

Threshold-based Network Structural Dynamics^{*}

Evangelos Kipouridis¹[0000–0002–5830–5830], Paul G. Spirakis^{2,3}[0000–0001–5396–3749], and Kostas Tsichlas³[0000–0003–4107–9520]

¹ Basic Algorithms Research Copenhagen (BARC), University of Copenhagen, Copenhagen, Denmark, Universitetsparken 1, 2100
kipouridis@di.ku.dk

² Department of Computer Science, University of Liverpool, Ashton Str., Liverpool L69 3BX, UK
P.Spirakis@liverpool.ac.uk



³ Computer Engineering and Informatics Department, University of Patras, Patras, Greece
{ktsichlas,spirakis}@ceid.upatras.gr

Abstract. The interest in dynamic processes on networks is steadily rising in recent years. In this paper, we consider the (α, β) -Threshold Network Dynamics ((α, β) -Dynamics), where $\alpha \leq \beta$, in which only structural dynamics (edge dynamics of the network) are allowed, guided by local threshold rules executed by each node. In particular, in each discrete round t , each active pair of nodes u and v , computes a value $\mathcal{E}(u, v)$ (the potential of the pair) as a function of the local structure of the network at round t around the two nodes. If $\mathcal{E}(u, v) < \alpha$ then the link (if it exists) between u and v is removed; if $\alpha \leq \mathcal{E}(u, v) < \beta$ then an existing link among u and v is maintained; if $\beta \leq \mathcal{E}(u, v)$ then a link between u and v is established if not already present. New nodes cannot be inserted as a result of the protocol, and existing nodes cannot be removed.

The microscopic structure of (α, β) -Dynamics appears to be simple, so that we are able to rigorously argue about it, but still flexible, so that we are able to design meaningful microscopic local rules that give rise to interesting macroscopic behaviors. Our goals are the following: a) to investigate the properties of the (α, β) -Threshold Network Dynamics and b) to show that (α, β) -Dynamics is expressive enough to solve complex problems on networks.

Our contribution in these directions is twofold. We rigorously exhibit the claim about the expressiveness of (α, β) -Dynamics, both by designing a simple protocol that provably computes the k -core of the network as well as by showing that (α, β) -Dynamics are in fact Turing-Complete. Second and most important, we construct general tools for proving stabilization that work for a subclass of (α, β) -Dynamics and prove speed of convergence in a restricted setting.

Keywords: Network Dynamics · Stabilization.

^{*} Evangelos Kipouridis received funding from the European Union’s Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 801199.  Evangelos Kipouridis is also supported by Thorup’s Investigator Grant 16582, Basic Algorithms Research Copenhagen (BARC), from the VILLUM Foundation. 

1 Introduction

The interplay between the microscopic and the macroscopic in terms of emergent behavior shows an increasing interest. The most striking examples come from biological systems that seem to form macroscopic structures out of local interactions between simpler structures (e.g., computation of shortest paths by *Physarum Polycephalum* [26] or of maximal independent sets by the fly's nervous systems [1]). The underlying common characteristic of these systems is the emergent behavior at the macroscopic level out of simple local interactions at the microscopic level. This is one of the reasons why we are interested on how very simple distributed protocols can give rise to complex emergent behavior due to their synergistic/antagonistic interactions [9]. In some of these systems, the dynamic processes are purely structural with respect to the network. Examples of such systems are network generation models [7,32], community detection [35], "life-like" cellular automata [30], robot motion [28] and go all the way up to being a fundamental physics theory for space [33,34]. In view of this recent trend, a stream of work is devoted to the study of such dynamics per se, without a particular application in mind (e.g., [16]). Motivated by such a plethora of examples, we study the stabilization properties of protocols that affect solely the structure of networks.

Henceforth, we will use the term *dynamic network* to represent networks that change due to some process, although in the literature one can find other terms like adaptive networks, time-varying networks, evolving networks and temporal networks that essentially refer to the same general idea of time-dependent networks w.r.t. structure and states. The study of the processes that drive dynamic networks and their resulting properties has been the focus of many different fields but in general one can discern between two distinct viewpoints without excluding overlap: **a) complex systems viewpoint (physics, sociology, ecology, etc.):** the main focus is on modeling (e.g., differential/difference equations, cellular automata, etc. - see [29]) and qualitative analysis (by means of mean field approximations, bifurcation analysis etc.). The main questions here are of qualitative nature and include phase transitions, complexity of system behavior, etc. Rigorous analysis is not usual and simulation is the main tool for providing results. **b) computational viewpoint (mainly computer science and communications):** the main focus is on the computational capabilities (computability/complexity) of dynamic networks in various settings and with different assumptions adopting a rigorous approach or simulation.

When designing local rules aiming at some particular global/emergent behavior, it is usually difficult, or at the very least cumbersome, to prove correctness [9]. This is why most studies in complex systems of this sort are based on experimental evidence for their correctness. Indeed, having a general framework for modeling algorithmic emergence and proving related properties, would impact directly the distributed computing field as well as provide another tool for analysis of natural systems that exhibit emergence. To accomplish such a feat, one should start from analyzing simple systems that give rise to interesting macroscopic behaviors. In this paper, we study a dynamic network driven by a simple

protocol that depends only on the network's structure. This protocol is executed by each node in a synchronous manner. The protocol is the same for all nodes and can only affect the structure of the network and not the state of edges or nodes. The locality of the protocol is defined with respect to the available interactions for each node. We define the (α, β) edge dynamics (we call the model (α, β) -Dynamics henceforth) and discuss related work in Section 2. In Section 3, we discuss a particular protocol that computes the α -core and the $(\alpha - 1)$ -crust [8] of an arbitrary provided network. In Section 4, we provide guarantees on the speed of stabilization for a subclass of (α, β) -Dynamics while in Section 5 we provide a proof of stabilization for a more general class of such protocols. In this way, we provide general results for (α, β) -Dynamics that may be directly applied elsewhere, e.g., in the case of restricted Network Automata [30]. In Section 6, we prove that (α, β) -Dynamics is Turing-Complete. Finally, in Section 7, we discuss some extensions of the proposed model and we conclude in Section 8.

2 Preliminaries

Assume that an undirected simple network $G^{(0)} = (V, E^{(0)})$ evolves over discrete time steps based on a set of rules that we specify later. We represent the network at time t by $G^{(t)} = (V, E^{(t)})$. We denote the *distance* between two nodes u, v in $G^{(t)}$ as $d^{(t)}(u, v)$. Let $n = |V|$, $m^{(t)} = |E^{(t)}|$ and let $N_{G^{(t)}}(u)$ be the set of all neighbors of node u and $d_{G^{(t)}}(u)$ be the *degree* of node u in network $G^{(t)}$. We define $|E^{(t)}(u, v)|$ to be the number of edges between u and v at time t (either 0 or 1), and more generally $|E^{(t)}(U)|$ to be the number of edges between nodes in the set $U \subseteq V$ at time t . It follows that $|E^{(t)}(N_{G^{(t)}}(u) \cap N_{G^{(t)}}(v))|$ is the number of edges between common neighbors of u and v at time t . Let $G^{(t)}[S]$ represent the induced subgraph of the node set $S \subseteq V$ in $G^{(t)}$.

