Stably Decidable Graph Languages by Mediated Population Protocols*,**

Ioannis Chatzigiannakis[1,2], Othon Michail[1,2], and Paul G. Spirakis[1,2]

[1] Research Academic Computer Technology Institute (RACTI)
[2] Computer Engineering and Informatics Department (CEID), University of Patras, 26500, Patras, Greece.
Email: {ichatz, michailo, spirakis}@cti.gr

Abstract. We work on an extension of the Population Protocol model of Angluin *et al.* that allows edges of the communication graph, G, to have *states* that belong to a constant size set. In this extension, the so called Mediated Population Protocol model (MPP), both *uniformity* and *anonymity* are preserved. We study here a simplified version of MPP in order to capture MPP's ability to stably compute *graph properties*. To understand properties of the communication graph is an important step in almost any distributed system. We prove that any graph property is not computable if we allow disconnected communication graphs. As a result, we focus on studying (at least) *weakly connected* communication graphs only and give several examples of computable properties in this case. To do so, we also prove that the class of computable properties is *closed under complement*, *union* and *intersection* operations. Node and edge parity, bounded out-degree by a constant, existence of a node with more incoming than outgoing neighbors, and existence of some directed path of length at least $k = \mathcal{O}(1)$ are some examples of properties whose computability is proven. Finally, we prove the existence of symmetry in two specific communication graphs and, by exploiting this, we prove that there exists no protocol, whose states eventually stabilize, to determine whether G contains some directed cycle of length 2.

1 Introduction

Most recent advances in microprocessor, wireless communication and sensor/actuator-technologies envision a whole new era of computing, popularly referred to as pervasive computing. Autonomous, ad-hoc networked, wirelessly communicating and *spontaneously interacting* computing devices of *small size* appearing in *great number*, and embedded into environments, appliances and objects of everyday use will deliver services adapted to the person, the time, the place, or the context of their use. The nature and appearance of devices will change to be hidden in the fabric of everyday life and will be augmenting everyday environments to form a pervasive computing landscape.

* This work has been partially supported by the ICT Programme of the European Union under contract number ICT-2008-215270 (FRONTS).

** A preliminary brief version of this work has appeared in [12].

In a seminal paper [2], Angluin *et al.* introduced the notion of a computation by a population to model such systems in which individual agents are extremely limited and can be represented as finite-state machines. In their model, finite-state, and complex behavior of the system as a whole emerges from simple rules governing pairwise interaction of the agents. The computation is carried out by a collection of agents, each of which receives a piece of the input. These agents move around and information can be exchanged between two agents whenever they come sufficiently close to each other. The most important innovations of the model are inarguably the *constant memory* constraint imposed to the agents and the *nondeterminism* inherent to the interaction pattern. These assumptions provide us with a concrete and realistic model for future systems.

A population protocol $\mathcal{A}$ consists of finite *input and output alphabets* X and Y, a finite set of *states* Q, an *input function* $I : X \to Q$ mapping inputs to states, an *output function* $O : Q \to Y$ mapping states to outputs, and a *transition function* $\delta : Q \times Q \to Q \times Q$. The model assumes a population of $n \equiv |V|$ agents and a protocol runs on a (simple) directed communication graph $G = (V, E)$. An *agent* has a *memory of constant size* (i.e., $\mathcal{O}(1)$ bits) and a *control unit* that updates the agent states according to the interactions taking place; the input and output of the agents may represent a *sensor* and/or an *actuator*. Each protocol has a constant-size description, i.e., independent of n, that can be stored in each agent of the population. This gives to population protocols two important properties: *uniformity* and *anonymity*; the transition function treats all agents in the same way and there is no room in the state of an agent to store a unique identifier.

The initial goal of the model was to study the *computational limitations* of cooperative systems consisting of many limited devices (agents), imposed to *passive* (but *fair*) communication by some *scheduler*. Much work showed that there exists an exact characterization of the computable predicates: they are precisely the *semilinear predicates* or equivalently the predicates definable by first-order logical formulas in *Presburger arithmetic* [2, 3, 5–7]. Some recent work has concentrated on performance, supported by a random scheduling assumption [4]. [10] proposed a generic definition of probabilistic schedulers and a collection of new fair schedulers, and revealed the need for the protocols to adapt when natural modifications of the mobility pattern occur. [9, 14] considered a huge population hypothesis (population going to infinity), and studied the dynamics, stability and computational power of probabilistic population protocols by exploiting the tools of continuous nonlinear dynamics. In [9] it was also proven that there is a strong relationship between classical finite population protocols and models given by ordinary differential equations.

There exist a few extensions of the basic model in the relevant literature to more accurately reflect the requirements of practical systems. In [1] they studied what properties of restricted communication graphs are stably computable, gave protocols for some of them, and proposed the model extension with *stabilizing inputs*. The results of [5] show that again the semilinear predicates are all that can be computed by this model. Finally, some works incorporated agent

failures [15] and gave to some agents slightly increased computational power [8] (heterogeneous systems). For an excellent introduction see [7].

Very recently, a natural variation of the basic model was proposed [13], where the *interactions* of the agents can be characterized by a state of constant size. Essentially the model is augmented to include a *Mediator*, i.e., a global storage capable of storing very limited information for each communication link (the state of the link). When pairs of agents interact, they can read and update the state of the link. Interestingly, although anonymity and uniformity are preserved, *the presence of a mediator allows us to obtain significant more computational power*; we can build systems with the ability of computing subgraphs and solve optimization problems concerning the communication graph. In [13] it was shown that the new model is capable of computing non-semilinear predicates and that any stably computable predicate belongs to $NSPACE(m)$, where m denotes the number of edges of the interaction graph. The latter inclusion was proven in [11] to hold with equality. Finally, [12] constitutes a preliminary brief version of this work.

