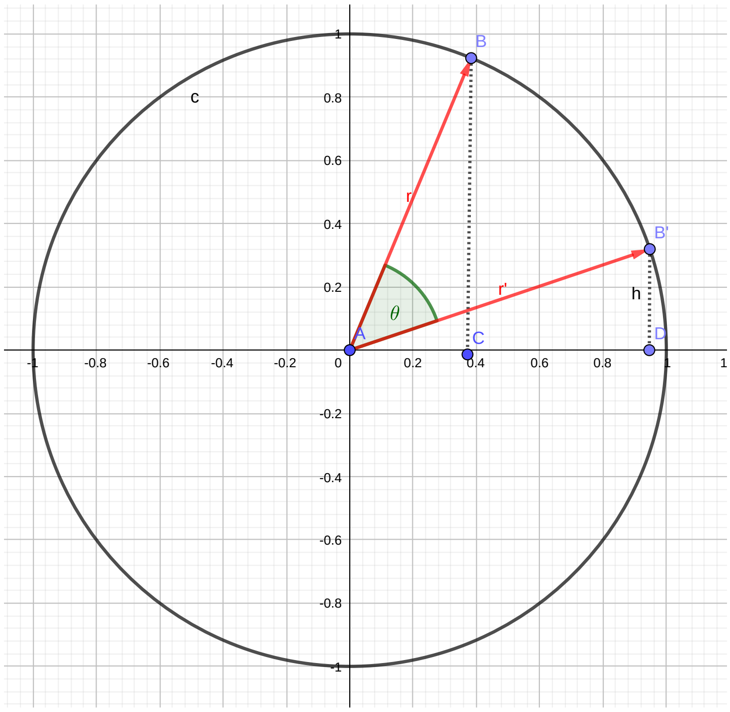

Figure 3: Vector $\vec{r}$ and $\vec{r'}$ in unit circle

Observe the figure 3 , it has two same-length vectors, $\vec{r}$ and $\vec{r'}$ in its Cartesian coordinate and their destination point located at the unit circle. The point A is the origin point with coordinate $(0, 0)$. Suppose it has already known for the coordinate of B is (x, y), and the θ angle. The question is that what the coordinate of $B^{'}$ (representing in $(x^{'}, y^{'})$) is, after the first vector $\vec{r}$ rotates the given angle θ and becomes $\vec{r'}$. Note that, the "positive" rotation direction in current context and the implementation of the simulator is defined as "clock-wised".

To solve this, the following equation can be concluded:

$$\begin{bmatrix} x^{'} \\ y^{'} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{1}$$

A brief proof can be found in appendix partition below and it also can be found in [5] with a more precise and general proof.

2.4.2 2D centred coordinate rotation, with any points as the centre

Given the equation 1 mentioned in the previous section, it can be easily deduced that the rotation equation for any centred points c with coordinate (c_x, c_y) through panning c to origin point with coordinate $(0, 0)$, carry out rotation and panning back to its original coordinate (c_x, c_y). Suppose the target point that is required to be rotated p (with coordinate (x, y)), it then has :

$$\begin{bmatrix} x^{'} \\ y^{'} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} c_x \\ c_y \end{bmatrix} \right) + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{2}$$

2.4.3 Affine transformation form

After a series of transformation, the equation 2 becomes the following form:

$$\begin{bmatrix} x^{'} \\ y^{'} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c_x(1-\cos\theta) - c_y\sin\theta \\ c_y(1-\cos\theta) + c_x\sin\theta \end{bmatrix} \tag{3}$$

Let $\mathbf{M} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} c_x(1-\cos\theta) - c_y\sin\theta \\ c_y(1-\cos\theta) + c_x\sin\theta \end{bmatrix}$, then $\begin{bmatrix} x^{'} \\ y^{'} \end{bmatrix} = \mathbf{M}\begin{bmatrix} x \\ y \end{bmatrix} + \mathbf{B}$, where it forms an *affine transformation* [6]. $\mathbf{M}$ and $\mathbf{B}$ can be calculated directly after knowing the rotation centre (c_x, c_y) and the rotation agle θ, and therefore the calculated $\mathbf{M}$ and $\mathbf{B}$ can be applied for multiple target coordinates (those coordinates required to be rotated). This simplifies the calculation for a large batch of centred rotation with multiple target coordinates.