Finally, we define a particular type of potential functions, that we call *proper* functions. Informally, these functions can be used to describe “rich get richer” behavior. We provide several results related to proper functions in Sections 4 and 5.

Definition 1. *We say that a function $f : \mathbb{N}^2 \rightarrow \mathbb{R}$ is proper if it has the following two properties: i) Non-decreasing, that is $f(x, y + \epsilon) \geq f(x, y)$ for $\epsilon > 0$ (similarly $f(x + \epsilon, y) \geq f(x, y)$) and ii) Symmetric, $f(x, y) = f(y, x)$.*

The second property is related to the fact that we consider undirected networks.

2.1 (α, β) -Dynamics - Threshold Network Dynamics

From a bird's eye view, (α, β) -Dynamics is a discrete-time dynamic network of agents. The dynamics are applied only on the edges, while there is no insertion or removal of agents. The agents update their neighborhood synchronously by executing the same protocol. At the beginning of each time step, a (possibly adversarial) scheduler chooses a set of so-called “*active pairs*” that interact in this

step; the active pairs may change in each step. The protocol applies a threshold over the value of a function defined on the active pair of nodes u and v . The function depends only on the c -neighborhood of the active pair of nodes. The value of the function on u and v determines whether the undirected edge uv will be inserted (if not present), deleted (if present) or remain unchanged.

More precisely, the (α, β) -Dynamics is a discrete-time dynamic stateless network of agents $G^{(t)} = (V, E^{(t)})$. It is stateless because the dynamics driven by the protocol depend only on the structure of the network and not on state information stored in each node/edge. The dynamics involve the edges of the network while the set of agents is static. All interactions are pairwise and for each interaction between nodes u and v , these execute the same protocol that affects the edge between them. The protocol is *consistent*, in the sense that it comes to the same decision about the existence of the edge between u and v , both when executed by u and by v . The execution evolves in synchronous discrete time rounds. In the following, the edge $e^{(t)}$ is also used as a boolean variable. In particular, when $e^{(t)} = 0$ then $e^{(t)} \notin E^{(t)}$, while $e^{(t)} = 1$ means that $e^{(t)} \in E^{(t)}$. Let α and β be parameters that correspond to a lower and an upper threshold, respectively. Initially, the network $G^{(0)}$ is given as well as the constant thresholds α and β . Formally, (α, β) -Dynamics is a triple $(G^{(0)}, \mathcal{S}, \mathcal{A}(\alpha, \beta))$ defined as follows:

- $G^{(0)} = (V, E^{(0)})$: A network of nodes V and edges $E^{(0)}$ between nodes at time 0. This is the initial configuration of the dynamic process. Each node $v \in V$ has a distinct id and maintains a routing table with all its edges.
- \mathcal{S} : A set that contains the pairwise interactions between nodes. We represent it by a possibly infinite series of sets of pairwise interactions $C^{(t)}$. Each set $C^{(t)}$ contains the pairwise interactions between nodes activated at time step t in the network $G^{(t)}$. If the pair $uv \in C^{(t)}$, then we call the pair uv *active* at time instance t . An interaction between nodes u and v , assumes direct communication between u and v irrespectively of whether u and v are connected by an edge in $G^{(t)}$. In the following, we refer to $C^{(t)}$ as the *interaction set* for time step t .
- $\mathcal{A}(\alpha, \beta, \mathcal{E})$: The protocol executed in each round by each active pair of nodes as defined by the interaction set $C^{(t)}$. The (α, β) -Dynamics is defined for the following family of protocols:

Protocol $\mathcal{A}(\alpha, \beta, \mathcal{E})$ at node u for an active pairwise interaction $uv \in C^{(t)}$:

Compute the potential $\mathcal{E}(u, v)$.

1. If $\mathcal{E}(u, v) < \alpha$ then edge $uv^{(t+1)} = 0$.
2. If $\alpha \leq \mathcal{E}(u, v) < \beta$ then edge $uv^{(t+1)} = uv^{(t)}$.
3. If $\mathcal{E}(u, v) \geq \beta$ then edge $uv^{(t+1)} = 1$.

The *potential* of a pair of nodes u and v at round t is a function related to this pair and is represented by $\mathcal{E}_{G^{(t)}}^{(t)}(u, v) : G^{(t)}[S_{uv}] \rightarrow \mathbb{R}$, for some $S_{uv} \subseteq V$. The domain of the potential is the induced subgraph $G^{(t)}[S_{uv}]$ defined by the set of nodes S_{uv} .

Discussion on the potential function: In this paper S_{uv} contains the nodes of the local neighborhood of nodes u and v . In particular, we consider S_{uv} to consist of all nodes that are within a constant distance from u or from v (the constant is 1 throughout the paper, except for Section 6 where it is 3). The local structure is defined explicitly by the potential function. We write $\mathcal{E}^{(t)}(u, v)$ or $\mathcal{E}(u, v)$ when the network and the time we are referring to are clear from the context. An example of such a function defined in [35] that is used to detect communities in networks is the following:

$$\mathcal{E}^{(t)}(u, v) = |N_{G^{(t)}}(u) \cap N_{G^{(t)}}(v)| + |E^{(t)}(u, v)| + |E(N_{G^{(t)}}(u) \cap N_{G^{(t)}}(v))|$$

The potential is equal to the number of common neighbors between u and v , plus the number of edges between u and v (0 or 1), plus the number of edges between the common neighbors of u and v . Finally, the fact that the protocol is consistent imposes the restriction on the potential function to be *computationally symmetric* for an arbitrary active pair uv , meaning that $\mathcal{E}(u, v)$ is the same when computed in u and in v .

The computational capabilities of each node are similar to a Turing machine with bounded memory (Linear Bounded Automaton - LBA). Each node has enough memory to store its routing table as well as the topological information necessary for computing the potential. To avoid complex computations we impose that the potential function should be computable in polynomial time with respect to its input size. The complexity of the protocol depends solely on the definition of the potential function, since the rest of the protocol are simple threshold comparisons. Similarly to consensus dynamics [9], we require our protocol to be simple and lightweight [10] and to realize natural, local and elementary rules subject to the constraint that structural dynamics are considered. To this end, we require the potential function to respect the following constraints:

1. The potential function has access to a small constant distance c away from the two interacting nodes.
2. The potential function must not distinguish between nodes - thus not allowing for special nodes (e.g., leaders) [10]⁴.
3. The potential function must be network-agnostic, in the sense that it is designed without having any access to the topology of $G^{(0)}$.

These restrictions combined with the computational capabilities of nodes do not allow the protocol to use shortcuts for computation in terms of hardwired information in the potential function (node ids) or in terms of replacing large subgraphs by other subgraphs.

Execution of the protocol: In each round, the protocol is executed on nodes of active pairs determined by the current interaction set. A pairwise interaction between nodes u and v requires the computation of the potential between the

⁴ Therefore, we only use identifiers of nodes for analysis purposes

two nodes and then a decision is made as for the edge between them based on the thresholds α and β . Each round of the computation for node u (symmetrically for v) is divided into the following phases:

1. u sends messages to its local neighborhood (with the exception of v , if edge uv exists) requesting information related to the computation of the potential function
2. u receives the requested information
3. u sends its information to v
4. u receives v 's information
5. u computes the potential
6. u decides as for the edge uv w.r.t. thresholds.