In this work (as [1] did for population protocols), we consider a simplification of the above model in order to explore one of its most important capabilities: The computability of graph properties. To understand properties of the communication graph is an important step in almost any distributed system. In particular, we temporarily disregard the input notion of the population and assume that all agents simply start from a unique initial state (and the same holds for the edges). We are interested in protocols of the new model, that we call the GDMPP model, that when executed on any communication graph G, after a finite number of steps stabilize to a configuration where all agents give 1 as output if G belongs to a graph language L, and 0 otherwise. This is motivated by the idea of having protocols that eventually accept all communication graphs (on which they run) that satisfy a specific property, and eventually reject all remaining communication graphs. The reason for proposing a simplified model is that it enables us to study what graph properties are stably computable by the MPP model without the need to keep in mind its remaining parameters.

2 Our Results - Roadmap

In Section 3, we give a formal definition of the GDMPP model. In Section 4, we focus on weakly connected communication graphs. We prove that the class of computable graph properties is closed under complement, union, and intersection operations. Node and edge parity, bounded out-degree by a constant, existence of a node with more incoming than outgoing neighbors, and existence of some directed path of length at least $k = \mathcal{O}(1)$ are some examples of properties whose computability is proven. Moreover, the existence of symmetry in two specific communication graphs is revealed and is exploited to prove that there exists no GDMPP, whose states eventually stabilize, to compute the graph language $2C$, consisting of all weakly connected communication graphs that contain some 2-cycle. We leave as an interesting open problem whether $2C$ isn't com-

putable in the general case. In Section 5, we focus on the universe of all possible communication graphs, containing also the disconnected ones. In this case (see Theorem 9) we prove that any nontrivial graph language (we exclude both the empty language and its complement) is not computable by the GDMPP model. As an interesting corollary we get that GDMPP cannot compute connectivity (Corollary 1). Finally, in Section 6 we discuss some future research directions.

3 The model

A *Graph Decision Mediated Population Protocol* (GDMPP) $\mathcal{A}$ consists of a *binary output alphabet* $Y = \{0, 1\}$, a finite set of *agent states* Q, an *output function* $O : Q \to Y$ mapping agent states to outputs, a finite set of *edge states* S, and a *transition function* $\delta : Q \times Q \times S \to Q \times Q \times S$. If $\delta(a, b, s) = (a', b', s')$ we call $(a, b, s) \to (a', b', s')$ a *transition*, and we define $\delta_1(a, b, s) = a'$, $\delta_2(a, b, s) = b'$ and $\delta_3(a, b, s) = s'$.

We assume that all agents are initially in an *initial agent state* $q_0 \in Q$ and all edges in an *initial edge state* $s_0 \in S$. A *graph universe* (or *graph family*) is any set of communication graphs. We denote by $\mathcal{H}$ the graph universe consisting of all possible communication graphs of any finite number of nodes greater or equal to 2 (we do not allow the empty graph, the graph with a unique node and we neither allow infinite graphs) and by $\mathcal{G}$ the subset of $\mathcal{H}$ containing the weakly connected ones. All the following definitions hold w.r.t. some fixed graph universe $\mathcal{U}$. A *graph language* L is a subset of $\mathcal{U}$ containing communication graphs that possibly share some common property., e.g. $L = \{G \in \mathcal{U} \mid G$ contains a directed hamiltonian path$\}$. A graph language L is said to be *nontrivial* if $L \neq \emptyset$ and $L \neq \mathcal{H}$.

A GDMPP runs on a graph $G = (V, E)$, where V is a population of $|V| = n$ agents and E is an irreflexive binary relation on V. The graph on which the protocol runs is considered as the *input graph* of the protocol. The input graph of a GDMPP may be any $G \in \mathcal{U}$.

A *network configuration* (or simply *configuration*) is a mapping $C : V \cup E \to Q \cup S$ specifying the agent state of each agent in the population and the edge state of each edge in the communication graph. Let C and C' be network configurations, and let u, v be distinct agents. We say that C goes to C' via encounter $e = (u, v)$, denoted $C \xrightarrow{e} C'$, if $C'(u) = \delta_1(C(u), C(v), C(e))$, $C'(v) = \delta_2(C(u), C(v), C(e))$, $C'(e) = \delta_3(C(u), C(v), C(e))$, and $C'(z) = C(z)$ for all $z \in (V - \{u, v\}) \cup (E - \{e\})$. We say that C can go to C' in one step, denoted $C \to C'$, if $C \xrightarrow{e} C'$ for some encounter $e \in E$. We write $C \xrightarrow{*} C'$ if there is a sequence of configurations $C = C_0, C_1, \ldots, C_t = C'$, such that $C_i \to C_{i+1}$ for all i, $0 \leq i < t$, in which case we say that C' is *reachable* from C.

An *execution* is a finite or infinite sequence of network configurations C_0, C_1, $C_2, \ldots$, where C_0 is an initial configuration and $C_i \to C_{i+1}$, for all $i \geq 0$. An infinite execution is *fair* if for every pair of network configurations C and C' such that $C \to C'$, if C occurs infinitely often in the execution, then so does C'. A *computation* is an infinite fair execution.