Bibliography

[1] J. Aspnes and E. Ruppert, "An introduction to population protocols," *Bulletin of the European Association for Theoretical Computer Science*, vol. 93, pp. 98–117, Oct. 2007.

[2] O. Michail and P. G. Spirakis, "Simple and efficient local codes for distributed stable network construction," *Distributed Computing*, vol. 29, no. 3, pp. 207–237, 2016. [Online]. Available: https://link.springer.com/article/10.1007/s00446-015-0257-4

[3] O. Michail, "Terminating distributed construction of shapes and patterns in a fair solution of automata," *Distributed Computing*, pp. 1–23, 2017. [Online]. Available: http://link.springer.com/article/10.1007/s00446-017-0309-z

[4] O. Michail, I. Chatzigiannakis, and P. G. Spirakis, *New Models for Population Protocols*, ser. N. A. Lynch (Ed), Synthesis Lectures on Distributed Computing Theory, N. A. Lynch, Ed. Morgan & Claypool, 2011.

[5] S. Widnall, "Lecture l3 - vectors, matrices and coordinate transformations," Fall 2009. [Online]. Available: https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-07-dynamics-fall-2009/lecture-notes/MIT16_07F09_Lec03.pdf Accessed: 2018-04-30.

[6] E. Weisstein, "Affine transformation." [Online]. Available: http://mathworld.wolfram.com/AffineTransformation.html Accessed: 2018-05-01.

A Brief Proof for equation (1)

For convenience, here name $\angle CAB$ as α in figure 3. Then it has

$$|AD| = |\vec{r'}| \cos(\alpha - \theta)$$

$$|B'D| = |\vec{r'}| \sin(\alpha - \theta)$$

By applying angle sum identities,

$$|AD| = |\vec{r'}|(\cos(\alpha)\cos(-\theta) - \sin(\alpha)\sin(-\theta))$$

$$|B'D| = |\vec{r'}|(\sin(\alpha)\cos(-\theta) + \sin(-\theta)\cos(\alpha))$$

Because $|r'| = |r|$, $\sin(\theta) = -\sin(-\theta)$ for $\theta \in [0, \pi/2)$ and $\cos(\theta) = \cos(-\theta)$ for $\theta \in [0, \pi/2)$,

$$|AD| = |\vec{r}|(\cos(\alpha)\cos(\theta) + \sin(\alpha)\sin(\theta))$$

$$|B'D| = |\vec{r}|(\sin(\alpha)\cos(\theta) - \sin(\theta)\cos(\alpha))$$

Then,

$$|AD| = (|\vec{r}|\cos(\alpha))\cos(\theta) + (|\vec{r}|\sin(\alpha))\sin(\theta)$$
$$|B'D| = (|\vec{r}|\sin(\alpha))\cos(\theta) - (|\vec{r}|\cos(\alpha))\sin(\theta)$$

Hence,

$$|AD| = |AC|\cos(\theta) + |BC|\sin(\theta)$$
$$|B'D| = |BC|\cos(\theta) - |AC|\sin(\theta)$$

Finally,

$$|x'| = |x|\cos(\theta) + |y|\sin(\theta)$$
$$|y'| = |x|(-\sin(\theta)) + |y|\cos(\theta)$$

For $\theta \in [0, \pi/2)$, the equations can be written simply as

$$x' = x\cos\theta + y\sin\theta$$

$$y' = x(-\sin\theta) + y\cos\theta$$

which can be easily extended to $\theta \in [0, 2\pi + 2k\pi)$ via discussion under different cases, given that $k \in Integer$.

The two equations can be written in matrix form :

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4}$$

B Brief Proof for equation (3)

The equation 2 can be transformed to:

$$
\begin{aligned}
\begin{bmatrix} x^{'} \\ y^{'} \end{bmatrix} &= \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} c_x \\ c_y \end{bmatrix} \right) + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \\
&= \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} c_x \\ c_y \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \\
&= \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} c_x \cos\theta + c_y \sin\theta \\ -c_x \sin\theta + c_y \cos\theta \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \\
&= \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c_x(1-\cos\theta) - c_y \sin\theta \\ c_y(1-\cos\theta) + c_x \sin\theta \end{bmatrix}
\end{aligned}
$$