There is no restriction on the size of the messages sent and received by each node. Direct communication is assumed (in phases (3) and (4)) between u and v irrespectively of the existence of edge uv . The consistency of the protocol guarantees that the result of its execution is the same for both nodes.

Some Notes on the Interaction Sets: The interaction set $C^{(t)}$ at time t supports parallelism since it is a set of pairwise interactions with size at most $\binom{n}{2}$. Thus, there may be many active pairs in each step. For example, consider the case where all $\binom{n}{2}$ possible pairs are active. This means that simultaneously the potential is computed for all possible pairwise interactions and the edges are updated analogously. In [35], a serialization of this setting is used to detect communities in networks. In general, we may assume anything about the generation of the interaction sets (adversarial, stochastic, etc.). Arguing about an arbitrary set of active pairs for each time instance t is the most general case, since \mathcal{A} can make no assumption at all about the active pairs within each step.

On a more technical note, the notion of the interaction sets has two different but not necessarily mutually exclusive uses. On the one hand, the interaction sets model restrictions set by the environment on the interactions. For example, in a social network the interaction pairs may be the established friendship links but interactions can also be imposed from other activities external to the social network, such as a random meeting in a cafeteria; interaction sets can thus differ from the edges of the network. On the other hand, interaction sets are used as a tool for analysis reasons, to describe the communication links that the protocol \mathcal{A} enforces on $G^{(t)}$ (e.g., when a node only communicates with nodes at distance at most 2). In this paper, we present some general results w.r.t. the choice of the interaction sets. For example, $C^{(t)}$ may be adversarial for all t , and, under a weak fairness condition, our algorithms are still able to stabilize (see the following discussion on convergence/stabilization for a formal definition of the weak fairness condition and stabilization, as well as Sections 3 and 5 for the aforementioned results).

Convergence/Stabilization: (α, β) -Dynamics is stateless, in the sense that the dynamics driven by the algorithm \mathcal{A} consider only the structure of the network. No states that are stored at nodes or edges are considered in the dynamic

evolution expressed by (α, β) -Dynamics. As a result, the network $G^{(t)}$ completely defines the configuration of the system at time t . We say that $G^{(t)}$ *yields* $G^{(t+1)}$, when a transition takes place from $G^{(t)}$ to $G^{(t+1)}$ after time step t , represented as $G^{(t)} \xrightarrow{C^{(t)}} G^{(t+1)}$, which is the result of the \mathcal{A} protocol for all active pairs in the interaction set $C^{(t)}$. Similarly, we write $G^{(t)} \rightsquigarrow G^{(t')}$, for $t' > t$, if there exists a sequence of transitions $G^{(t)} \xrightarrow{C^{(t)}} G^{(t+1)} \xrightarrow{C^{(t+1)}} \dots \xrightarrow{C^{(t'-1)}} G^{(t')}$. An *execution* of (α, β) -Dynamics is a finite or infinite sequence of configurations $G^{(0)}, G^{(1)}, G^{(2)}, \dots$ such that for each t , $G^{(t)}$ yields $G^{(t+1)}$, where $G^{(0)}$ is the initial network.

We say that the algorithm *converges* or *stabilizes* when $\exists t$ such that $\forall t' > t$ it holds that $G^{(t)} = G^{(t')}$, meaning that the network does not change after time t . The *output* of the (α, β) -Dynamics is the network that results after stabilization has been reached. The time complexity of the protocol is the number of steps until stabilization. The time complexity of the protocol heavily depends on $C^{(t)}$. If, for example, there exists a T where for all $t \geq T$ it holds that $C^{(t)}$ is always the null set, then the algorithm stabilizes although it would not stabilize for a different choice of $C^{(t)}$. To avoid stalling, we employ the *weak fairness condition* [2,3] that essentially states that all pairs of nodes interact infinitely often, thus imposing that at some point of time an interaction set will trigger a change in the network, if one is possible.

2.2 Related Work

The main work on dynamic networks stems either from computer science or from complex systems and is inherently interdisciplinary in nature. In the following, we only highlight results that are directly related to ours (a more extensive discussion can be found in [23]). In computer science, a nice review of the dynamic network domain [25] proposes a partitioning of the current literature into three subareas: Population Protocols ([3], [4]), Powerful Dynamic Distributed Systems (e.g., [27]) and models for Temporal Graphs (e.g., [12]). (α, β) -Dynamics can be compared to Population Protocols, where anonymous agents with only a constant amount of memory available interact with each other and are able to compute functions, like leader election. Their scheduler determines the set of active pairs of nodes at each time step. The choice is made by a scheduler either arbitrarily (adversarial scheduler) or uniformly at random (uniform random scheduler). The uniform scheduler is used for designing various protocols due to the probabilistic accommodations for analysis it provides. The major differences to our approach are with respect to dynamics and the scheduler. Population protocols study state dynamics while in our case we study stateless structural dynamics. In addition, in our approach, the scheduler consists of a set of pairwise interactions, thus allowing for many computations between pairs of nodes during a time step (parallel time). This parallelism of the scheduler may "artificially" reduce the number of rounds but it can also complicate the protocol leading to interesting research questions. Similarly to population protocols, the notion of consensus dynamics [10,9] that refers to distributed processes that resemble interacting particle systems considers simple and lightweight protocols on states of

agents. (α, β) -Dynamics could be cast in such a framework as purely structural dynamics that on the one hand supports simple, uniform and lightweight protocols while on the other hand requires necessarily the communication of structural information between nodes. In the same manner, motivated by population protocols, the Network Constructors model also studies state dynamics that affect the structure of the network resulting in structural dynamics as well, and thus it is much closer to (α, β) -Dynamics. In [23,24] the authors study what stable networks can be constructed (like paths, stars, and more complex networks) by a population of finite-automata. Among other complexity related results they also argue that the Network Constructors model is Turing-Complete. Our main differences to the network constructors model are the following:

1. Our motivation comes from the complex systems domain as well, and thus we are more interested in as general as possible convergence/stabilization theorems apart from particular network constructions (like the α -core in our case).
2. They use states for the structural dynamics while in our case the dynamics are stateless. This means that Network Constructors use states that change according to the protocol, which in turn drive the structural changes of the network (coupled dynamics). In our case, we use only the knowledge of the structure of the network to make structural changes.
3. They always start from a null network while we start from an arbitrary one.

A similar notion is graph relabeling systems [21], where one chooses a subgraph and changes it based on certain rules. These systems are usually applied on static graphs but they have also been applied to dynamic graphs [11]. The focus in this case is to *impose properties on the dynamic graphs so that a particular computation is possible*, assuming adversarial dynamic graphs. (α, β) -Dynamics is also related to - in fact can easily simulate - graph generating models. The Barabási-Albert model [7] can be simulated by simply setting \mathcal{A} to add an edge between two nodes in $G^{(t)}$ for each interacting pair in $C^{(t)}$. These interacting pairs in $C^{(t)}$ are specified based on the stochastic preferential-attachment mechanism. Similarly, the Watts-Strogatz model [32] can be simulated by starting with a regular ring lattice and then in each step set the appropriate edges stochastically in $C^{(t)}$ to rewire them. Finally, (α, β) -Dynamics is also related to topological self-stabilization [14], which aims at allowing the nodes themselves to establish a desired overlay network, independently of the initial configuration by forwarding, inserting and deleting links to adjacent nodes. Although our model could also be used for topological self-stabilization it would do so under a more restrictive setting since the protocol has a very particular form.