At any point during the execution of a GDMPP, each agent's state determines its output at that time. The output of any agent u under configuration C is $O(C(u))$. Note also that the *code* of any GDMPP is of *constant size* (independent of the population size) and, thus, can be stored in each agent (device) of the population.

Definition 1. *Let L be a graph language consisting of all $G \in \mathcal{U}$ for which, in any computation of a GDMPP $\mathcal{A}$ on G, all agents eventually output* 1*. Then L is* the language stably recognized by $\mathcal{A}$*. A graph language is said to be* stably recognizable *by the GDMPP model (also called $GDMPP$-*recognizable*) if some GDMPP stably recognizes it.*

Thus, any protocol *stably recognizes* the graph language consisting of those graphs on which the protocol always answers "accept", i.e. eventually all agents output the value 1 (possibly the empty language).

Definition 2. *We say that a GDMPP $\mathcal{A}$* stably decides *a graph language $L \subseteq \mathcal{U}$ (or equivalently a predicate $p_L : \mathcal{U} \to \{0,1\}$ defined as $p_L(G) = 1$ iff $G \in L$) if for any $G \in \mathcal{U}$ and any computation of $\mathcal{A}$ on G, all agents eventually output* 1 *if $G \in L$ and all agents eventually output* 0 *if $G \notin L$. A graph language is said to be* stably decidable *by the GDMPP model (also called $GDMPP$-*decidable*) if some GDMPP $\mathcal{A}$ stably decides it.*

A GDMPP $\mathcal{A}$ has *stabilizing states* if in any computation of $\mathcal{A}$, after a finite number of interactions, the states of all agents stop changing.

In some cases, a protocol, instead of stably deciding a language L, may provide some different sort of guarantee. For example, whenever runs on some $G \in L$, it may forever remain to configurations where at least one agent is in state a, and when $G' \notin L$ no agent will remain in state a. To formalize this, we say that a GDMPP $\mathcal{A}$ *guarantees* $t : Q^* \to \{0,1\}$ *w.r.t.* $L \subseteq \mathcal{U}$ if, for any $G \in \mathcal{U}$, any computation of $\mathcal{A}$ on G eventually reaches a configuration C, s.t. for all C', where $C \xrightarrow{*} C'$, it holds that $t(C') = t(C) = 1$ if $G \in L$ and $t(C') = t(C) = 0$, otherwise.[3]

4 Weakly Connected Graphs

In this section, we study an interesting case in which the graph universe is not allowed to contain disconnected graphs. Thus, here the graph universe is $\mathcal{G}$ and, thus, a graph language can only be a subset of $\mathcal{G}$. The main reason for selecting this specific universe for devising our protocols is that, if we also allow disconnected graphs, then, as we shall see, it can be proven that no graph language is stably decidable.

[3] By assuming an ordering on V we can define configurations as strings from Q^*.

4.1 Decidable Graph Languages

Our goal is to show the stable decidability of some interesting graph languages by providing protocols for them and proving their correctness. To begin, we prove some closure results to obtain a useful tool for our purpose.

Theorem 1. *The class of stably decidable graph languages is closed under complement, union and intersection operations.*

Proof. First we show that for any stably decidable graph language L its complement $\overline{L}$ is also stably decidable. From definition of stable decidability there exists GDMPP $\mathcal{A}_L$ that decides L. Thus, for any $G \in \mathcal{G}$ and any computation of $\mathcal{A}_L$ on G all agents eventually output 1 if $G \in L$ and 0 otherwise. By complementing the output map $O_{\mathcal{A}}$ of $\mathcal{A}$ we obtain a new protocol $\overline{\mathcal{A}}$, with output map defined as $O_{\overline{\mathcal{A}}}(q) = 1$ iff $O_{\mathcal{A}}(q) = 0$, for all $q \in Q_{\mathcal{A}} = Q_{\overline{\mathcal{A}}}$, whose agents eventually output 1 if $G \notin L$ and 0 otherwise, thus stably deciding $\overline{L}$.

Now we show that for any stably decidable graph languages L_1 and L_2, $L_3 = L_1 \cup L_2$ is also stably decidable. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be GDMPPs that stably decide L_1 and L_2, respectively (we know their existence). We let the two protocols operate in parallel, i.e. we devise a new protocol $\mathcal{A}_3$ whose agent and edge states consist of two components, one for protocol $\mathcal{A}_1$ and one for $\mathcal{A}_2$. Let O_1 and O_2 be the output maps of the two protocols. We define the output map O_3 of $\mathcal{A}_3$ as $O_3(q, q') = 1$ iff at least one of $O_1(q)$ and $O_2(q')$ equals to 1, for all $q \in Q_{\mathcal{A}_1}$ and $q' \in Q_{\mathcal{A}_2}$. If $G \in L_3$ then at least one of the two protocols has eventually all its agent components giving output 1, thus $\mathcal{A}_3$ correctly answers "accept", while if $G \notin L_3$ then both protocols have eventually all their agent components giving output 0, thus $\mathcal{A}_3$ correctly answers "reject". We conclude that $\mathcal{A}_3$ stably decides L_3 which proves that L_3 is stably decidable.

By defining the output map O_3 of $\mathcal{A}_3$ as $O_3(q, q') = 1$ iff $O_1(q) = O_2(q') = 1$, for all $q \in Q_{\mathcal{A}_1}$ and $q' \in Q_{\mathcal{A}_2}$, and making the same composition as before, it is easy to see that in this case $\mathcal{A}_3$ stably decides the intersection of L_1 and L_2. □

In some cases it is not easy to devise a protocol that respects the *predicate output convention* (the predicate output convention was defined in [2] and simply requires all agents to eventually agree on the correct output value). In such cases, we can use the following variation of the Composition Theorem (Theorem 6) of [13] that facilitates the proof of existence of GDMPP protocols that stably decide a language.

Theorem 2. *If there exists a GDMPP $\mathcal{A}$ with stabilizing states that w.r.t. to a language L guarantees a semilinear predicate, then L is GDMPP-decidable.*

Proof. Immediate from the proof of the Theorem 6 of [13]. $\mathcal{A}$ can be composed with a provably existing GDMPP $\mathcal{B}$ whose stabilizing inputs are $\mathcal{A}$'s agent states to give a new GDMPP $\mathcal{C}$ that stably decides L w.r.t. the predicate output convention. Note that $\mathcal{B}$ is in fact a GDMPP, since its stabilizing inputs are not real inputs (GDMPPs do not have inputs). It simply updates its state components by taking also into account the eventually stabilizing state components of $\mathcal{A}$. Thus, their composition, $\mathcal{C}$, is also a GDMPP.

Theorem 3. (Node Parity & Edge Parity) *The graph languages* $N_{even} = \{G \in \mathcal{G} \mid |V(G)| \text{ is even}\}$, $N_{odd} = \overline{N}_{even}$, $E_{even} = \{G \in \mathcal{G} \mid |E(G)| \text{ is even}\}$, *and* $E_{odd} = \overline{E}_{even}$ *are stably decidable.*

Theorem 4. (Constant Neighbors - Some Node) *The graph language* $N_k^{out} = \{G \in \mathcal{G} \mid G \text{ has some node with at least } k \text{ outgoing neighbors}\}$ *is stably decidable for any* $k = \mathcal{O}(1)$ *(the same holds for* $\overline{N}_k^{out}$*).*

Proof. Initially all agents are in q_0 and all edges in 0. The set of agent states is $Q = \{q_0, \ldots, q_k\}$ the set of edge states is binary and the output function is defined as $O(q_k) = 1$ and $O(q_i) = 0$ for all $i \in \{0, \ldots, k-1\}$. We now describe the transition function. In any interaction through an edge in state 0, the initiator visits an unvisited outgoing edge, so it marks it by updating the edge's state to 1 and increases its own state index by one, e.g. initially $(q_0, q_0, 0)$ yields $(q_1, q_0, 1)$, and, generally, $(q_i, q_j, 0) \to (q_{i+1}, q_j, 1)$, if $i + 1 < k$ and $j < k$, and $(q_i, q_j, 0) \to (q_k, q_k, 1)$, otherwise. Whenever two agents meet through a marked edge they do nothing, except for the case where only one of them is in the special alert state q_k. If the latter holds, then both go to the alert state, since in this case the protocol has located an agent with at least k outgoing neighbors. To conclude, all agents count their outgoing edges and initially output 0. Iff one of them marks its k-th outgoing edge, both end points of that edge go to an alert state q_k that propagates to the whole population and whose output is 1, indicating that G belongs to N_k^{out}. □

Note that $\overline{N}_k^{out}$ contains all graphs that have no node with at least $k = \mathcal{O}(1)$ outgoing neighbors, in other words, all nodes have fewer than k outgoing edges, which is simply the well-known bounded by k out-degree predicate. The same statement for population protocols appears as Lemma 3 in [1].

Theorem 5. (Constant Neighbors - All Nodes) *The graph language* $K_k^{out} = \{G \in \mathcal{G} \mid \textit{Any node in } G \textit{ has at least } k \textit{ outgoing neighbors}\}$ *is stably decidable for any* $k = \mathcal{O}(1)$ *(the same holds for* $\overline{K}_k^{out}$*).*

Proof. Note, first of all, that another way to think of K_k^{out} is $K_k^{out} = \{G \in \mathcal{G}_{con} \mid \text{No node in } G \text{ has less than } k \text{ outgoing neighbors}\}$, for some $k = \mathcal{O}(1)$. The protocol we describe is similar to the one described in the proof of Theorem 4. The only difference is that when an agent counts its k-th outgoing neighbor as the initiator of an interaction, it goes to the special alert state q_k, but the alert state is not propagated (e.g. the responder of this interaction keeps its state). It follows that eventually any node that has marked at least k outgoing edges will be in the alert state, while any other node that has less than k outgoing edges will be in some state q_i, where $i < k$. Clearly the protocol has stabilizing states and provides the following semilinear guarantee:

- If $G \notin K_k^{out}$ then at least one agent remains in some state q_i, where $i < k$.
- If $G \in K_k^{out}$ no such state remains.