In the study of complex systems, one of the tools used for modeling is cellular automata. Cellular automata use simple update rules that give rise to interesting patterns [6], [17]. Structurally Dynamic Cellular Automata (SDCA) that couples the topology with the local site 0/1 value configuration were introduced in [19]. They formalize this notion and move to an experimental qualitative analysis of its behavior for various parameters. They left as an extension (among others) of SDCA purely structural CA models in which there are no value configurations

as it holds in the (α, β) -Dynamics studied in this paper. A model for coupling topology with functional dynamics was given in [30], termed Functional Network Automata (FNA), and was used as a model for a biological process. They also defined the restricted Network Automata (rNA), which as (α, β) -Dynamics allows only for stateless structural network dynamics. rNA forces every possible pair of interactions to take place, meaning that for all t it holds that $C^{(t)}$ contains all $\binom{n}{2}$ possible edges of the n nodes. All their results are qualitative and are based on experimentation. By using the machinery built in Section 5, we show that for the family of protocols we consider, rNA always stabilizes. To further stimulate the reader as for the need of looking at (α, β) -Dynamics, the author in [28] looked at modular robots as an evolving network with respect only to their topology. The author designed a protocol common to all modules of the robot, that turns a tree topology to a chain topology conjecturing that stabilization is always achieved, but to the best of our knowledge it is still unresolved.

3 Taking the Minimum

As a motivation and exhibition of (α, β) -Dynamics, we first discuss the following interesting example. We define the potential of a pair of nodes u and v as $\mathcal{E}(u, v) = \min\{d_{G^{(t)}}(u), d_{G^{(t)}}(v)\}$, that is the potential is equal to the minimum degree of the two nodes.

This potential function respects all constraints described in 2.1. As required, computing the function can be done in linear time. Additionally, the potential function only depends on nodes at a constant distance (at most 1) from either u or v . It is also computationally symmetric and thus the protocol is consistent.

It is interesting to notice the similarity of our process, and the process of acquiring the k -core (or complementary the $(k-1)$ -crust) of a simple undirected graph [8,31].

Definition 2. *The k -core H of a graph G is the unique maximal subgraph of G such that $\forall u \in H$ it holds that $\deg_H(u) \geq k$. All nodes not in H form the $(k-1)$ -crust of G .*

The k -core plays an important role in studying the clustering structure of networks [22]. In [8] it was proved that the following process efficiently computes the k -core of a graph:

Lemma 1. *Given a graph G and a number k , one can compute G 's k -core by repeatedly deleting all nodes whose degree is less than k .*

The following theorem states that stabilization to the k -core is achieved for an arbitrary set of interaction sets \mathcal{S} . Furthermore, the stabilization occurs after $O(m)$ rounds of changes in the network, where m is the number of edges in $G^{(0)}$. Note that this is not the time complexity of the protocol, since there may be many idle rounds between rounds with changes, depending on the interaction sets.

Theorem 1. *When $\mathcal{E}(u, v) = \min\{d_{G^{(t)}}(u), d_{G^{(t)}}(v)\}$, (α, β) -Dynamics for any value of $\alpha \leq n - 1 < \beta$ and any \mathcal{S} , stabilizes in a network where all isolated nodes form the $(\alpha - 1)$ -crust and the rest the α -core of $G^{(0)}$ in $O(m)$ rounds where changes happen, where m is the number of edges in $G^{(0)}$.*

Proof. First of all, even if a node connects with any other node, its degree will be $n - 1$. Thus, it holds that $\min\{d(u), d(v)\} \leq n - 1 < \beta$. This ensures that no edge will ever be created by the (α, β) -Dynamics. Thus, only deletions of edges can be performed. As a result, the maximum number of rounds where a change happens is a straightforward $O(m)$. What we need to show is that the output of the protocol is a network where all isolated nodes belong to the $(\alpha - 1)$ -crust of $G^{(0)}$ and the rest of the nodes belong to the α -core of $G^{(0)}$.

To prove our claim we change slightly the algorithm described in Lemma 1 to process edges instead of nodes. This change is made so that the (α, β) -Dynamics described in this section will be in fact a realization of this main memory algorithm and thus its output will be the α -core of $G^{(0)}$. Indeed, one can compute G 's α -core by repeatedly deleting all edges for which one of its endpoints has degree $< \alpha$. The procedure stops when there is no such remaining edge, that is, all edges have endpoints with degree $\geq \alpha$. The order in which the edges are considered is irrelevant. It is easy to see that this algorithm computes the α -core of the given network and in fact it is the (α, β) -Dynamics described in this section. \square

A final note concerns the time complexity. Note that the aforementioned theorem does not state anything about the time complexity of the protocol; it just states the maximum number of rounds where a change happens. We can compute the time complexity if we describe the interaction sets \mathcal{S} . If we assume that $\forall t : C^{(t)} = E^{(t)}$, that is the interaction sets contain all edges and only those of the $G^{(t)}$ network then the time complexity is $O(n)$. This is because, at each round it is guaranteed that one node will become isolated unless stabilization has been achieved. Similarly, if we assume that at each time step only one pair of nodes is active and it is chosen uniformly at random, then the (α, β) -Dynamics stabilizes in $O(mn^2 \log m)$ steps by a simple application of the coupon collector problem [15] on the selection of edges.

4 Speed of stabilization results

In this section we study (α, β) -Dynamics where the potential of a pair of nodes is any symmetric non-decreasing function on the degrees of its two endpoints, as we saw in Section 3. We prove stabilization as well as that the number of steps needed until stabilization is $O(n)$, assuming $\alpha = \beta$ and that each interaction set contains always all possible pairs. More formally, we define the potential of a pair uv to be $\mathcal{E}(u, v) = f(d_{G^{(t)}}(u), d_{G^{(t)}}(v))$, where f is a *proper* (symmetric and non-decreasing in both variables) function. \mathcal{S} is fixed and contains all $\binom{n}{2}$ possible pairwise interactions.

For the graph $G^{(t)}$, let $R^{(t)}(u, v)$ be an equivalence relation defined on the set of nodes V for time t , such that $uv \in R^{(t)}$ if and only if $d_{G^{(t)}}(u) = d_{G^{(t)}}(v)$. The

equivalence class $R_i^{(t)}$ corresponds to all nodes with degree $d(R_i^{(t)})$, where i is the rank of the degree in decreasing order. This means that the equivalence class $R_1^{(t)}$ contains all nodes with maximum degree in $G^{(t)}$. Assuming that $n = |V|$, the maximum number of equivalence classes is $n - 1$, since the degree can be in the range $[0, n - 1]$ and no pair of nodes u, v with $d(u) = 0$ and $d(v) = n - 1$ can exist. Let $|G^{(t)}|$ be the number of equivalence classes in graph $G^{(t)}$. Before moving to the proof, we give certain properties of the dynamic process that hold for all $t \geq 1$, that is they hold after at least one round of the process (initialization). These properties will be used in the proof for stabilization.