Thus, Theorem 2 applies, implying that there exists some GDMPP stably deciding K_k^{out} w.r.t. the predicate output convention. Thus, both K_k^{out} and $\overline{K}_k^{out}$ are stably decidable and the proof is complete. □

Theorem 6. (Compare Incoming and Outgoing Neighbors) *The graph language $M_{out} = \{G \in \mathcal{G} \mid G$ has some node with more outgoing than incoming neighbors$\}$ is stably decidable (the same holds for $\overline{M}_{out}$).*

Proof. Consider the following protocol: Initially all agents are in state 0 which is the *equality* state. An agent can also be in state 1 which is the *more-outgoing* state. Initially all edges are in state s_0 and S contains also o, i and b, where state o means that the edge has been used by the protocol only as outgoing so far, i means only as incoming and b is for "both". Any agent always remembers if it has seen so far more outgoing edges or the same number of incoming and outgoing edges. So, if it is in equality state and is the initiator in an interaction where the edge has not been used at all (state s_0) or has been used only as an incoming edge (state i), which simply means that only the responder has counted it, then the agent goes to the more-outgoing state and updates the edge accordingly to remember that it has counted it. Similarly, if an agent in the more-outgoing state is the responder of an interaction and the edge is in one of the states s_0 or o, then it goes to the equality state and updates the edge accordingly. If we view the interaction from the edge's perspective, then we distinguish the following cases:

1. The edge is in state s_0. Both the initiator and the responder can use it. If only the initiator uses it (both initiator and responder in equality state), then the edge goes to state o. If only the responder uses it (both in more-outgoing state) then the edge goes to state i. If both use it (initiator in equality and responder in more-outgoing) then it goes to state b. If no one uses it it remains in s_0.
2. The edge is in state o. The initiator cannot use it, since it has already counted it. If the responder is in more-outgoing state, then it counts it, thus the edge goes to state b. If, instead, it is in the equality state, the edge remains in state o.
3. The edge is in state i. The responder cannot use it. If the initiator is in equality state, then it counts it, thus the edge goes to state b. If, instead, it is in the more-outgoing state, the edge remains in state i.
4. The edge is in state b. Both the initiator and the responder have used it, thus nothing happens.

The equality state outputs 0 and the more-outgoing state outputs 1. If there exists a node with more outgoing edges, then it will eventually remain in the more-outgoing state giving 1 as output, otherwise all nodes will eventually remain in equality state (although some of them may have more incoming edges), thus giving 0 as output. Computing that at least one more-outgoing state eventually remains is semilinear and the protocol, obviously, has stabilizing states, thus Theorem 2 applies and we conclude that M_{out} is stably decidable. Closure under complement implies that $\overline{M}_{out}$ is also stably decidable. □

Remark 1. By symmetry, the corresponding languages N_k^{in}, $\overline{N}_k^{in}$, K_k^{in} and $\overline{K}_k^{in}$ concerning incoming neighbors, $M_{in} = \{G \in \mathcal{G} \mid G$ has some node with more incoming than outgoing neighbors$\}$ and $\overline{M}_{in}$ are also stably decidable.

Theorem 7. (Directed Path of Constant Length) *The graph language $P_k = \{G \in \mathcal{G} \mid G$ has at least one directed path of at least k edges$\}$ is stably decidable for any $k = \mathcal{O}(1)$ (the same holds for $\overline{P}_k$).*

Proof. If $k = 1$ the protocol that stably decides P_1 is trivial, since it accepts iff at least one interaction happens (in fact it can always accept since all graphs have at least two nodes and they are weakly connected, and thus $P_1 = \mathcal{G}_{con}$). We give a general protocol, $DirPath$ (Protocol 1), that stably decides P_k for any constant $k > 1$.

Protocol 1 *DirPath*

1: $Q = \{q_0, q_1, 1, \ldots, k\}$, $S = \{0, 1\}$,
2: $O(k) = 1$, $O(q) = 0$, for all $q \in Q - \{k\}$,
3: δ:

$$
\begin{aligned}
(q_0, q_0, 0) &\to (q_1, 1, 1) \\
(q_1, x, 1) &\to (x-1, q_0, 0), \text{ if } x \geq 2 \\
&\to (q_0, q_0, 0), \text{ if } x = 1 \\
(x, q_0, 0) &\to (q_1, x+1, 1), \text{ if } x + 1 < k \\
&\to (k, k, 0), \text{ if } x + 1 = k \\
(k, \cdot, \cdot) &\to (k, k, \cdot) \\
(\cdot, k, \cdot) &\to (k, k, \cdot)
\end{aligned}
$$

Intuitively, the protocol expands non-communicating active paths (they can interact but the corresponding transitions do nothing, that's why they are not appearing in δ). The head of each path counts its length. If the length of an active path ever becomes equal to k, then a state giving 1 as output is propagated. Note that, to avoid getting stuck, the protocol keeps backtracking and even totally releasing the active paths. Fairness condition ensures that if a path of length at least k exists, then $DirPath$ will eventually find it. □

4.2 Non Stably Decidable Languages

Now we are about to prove that a specific graph language cannot be stably decided by GDMPPs with stabilizing states. First we state and prove a useful lemma.

Lemma 1. *For any GDMPP $\mathcal{A}$ and any computation (infinite fair execution) $C_0, C_1, C_2, \ldots$ of $\mathcal{A}$ on G (Figure 1(a)) there exists a computation $C'_0, C'_1, C'_2, \ldots$,*

$C'_i, \ldots$ *of* $\mathcal{A}$ *on* G' *(Figure 1(b)) s.t.*

$$C_i(v_1) = C'_{2i}(u_1) = C'_{2i}(u_3)$$
$$C_i(v_2) = C'_{2i}(u_2) = C'_{2i}(u_4)$$
$$C_i(e_1) = C'_{2i}(t_1) = C'_{2i}(t_3)$$
$$C_i(e_2) = C'_{2i}(t_2) = C'_{2i}(t_4)$$

for any finite $i \geq 0$.

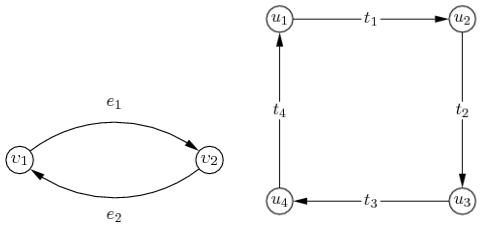
Proof. The proof is by induction on i. We assume that initially all nodes are in q_0 and all edges in s_0 (initial states). So the base case (for $i = 0$) holds trivially. Now we make the following assumption: Whenever the scheduler of $\mathcal{A}$ on G (call it S_1) selects the edge e_1 we assume that the scheduler, S_2, of $\mathcal{A}$ on G' takes two steps; it first selects t_1 and then selects t_3. Whenever S_1 selects the edge e_2, S_2 first selects t_2 and then t_4. Formally, if $C_{i-1} \xrightarrow{e_1} C_i$ then $C'_{2(i-1)} \xrightarrow{t_1} C'_{2i-1} \xrightarrow{t_3} C'_{2i}$ and if $C_{i-1} \xrightarrow{e_2} C_i$ then $C'_{2(i-1)} \xrightarrow{t_2} C'_{2i-1} \xrightarrow{t_4} C'_{2i}$ for every finite $i \geq 1$. Obviously, S_2 is not a fair scheduler so to be able to talk about computation we only require this predetermined behavior to be followed by S_2 for a finite number of steps. After this finite number of steps, S_2 goes on arbitrarily but in a fair manner.

Now assume that all conditions are satisfied for some finite step i (inductive hypothesis). We will prove that the same holds for step $i + 1$ to complete the proof (inductive step). There are two cases:

1. $C_i \xrightarrow{e_1} C_{i+1}$ *(i.e. in step* $i+1$ S_1 *selects the edge* e_1*)*: Then we know that S_2 first selects t_1 and then t_3 (in its corresponding steps $2i{+}1$ and $2i{+}2$). That is, its first transition is $C'_{2i} \xrightarrow{t_1} C'_{2i+1}$ and its second is $C'_{2i+1} \xrightarrow{t_3} C'_{2(i+1)}$. But from the inductive hypothesis we know that $C'_{2i}(u_1) = C_i(v_1)$, $C'_{2i}(u_2) = C_i(v_2)$ and $C'_{2i}(t_1) = C_i(e_1)$ which simply means that interaction e_1 on G has the same effect as interaction t_1 on G' (u_1 has the same state as v_1, u_2 as v_2 and t_1 as e_1). Thus, $C'_{2i+1}(u_1) = C_{i+1}(v_1)$, $C'_{2i+1}(u_2) = C_{i+1}(v_2)$ and $C'_{2i+1}(t_1) = C_{i+1}(e_1)$. Moreover, in this step t_3 and both its endpoints do not change state (since the interaction concerned t_1), thus $C'_{2i+1}(u_3) = C'_{2i}(u_3) = C_i(v_1)$ (the last equation comes from the inductive hypothesis), $C'_{2i+1}(u_4) = C'_{2i}(u_4) = C_i(v_2)$ and $C'_{2i+1}(t_3) = C'_{2i}(t_3) = C_i(e_1)$. When in the next step S_2 selects t_3, t_1 and both its endpoints do not change state, thus $C'_{2(i+1)}(u_1) = C'_{2i+1}(u_1) = C_{i+1}(v_1)$, $C'_{2(i+1)}(u_2) = C'_{2i+1}(u_2) = C_{i+1}(v_2)$ and $C'_{2(i+1)}(t_1) = C'_{2i+1}(t_1) = C_{i+1}(e_1)$. Now let's see what happens to t_3 and its endpoints. Before the interaction the state of u_3 is $C_i(v_1)$, the state of u_4 is $C_i(v_2)$ and the state of t_3 is $C_i(e_1)$, which means that, in $C'_{2(i+1)}$, u_3 has gone to $C_{i+1}(v_1)$, u_4 to $C_{i+1}(v_2)$ and t_3 to $C_{i+1}(e_1)$. Finally, t_2 and t_4 have not participated in any of the two interactions of S_2 and thus they have maintained their states, that is $C'_{2(i+1)}(t_2) = C'_{2i}(t_2) = C_i(e_2) = C_{i+1}(e_2)$ (the last two equations follow from the inductive hypothesis and the fact that, in step $i + 1$, S_1 selects e_1 which means that e_2 maintains its state, respectively), and similarly $C'_{2(i+1)}(t_4) = C_{i+1}(e_2)$.

2. $C_i \xrightarrow{e_2} C_{i+1}$ *(i.e. in step $i+1$ S_1 selects the edge e_2)*: This case is symmetric to the previous one.

□



(a) Graph G (b) Graph G'

Fig. 1. $G \in 2C$ and $G' \notin 2C$.

Let now $\mathcal{A}$ be a GDMPP that stably decides the graph language $2C = \{G \in \mathcal{G} \mid G$ has at least two nodes u, v s.t. both $(u,v),(v,u) \in E(G)$ (in other words, G has at least one 2-cycle)$\}$. So for any computation of $\mathcal{A}$ on G, after finitely many steps, both v_1 and v_2 go to some state that outputs 1, since $G \in 2C$, and do not change their output value in any subsequent step (call the corresponding output stable configuration C_i, where i is finite). But according to Lemma 1 there exists a computation of $\mathcal{A}$ on G' that under configuration C'_{2i} has u_1, u_2, u_3 and u_4 giving output 1. We use this fact to prove the following impossibility result.