From a bird eye's view of what follows, we notice that in this framework two nodes behave in the same way if their degrees are the same, due to the definition of the potential function and the interaction set. Furthermore, if at any time a node u has degree at least as large as the degree of another node v , then it will form at least as many edges in the next time step, thus preserving the relative order of their degrees. These observations lead us to define the aforementioned equivalence classes related to the degrees of the nodes, whose properties allow us to inductively prove our upper bounds. This intuition is formalized in the following properties:

Property 1 *If $d_{G^{(t)}}(u) \geq d_{G^{(t)}}(w)$, then $N_{G^{(t+1)}}(u) \supseteq N_{G^{(t+1)}}(w)$.*

Proof. As each interaction set contains all pairs of nodes and $\alpha = \beta$, if v is a neighbor of w in $G^{(t+1)}$, then in the previous time-step the potential was $\mathcal{E}^{(t)}(v, w) \geq \beta$. Then it also holds that $\mathcal{E}^{(t)}(v, u) \geq \beta$, since f is non-decreasing. This implies that v is also a neighbor of u in $G^{(t+1)}$. \square

Nodes that have the same degree at time t , share the same neighbors at time $t + 1$.

Property 2 *If $d_{G^{(t)}}(u) = d_{G^{(t)}}(w)$, then $N_{G^{(t+1)}}(u) = N_{G^{(t+1)}}(w)$.*

Proof. As in the proof of Property 1, due to the equality of the degrees, it also holds that any neighbor v of u is a neighbor of w and respectively any neighbor v of w is a neighbor of u . \square

In the following, we discuss properties related to equivalence classes.

Property 3 *The number of equivalence classes in $G^{(t+1)}$ is less than or equal to the number of equivalence classes in $G^{(t)}$.*

Proof. By Property 2, nodes that belong to the same equivalence class at time $t > 0$ will always belong to the same equivalence class for all $t' > t$. \square

Property 4 *If $G^{(t+1)}$ has the same number of equivalence classes as $G^{(t)}$, then $\forall i, R_i^{(t)} = R_i^{(t+1)}$.*

Proof. By Property 2, all nodes in $R_i^{(t)}$ are in the same equivalence class in $G^{(t+1)}$. Furthermore, for any $i < i'$ such that the nodes of $R_i^{(t)}$ are in $R_j^{(t+1)}$ and the nodes of $R_{i'}^{(t)}$ are in $R_{j'}^{(t+1)}$, it holds that $j \leq j'$ by Property 1. As the number of equivalence classes stay the same, it must be that $j \neq j'$. This can only happen if for all i we have $R_i^{(t)} = R_i^{(t+1)}$. \square

The following lemma shows how equivalence classes behave w.r.t. edge distribution.

Lemma 4. *If an arbitrary node u in $R_i^{(t)}$ is connected with some node w in $R_j^{(t)}$, then u is connected with every node x in every equivalence class $R_k^{(t)}$, such that $k \leq j$ and $t > 0$.*

Proof. By definition of equivalence classes, for all nodes $x \in R_k^{(t)}$ it holds that $d_{G^{(t)}}(x) \geq d_{G^{(t)}}(w)$. If $d_{G^{(t-1)}}(x) < d_{G^{(t-1)}}(w)$, then by Property 1 we have $N_{G^{(t)}}(x) \subseteq N_{G^{(t)}}(w)$; since $d_{G^{(t)}}(x) \geq d_{G^{(t)}}(w)$, this can only happen if $N_{G^{(t)}}(x) = N_{G^{(t)}}(w)$, meaning that x is also connected with u .

Else, we have $d_{G^{(t-1)}}(x) \geq d_{G^{(t-1)}}(w)$. As each interaction set contains all pairs of nodes, w being connected with u at time step t means that $\mathcal{E}^{(t-1)}(u, w) = f(d_{G^{(t-1)}}(u), d_{G^{(t-1)}}(w)) \geq \beta$. But f is proper, meaning that $\mathcal{E}^{(t-1)}(u, x) = f(d_{G^{(t-1)}}(u), d_{G^{(t-1)}}(x)) \geq f(d_{G^{(t-1)}}(u), d_{G^{(t-1)}}(w)) \geq \beta$. \square

We prove by induction that this (α, β) -Dynamics always stabilizes in at most $|G^{(0)}| + 1$ steps. To begin with, it is obvious that the clique \mathcal{K}_n as well as the null graph $\overline{\mathcal{K}_n}$ both stabilize in at most one step, for any value of β . The following renormalization lemma describes how the number of equivalence classes is reduced and is crucial to the induction proof.

Lemma 5. *If $d(R_1^{(t)}) = n - 1$, $\forall t \geq c, c \in \mathbb{N}$, and the subgraph $G^{(c)} \setminus R_1^{(c)}$ stabilizes for any value of β and proper function f , then $G^{(c)}$ stabilizes as well. Similarly, if $d(R_{|G^{(t)}|}^{(t)}) = 0$, $\forall t \geq c, c \in \mathbb{N}$, and the subgraph $G^{(c)} \setminus R_{|G^{(c)}|}^{(c)}$ stabilizes for any value of β and proper function f , then $G^{(c)}$ stabilizes as well. The time it takes for $G^{(c)}$ to stabilize is the same as the time it takes for the induced subgraph to stabilize for both cases.*

Proof. The main idea is that we consider two different sets of nodes: $R_1^{(c)}$ and $V \setminus R_1^{(c)}$. Due to our hypothesis, at all future time steps the edges between these two groups, and the edges with both endpoints in $R_1^{(c)}$ are fixed. Concerning the edges with both endpoints in $V \setminus R_1^{(c)}$, we can almost study this subgraph independently. That's because the effect of $R_1^{(c)}$ on $V \setminus R_1^{(c)}$ is completely predictable: it always increases the degree of all nodes by the exact same amount. The same reasoning applies for $R_{|G^{(c)}|}^{(c)}$.

More formally, by Property 1, for all $t \geq c$ it holds that $R_1^{(t)} \subseteq R_1^{(t+1)}$. This means that the nodes in $R_1^{(c)}$ are always connected to every node after time c . As a result, for all $u \in V \setminus R_1^{(c)}$ it holds that their degree in the induced subgraph

$G^{(t)} \setminus R_1^{(c)}$ is $d_{G^{(t)}}(u) - |R_1^{(c)}|$. As each interaction set contains all pairs of nodes, the decision for the existence of an edge uv , where $u, v \in G^{(t)} \setminus R_1^{(c)}$ only relies on:

$$\mathcal{E}^{(t)}(u, v) = f(d_{G^{(t)} \setminus R_1^{(c)}}(u) + |R_1^{(c)}|, d_{G^{(t)} \setminus R_1^{(c)}}(v) + |R_1^{(c)}|) \geq \beta$$

which can be written as:

$$\mathcal{E}^{(t)}(u, v) = g(d_{G^{(t)} \setminus R_1^{(c)}}(u), d_{G^{(t)} \setminus R_1^{(c)}}(v)) \geq \beta$$

where

$$g(x, y) = f(x + |R_1^{(c)}|, y + |R_1^{(c)}|)$$

Clearly, g is a proper function assuming that f is a proper function. Thus, the choice of whether the edge exists between u and v is equivalent between $G^{(t)}$ and $G^{(t)} \setminus R_1^{(c)}$ by appropriately changing f to g . But due to our hypothesis $G^{(c)} \setminus R_1^{(c)}$ stabilizes, and thus $G^{(c)}$ also stabilizes in the same number of steps. Note that we need not compute g since this is only an analytical construction; the dynamic process continues as defined. The proof of the second part of the lemma is similar in idea but much simpler since function f does not change due to the fact that the removed nodes have degree 0. \square

The following theorem establishes that this (α, β) -Dynamics stabilizes in linear time.

Theorem 5. *When $\alpha = \beta$, f is proper, $\mathcal{E}(u, v) = f(d_{G^{(t)}}(u), d_{G^{(t)}}(v))$, and all interaction sets $C^{(t)}$ contain all $\binom{n}{2}$ possible pairwise interactions, (α, β) -Dynamics stabilizes on given $G^{(0)}$ in at most $|G^{(0)}| + 1$ steps.*

Proof. By Property 3 we have that $|G^{(1)}| \leq |G^{(0)}|$. Therefore, it suffices to prove that (α, β) -Dynamics stabilizes in at most $|G^{(1)}| + 1$ steps, or equivalently that it stabilizes in at most $|G^{(1)}|$ steps after time 1; for technical reasons, we prove that for any $t_0 > 0$, (α, β) -Dynamics stabilizes in at most $|G^{(t_0)}|$ steps after t_0 . This is necessary for some of the needed tools to work (for example Lemma 4, which doesn't work for time 0).

We prove our claim inductively, on the number of equivalence classes at time t_0 . Recall that each interaction set contains all pairs of nodes. For the base case, if $|G^{(t_0)}| = 1$, then we have a regular graph. If $f(d(R_1^{(t_0)}), d(R_1^{(t_0)})) < \beta$, we get that $G^{(t_0+1)}$ is the null graph \bar{K}_n , which indeed stabilizes because $f(d(R_1^{(t_0+1)}), d(R_1^{(t_0+1)})) = f(0, 0) \leq f(d(R_1^{(t_0)}), d(R_1^{(t_0)})) < \beta$. Similarly, if $f(d(R_1^{(t_0)}), d(R_1^{(t_0)})) \geq \beta$ we get that $G^{(t_0+1)}$ is the complete graph K_n , which stabilizes because $f(d(R_1^{(t_0+1)}), d(R_1^{(t_0+1)})) = f(n-1, n-1)$, which is at least $f(d(R_1^{(t_0)}), d(R_1^{(t_0)})) \geq \beta$, due to f being proper.

For the inductive step, suppose that $|G^{(t_0)}| > 1$. If $|G^{(t_0+1)}| < |G^{(t_0)}|$, then the lemma follows by our inductive hypothesis. From now on we thus assume that $|G^{(t_0+1)}| = |G^{(t_0)}|$. We discern two cases, namely whether $f(n-1, 0) < \beta$ or $f(n-1, 0) \geq \beta$.

We begin with the case $f(n-1, 0) < \beta$. If at some time step $t \geq t_0$ it holds that $d(R_{|G^{(t)}|}^{(t)}) = 0$, then the assumption that $f(n-1, 0) < \beta$ guarantees that for all $t' \geq t$ it still holds that $d(R_{|G^{(t')}|}^{(t')}) = 0$. To see this, notice that if it does not hold, then there exists a minimal $t' > t$ such that a node $u \in R_{|G^{(t)}|}^{(t)}$ has degree $d^{(t')}(u) > 0$. But this means that there exists some vertex $v \neq u$ such that $f(d^{(t'-1)}(v), d^{(t'-1)}(u)) = f(d^{(t'-1)}(v), 0) \geq \beta$. But since $d^{(t'-1)}(v) \leq n-1$, and $f(n-1, 0) < \beta$, we reach a contradiction.

If $d(R_{|G^{(t_0+1)}|}^{(t_0+1)}) = 0$, then the observation of the previous paragraph shows that we can use Lemma 5. The graph we get by removing the nodes of degree 0 has one less equivalence class and by the inductive hypothesis it stabilizes in $|G^{(t_0+1)}| - 1$ steps; Lemma 5 thus ensures stabilization of $G^{(t_0)}$ in $1 + (|G^{(t_0+1)}| - 1) = |G^{(t_0+1)}| = |G^{(t_0)}|$ steps. Notice that if $d(R_{|G^{(t_0)}|}^{(t_0)}) = 0$, then we would also get $d(R_{|G^{(t_0+1)}|}^{(t_0+1)}) = 0$.

Therefore, we are only left with the case where $|G^{(t_0+1)}| = |G^{(t_0)}|$ and no node has degree 0, neither in $G^{(t_0)}$ nor in $G^{(t_0+1)}$. By Lemma 4, each of the $|G^{(t_0)}|$ equivalence classes at time t_0 has only $|G^{(t_0)}| + 1$ possible values for its degree, and, by definition, no two classes have the same degree. However, one of these values is 0, which we ruled out for any equivalence class, meaning that there are only $|G^{(t_0)}|$ possible values for the $|G^{(t_0)}|$ pairwise disjoint degrees. The same argument can be made for $t_0 + 1$. However, by Property 4 and Lemma 4, we get that the possible values for both time steps are the same, concluding that for all $i \in \{1, \dots, |G^{(t_0)}|\}$, we have $d(R_i^{(t_0)}) = d(R_i^{(t_0+1)})$. Then Lemma 4 implies that the two graphs are equal, and thus we have stabilization in 0 steps.

The case $f(n-1, 0) \geq \beta$ is completely similar, by using the degree $n-1$ in place of degree 0. □

5 (α, β) -Dynamics Stabilization for Arbitrary Interaction Sets

In this section, we prove stabilization (with no speed bound) for any $\alpha \leq \beta$ in an adversarial setting where the set of interaction sets \mathcal{S} may be completely arbitrary subject to the fairness condition. In addition, we further generalize by changing the definition of potential, from $\mathcal{E}(u, v) = f(d_{G^{(t)}}(u), d_{G^{(t)}}(v))$ to $\mathcal{E}(u, v) = f(g_{G^{(t)}}(u), g_{G^{(t)}}(v))$, for a family of functions $g_G : \mathbb{R}^k \rightarrow \mathbb{R}, k \in \mathbb{N}$. We call a function $g_G(u)$ *degree-like* if it only depends on the neighborhood $N_G(u)$ of node u and has the following property: assuming that the neighborhood of node u at time t is $N_{G^{(t)}}(u)$, and the neighborhood of v at time t' is $N_{G^{(t')}}(v)$, and $N_{G^{(t)}}(u) \supseteq N_{G^{(t')}}(v)$, then we require that $g_{G^{(t)}}(u) \geq g_{G^{(t')}}(v)$. The reason we extend the notion of degree is to represent more interesting rules as shown in the toy model of social dynamics of Section 7.