Theorem 8. *There exists no GDMPP with stabilizing states to stably decide the graph language $2C = \{G \in \mathcal{G} \mid G$ has at least two nodes u, v s.t. both $(u,v),(v,u) \in E(G)\}$.*

Proof. Let $\mathcal{A}$ be a GDMPP with stablizing states that stably decides $2C$. It follows that when $\mathcal{A}$ runs on G (Figure 1(a)) after a finite number of steps v_1 and v_2 obtain two states w.l.o.g. q_1 and q_2, respectively, that output 1 (since $\mathcal{A}$ stably decides $2C$) and do not change in any subsequent step (since $\mathcal{A}$ has stabilizing states). Assume that at that point e_1 is in s_1 and e_2 in s'_1. Assume also that there exists a subset $S_1 = \{s_1, s_2, \ldots, s_k\}$ of S, of edge states that can be reached by subsequent interactions of the pair (v_1, v_2) and a subset $S_2 = \{s'_1, s'_2, \ldots, s'_l\}$ of S, of edge states that can be reached by subsequent interactions of the pair (v_2, v_1), where k and l are both constants independent of n (note that S_1 and S_2 are not necessarily disjoint). It follows that for all $s_i \in S_1$, $(q_1, q_2, s_i) \rightarrow (q_1, q_2, s_j)$, where $s_j \in S_1$, and for all $s'_i \in S_2$, $(q_2, q_1, s'_i) \rightarrow (q_2, q_1, s'_j)$, where $s'_j \in S_2$. In words, none of these reachable edge states can be responsible for a change in some agent's state. According to Lemma 1 there exists a computation of $\mathcal{A}$ on G' (Figure 1(b)) such that after a finite number of steps u_1, u_3 are in q_1, u_2, u_4 are in q_2, t_1, t_3 are in s_1 and t_2, t_4 are in s'_1. Since $\mathcal{A}$ stably decides $2C$, at some subsequent finite step (after we let the protocol run in a fair manner in G'),

some agent obtains a new state q_3, since if it didn't then all agents would always remain to states q_1 and q_2 that output 1 (but in G' there is no 2-cycle and such a decision is wrong). This must happen through some interaction of the following two forms: (i) (q_1, q_2, s_i), where $s_i \in S_1$ and (ii) (q_2, q_1, s_i'), where $s_i' \in S_2$. But this is a contradiction, since we showed earlier that no such interaction can modify the state of any of its end points. Intuitively, if there exists some way for $\mathcal{A}$ to modify one of q_1 and q_2 in G' then there would also exist some way for $\mathcal{A}$ to modify one of q_1 and q_2 in G, after the system has obtained stabilizing states there, which is an obvious contradiction. □

5 Graphs not even weakly connected

In this section, our universe is $\mathcal{H}$ and, thus, a graph language can only be a subset of $\mathcal{H}$. Any disconnected graph G in $\mathcal{H}$ consists of (weakly or strongly connected) components $G_1, G_2, \ldots, G_t$, where $t \geq 2$ (note also that any component must have at least two nodes, to allow computation).

Lemma 2. *For any nontrivial graph language L, there exists some disconnected graph G in L where at least one component of G does not belong to L or there exists some disconnected graph G' in $\overline{L}$ where at least one component of G' does not belong to $\overline{L}$ (or both).*

Proof. Let L be such a nontrivial graph language and assume that the statement does not hold. Then for any disconnected graph in L, all of its components also belong to L and for any disconnected graph in $\overline{L}$, all of its components also belong to $\overline{L}$. There are two main cases:

1. L contains all connected graphs. But $\overline{L}$ is nontrivial which means that it must contain at least one disconnected graph. We know that for any disconnected graph in $\overline{L}$ all of its connected components belong to $\overline{L}$, but this is a contradiction, since all connected graphs belong to L.
2. L does not contain all connected graphs. There are now two possible subcases:
   (a) L contains at least one connected graph (but not all). This means that $\overline{L}$ contains also at least one connected graph. Let $G' = (V', E')$ be a connected graph from L and $G'' = (V'', E'')$ be a connected graph from $\overline{L}$. The disjoint union of G' and G'', $U = (V' \cup V'', E' \cup E'')$ is a disconnected graph consisting of two connected components, one belonging to L and one to $\overline{L}$. U itself must belong in one of L and $\overline{L}$ implying that all of its components must belong to L or all to $\overline{L}$, which is a contradiction.
   (b) L contains no connected graph. Thus, since L is nontrivial, it contains at least one disconnected graph whose connected components belong to L. But all connected graphs belong to $\overline{L}$ which is a contradiction.