The potential function is computationally symmetric since f is proper and g is common for u and v . The protocol in Section 4 is a special case of this

protocol, where g is the degree of the node, each interaction set contains all $\binom{n}{2}$ possible pairwise interactions at each time step and $\alpha = \beta$. We first need the following definition:

Definition 3. A pair (t, D) is $|D| - \text{Done}$ if $t \in \mathbb{N}$, $D \subseteq V$ and $\forall u \in D$ it holds that their neighborhood does not change after time t . That is, $N_{G^{(t')}}(u) = N_{G^{(t)}}(u)$, for $t' \geq t$.

Our stabilization proof repeatedly detects $|D| - \text{Done}$ pairs with increasing $|D|$. When $D = V$, all neighborhoods do not change, and thus the process stabilizes.

Lemma 6. If there exists a $|D| - \text{Done}$ pair (t, D) at round t with $|D| < |V|$, then $\exists t' > t$ such that at round t' there exists a $(|D| + 1) - \text{Done}$ pair (t', D') .

Proof. The core idea is to find a time-step t_1 where a node $u \notin D$ maximizes g , as specified in the next paragraph; if u never drops any edge in subsequent time steps, we prove that its neighborhood is stabilized, and we extend D by u ; if it drops an edge with a node w , this node w is not able to preserve any other edge, due to the selection of u , and we are able to extend D by w .

More formally, we denote by $t_1 \geq t$ the time-step where there is some node $u \notin D$ such that $g_{G^{(t_1)}}(u) \geq g_{G^{(t'_1)}}(v)$, for all $t'_1 \geq t_1$ and $v \notin D$. If there are many choices for t_1 and u , we pick any t_1 and u such that u has the highest degree possible. Note that, later in time (say at $t'_1 > t_1$), it is entirely possible that u 's neighborhood shrinks and thus its g value drops ($g_{G^{(t'_1)}}(u) < g_{G^{(t_1)}}(u)$). It is guaranteed that t_1 exists, as there are finitely many graphs with $|V|$ nodes, and finitely many nodes. Thus, there are finitely many values of $g_G(u)$ to appear after time t . Additionally, the fairness condition guarantees that the pairwise interaction between u and v will be eventually activated, for any v .

If u never drops any edge after t_1 , then its neighborhood can only grow or stay the same. But if its neighborhood grows, due to the properties of function g , its value will not drop and the degree of u will increase. However, the way we picked u and t_1 does not allow this. We conclude that the neighborhood of u does not change after time t_1 , and thus we can extend D by $\{u\}$, that is $(t_1, D \cup \{u\})$ is $(|D| + 1) - \text{Done}$. Else, if u drops an edge after t_1 , let $t_2 > t_1$ be the first time step that a neighbor w of u in $G^{(t_2-1)}$ is not a neighbor of u in $G^{(t_2)}$. Since u 's neighborhood stays the same until $t_2 - 1$, it follows that $g_{G^{(t_1)}}(u) = g_{G^{(t_2-1)}}(u)$. The neighborhood of w does not grow at subsequent time steps, that is $N_{G^{(t'_2)}}(w) \supseteq N_{G^{(t_2+1)}}(w)$, $t'_2 \geq t_2 - 1$. To prove this, we show that w never forms a new edge after $t_2 - 1$. Suppose it does at $t'_2 + 1$ for the first time. Then w forms an edge with some node $v \notin D$, due to the definition of D . However, we know that $\beta \geq \alpha > f(g_{G^{(t_2-1)}}(u), g_{G^{(t_2-1)}}(w)) = f(g_{G^{(t_1)}}(u), g_{G^{(t_2-1)}}(w)) \geq f(g_{G^{(t'_2)}}(v), g_{G^{(t'_2)}}(w))$, due to f being non-decreasing, g being degree-like, and the definition of u and t_1 . Thus, an edge between v and w cannot be formed.

We conclude that the neighborhood of w can only shrink after time t_2 . But there are only finitely many options for the neighborhood of w , and thus there is a time $t_3 \geq t_2$ where the neighborhood of w is the same in all subsequent graphs. Therefore, we can extend D by $\{w\}$, that is $(t_3, D \cup \{w\})$ is $(|D| + 1) - \text{Done}$. \square

Theorem 6. *For $\mathcal{E}(u, v) = f(g_{G^{(t)}}(u), g_{G^{(t)}}(v))$, (α, β) -Dynamics stabilizes for any $\alpha \leq \beta$, proper function f , degree-like function g and arbitrary set of interaction sets \mathcal{S} subject to the fairness condition.*

Proof. It trivially holds that $(0, \emptyset)$ is 0 – Done. By applying Lemma 6 once, we increase the size of D by 1. Thus, by applying it $|V|$ times, we end up with a $|V|$ – Done pair (t, V) . Since all neighborhoods stay the same for all future steps, $G^{(t')} = G^{(t)}$ for all $t' \geq t$. \square

Theorem 6 can directly prove stabilization of the protocol in Section 3.

6 Turing-Completeness

In this section we describe the (α, β) -Dynamics that is able to simulate Rule 110, a one-dimensional Cellular Automaton (CA) that Cook proved to be Turing-Complete [13]. Thus, we prove that (α, β) -Dynamics is Turing-Complete as well, meaning that it is computationally universal since it can simulate any Turing machine (or in other terms any algorithm).

We first provide a short discussion on CA and Rule 110.

Cellular Automata and Rule 110: A one-dimensional cellular automaton, or, as called by Wolfram, an elementary cellular automaton, is a discrete model of computation. It consists of a one-dimensional grid of infinitely many cells, each containing a binary value. The value of all cells is updated synchronously, in discrete time steps. Each cell updates its value based on its own value and the values of its two neighboring cells.

Since the new value of each cell depends on 3 binary values, there are only 8 different cases for this update. We write 001 for the case where the left neighbor's value and the current value of a cell is 0 while the right neighbor's value is 1, 101 for the case where both neighbors have value 1 while the current value is 0, and so on. Wolfram proposed the following numbering scheme for elementary cellular automata. Suppose we create a binary number whose most significant bit is the updated value of a cell in case 111, the second most significant bit is the updated value in case 110, and so on until the least significant bit, the updated value in case 000. If we acquire number X by translating this binary number to decimal, then this particular cellular automaton is *Rule X*.

Therefore, Rule 110 is the cellular automaton corresponding to the binary number 01101110; simply put, the updated value of a cell is equal to its right neighbor's value, if its current value is 0. Else, it is 0 iff both its neighbors have value 1. What is interesting about Rule 110 is that although it is very easy to describe, Cook proved it to be Turing-Complete [13]. One shall think of the initial configuration of the cells to contain both the program and its input; if the Turing machine corresponding to the program would halt on this input, then Rule 110 stabilizes to a state that keeps on repeating forever. From this state, one is able to directly retrieve what the Turing machine would output. This allows us to prove Turing Completeness for some model of computation by just

showing that it is able to simulate Rule 110, which is much simpler than a Turing machine.

Definition 4. *Rule 110 is a one-dimensional CA. Let $cell^{(t)}(i)$ be the binary value of the i -th cell at time t . If $cell^{(t)}(i) = 0$, then $cell^{(t+1)}(i) = cell^{(t)}(i + 1)$. Else, $cell^{(t+1)}(i)$ is 0 if $cell^{(t)}(i - 1) = cell^{(t)}(i + 1) = 1$, and 1 otherwise.*

In more detail about this section's result, we design a potential function, a transformation from the initial configuration of Rule 110 to a graph $G^{(0)}$ with a constant number of nodes per cell of Rule 110, and an injection from cells of Rule 110 to pairs of nodes in $G^{(0)}$. Our claim is that the value of a particular cell of Rule 110 at time t is 1 if and only if the corresponding pair of nodes is connected in $G^{(t)}$.