□

Theorem 9. *Any nontrivial graph language $L \subset \mathcal{H}$ is not stably decidable by the GDMPP model.*

Proof Idea. The proof of the result is based on the very simple observation that in disconnected graphs, the various components cannot communicate with each other. Then Lemma 2 can be used to argue that a language (or its complement) must contain at least one disconnected graph with a component not in the language, so any protocol making some decision on the whole graph would make the opposite decision on the component (since this component does not belong to the language and is isolated from the other components), which is contradictory. □

Proof. Let L be such a language and assume that there exists a GDMPP $\mathcal{A}_L$ that stably decides it. Thus, $\mathcal{A}_L$ has eventually all the agents of G giving output 1 if $G \in L$ and all giving output 0 if $G \notin L$. Moreover, the protocol $\mathcal{A}_{\overline{L}}$ that has the output map of $\mathcal{A}_L$ complemented stably decides $\overline{L}$. Those GDMPPs (and in fact any GDMPP) have no way to transmit data between agents of different components when run on disconnected graphs. In fact it is trivial to see that, when run on disconnected graphs, those protocols essentially run individually on the different components of those graphs. This means that when, for example, $\mathcal{A}_L$ runs on a disconnected graph G, where G has at least two components $G_1, G_2, \ldots, G_t$, then $\mathcal{A}_L$ runs in t different copies, one for each component, and each such copy stably decides the membership of the corresponding component (on which it runs on) in L. The same holds for $\mathcal{A}_{\overline{L}}$. By Lemma 2 there exists at least one disconnected graph in L with at least one component in $\overline{L}$ or at least one disconnected graph in $\overline{L}$ with at least one component in L. If L contains such a disconnected graph then, obviously, $\mathcal{A}_L$ when run on this graph, call it G, has eventually all the nodes of the component(s) in $\overline{L}$ giving 0 as output. This is a contradiction, because $G \in L$ and $\mathcal{A}_L$ stably decides L, which means that all agents should eventually output 1. If $\overline{L}$ contains such a disconnected graph then the contradiction is symmetric. □

As an immediate consequence we get the following corollary:

Corollary 1. *The graph language $C = \{G \in \mathcal{H} \mid G$ is (weakly) connected$\}$ is not GDMPP-decidable.*

Proof. C is a nontrivial graph language and Theorem 9 applies. □

6 Future Work

Whether the graph language $2C$ (Theorem 8) is not stably decidable by the GDMPP model in the general case remains an interesting open problem. If it were, we believe that proving the undecidability of many other properties, like kC (all graphs with at least one directed cycle of length k) and k-transivity, would become an easy next step. Our primary focus is to eventually provide a complete characterization of the class of stably decidable graph languages in the weakly-connected case. Moreover, consider the variant of the GDMPP model in which the communication graph G is always complete and an *edge initialization function* $\iota : E \to \{0, 1\}$ indicates a subgraph of G whose membership in a language is sought. What are the stably decidable graph languages here?

References

1. D. Angluin, J. Aspnes, M. Chan, M. J. Fischer, H. Jiang, and R. Peralta. Stably computable properties of network graphs. In Proc. Distributed Computing in Sensor Systems: 1st IEEE International Conference, pages 63-74, 2005.
2. D. Angluin, J. Aspnes, Z. Diamadi, M.J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. In *23rd Annual ACM Symposium on Principles of Distributed Computing* (*PODC*), pages 290-299, New York, NY, USA, 2004. ACM.
3. D. Angluin, J. Aspnes, Z. Diamadi, M.J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4): 235-253, 2006.
4. D. Angluin, J. Aspnes, and D. Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3): 183-199, Sept. 2008.
5. D. Angluin, J. Aspnes, and D. Eisenstat. Stably computable predicates are semilinear. In *Proc. 25th Annual ACM Symposium on Principles of Distributed Computing*, pages 292-299, 2006.
6. D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4): 279-304, November 2007.
7. J. Aspnes and E. Ruppert. An introduction to population protocols. *Bulletin of the European Association for Theoretical Computer Science*, 93:98-117, October 2007. Columns: *Distributed Computing*, Editor: M. Mavronicolas.
8. J. Beauquier, J. Clement, S. Messika, L. Rosaz, and B. Rozoy. Self-stabilizing counting in mobile sensor networks. Technical Report 1470, LRI, Université Paris-Sud 11, 2007.
9. O. Bournez, P. Chassaing, J. Cohen, L. Gerin, and X. Koegler. On the convergence of population protocols when population goes to infinity. To appear in *Applied Mathematics and Computation*, 2009.
10. I. Chatzigiannakis, S. Dolev, S. P. Fekete, O. Michail, and P. G. Spirakis. Not all fair probabilistic schedulers are equivalent. In *13th International Conference On Principles Of DIstributed Systems (OPODIS)*, pages 33-47, 2009.
11. I. Chatzigiannakis, O. Michail, S. Nikolaou, A. Pavlogiannis, and P. G. Spirakis. All symmetric predicates in $NSPACE(n^2)$ are stably computable by the mediated population protocol model. FRONTS Technical Report FRONTS-TR-2010-17, http://fronts.cti.gr/aigaion/?TR=155, April 2010.
12. I. Chatzigiannakis, O. Michail, and P. G. Spirakis. Brief announcement: decidable graph languages by mediated population protocols. In *23rd International Symposium on Distributed Computing (DISC)*, Elche, Spain, Sept. 2009.
13. I. Chatzigiannakis, O. Michail, and P. G. Spirakis. Mediated population protocols. In *36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 363-374, Rhodes, Greece, 2009.
14. I. Chatzigiannakis and P. G. Spirakis. The dynamics of probabilistic population protocols. In *Distributed Computing, 22nd International Symposium, DISC*, volume 5218 of *Lecture Notes in Computer Science*, pages 498-499, 2008.
15. C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and E. Ruppert. When birds die: Making population protocols fault-tolerant. In *Proc. 2nd IEEE International Conference on Distributed Computing in Sensor Systems*, pages 51-66, 2006.