We note that this does not imply that we can reach any desirable graph G_d . In fact, certain pairs of nodes (not corresponding to any cell of Rule 110) are either always connected or always disconnected, in all graphs $G^{(t)}$.

Let $CN^{(t)}(u, v) = |N_{G^{(t)}}(u) \cap N_{G^{(t)}}(v)|$ be the number of common neighbors of u and v at time t , and $CE^{(t)}(u, v) = |E(CN^{(t)}(u, v))|$ be the number of edges between the common neighbors of u and v at time t . We also introduce $F^{(t)}(u, v) = CN^{(t)}(u, v) + |E^{(t)}(u, v)|$ to make the presentation clearer.

For the following simulation we assume w.l.o.g. that $\alpha = \beta$ and that the \mathcal{S} contains all possible $\binom{n}{2}$ interactions, for all time steps. The potential between nodes u and v is defined as follows:

$$\mathcal{E}^{(t)}(u, v) = \begin{cases} \beta + 60 + CE^{(t)}(u, v) - CN^{(t)}(u, v) & \text{if } 66 \leq F^{(t)}(u, v) \leq 70 \\ \beta + 12 - CE^{(t)}(u, v) & \text{if } F^{(t)}(u, v) = 71 \\ \beta - |E^{(t)}(u, v)| & \text{if } 40 \leq CN^{(t)}(u, v) \leq 41 \\ \beta - 1 + |E^{(t)}(u, v)| & \text{otherwise} \end{cases}$$

The first 2 branches are the ones that are actually related to Rule 110, and are used only in Lemma 8. As we sketch later, our construction uses certain gadgets which allow us to simulate cells of Rule 110; some of these gadgets always follow the behavior dictated by the first branch, and the rest of them always follow the behavior dictated by the second. The rest of the branches are only used in Lemma 7 and ensure technical details, namely that some pairs of nodes always flip the status of their connection (Branch 3), effectively providing us with a clock, and some of them always preserve it (Branch 4).

As required, computing the function only uses a constant number of words in the working memory, which have logarithmic size in bits compared to the input memory (which contains the neighborhoods of u and v), and requires polynomial time in the size of the input memory. For example, to compute $CN^{(t)}(u, v)$, one could iterate over all pairs $u'v'$ such that $u \in N_{G^{(t)}}(u)$, $v \in N_{G^{(t)}}(v)$, and increment a counter initially set to zero, every time $u' = v'$. Similarly, to compute $CE^{(t)}(u, v)$, one can iterate over quadruples u', u'', v', v'' and increment a counter whenever $u' = v'$, $u'' = v''$ and there exists an edge

between u' and u'' . Additionally, the potential function only depends on nodes at a constant distance (at most 1) from either u or v , and it is network-agnostic (not assuming access on the topology of $G^{(0)}$). Finally, it is computationally symmetric and thus the protocol is consistent.

Informally, our simulation of Rule 110 consists of the following steps. First, we design a primitive cell-gadget (henceforth *PCG*) that stores binary values, but fails to capture Rule 110 since it doesn't distinguish between the left and the right cell. Then, by making use of the *PCG* as a building block, we build the main cell-gadget (henceforth *CG*) that is used to simulate a single cell of the CA. Then, each time step from Rule 110 is simulated using 2 rounds of the (α, β) -Dynamics; on the first round, some *PCGs* acquire their proper value while on the second round, the rest of the *PCGs* copy the correct value from the ones that already acquired it. Finally, the two steps are merged into one in order to achieve stabilization of the dynamics when Rule 110 has also stabilized.

For clarity purposes, we slightly abuse notation, and we count the rounds of the (α, β) -Dynamics by multiples of 0.5 instead of 1. Thus, we write that the sequence of configurations is $G^{(0)}, G^{(0.5)}, G^{(1)}, \dots$, where configurations $G^{(t+0.5)}$, for $t \in \mathbb{N}$, are transitional states of the network and have no correspondence with cell states of the CA.

In order to construct the *PCG* and the *CG*, we first construct two auxiliary gadgets, the always-on (x, y) -gadget and the flip (x, y) -gadget. The always-on (x, y) -gadget is simply a clique of 22 nodes. 20 of them have no edges to other nodes in the network, while 2 of them (namely x and y) may be connected with other nodes. The flip (x, y) -gadget is basically two always-on (x, y) -gadgets, with nodes x and y being the same for both gadgets, with the exception that the edge between x and y may not exist. See Figure 1 for both of these gadgets. We later show that, under certain conditions, the edge between x and y always exists in an always-on gadget, and flips its state at each time step, in a flip gadget.

A *PCG* consists of a pair of nodes hl , such that the existence of an edge between them corresponds to value 1 and otherwise it corresponds to value 0, and 60 auxiliary nodes a_1, \dots, a_{60} . Furthermore, for each of the 120 pairs of the form ha_i and la_i , there exists a corresponding (h, a_i) and (l, a_i) -flip gadget. When we have two different *PCGs*, say A and B , we write $A(h), A(l), A(a_1), \dots, A(a_{60})$ for the nodes of A and similarly $B(h), B(l), B(a_1), \dots, B(a_{60})$ for the nodes of B . We write $A^{(t)}$ to denote the value of A at time t ; in other words $A^{(t)} = |E^{(t)}(A(h), A(l))|$.

In order to connect two different *PCGs* (say A and B) we add 4 always-on gadgets: the always-on $(A(h), B(h))$ gadget, the always-on $(A(h), B(l))$ gadget, the always-on $(A(l), B(h))$ gadget and the always-on $(A(l), B(l))$ gadget, as shown in Figure 1. Intuitively, this relates $CE^{(t)}(A(h), A(l))$ to the sum of values of the connected *PCGs*.

The i -th *CG* that corresponds to the i -th cell (we write $CG(i)$) consists of 4 *PCGs*, which we identify as $A_1(i), A_2(i), B_1(i)$ and $B_2(i)$. At time $t = 0$, the edge in each flip gadget of $A_1(i), A_2(i)$ exists, while the edge in each flip gadget of $B_1(i), B_2(i)$ does not exist. We connect each $A_j(i)$ with each $B_k(i)$ (4

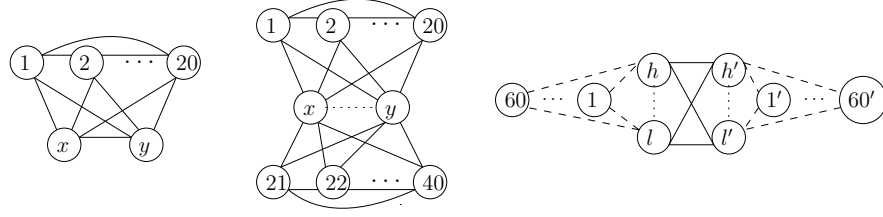


Fig. 1. To the left, we have an always-on (x, y) gadget. In the middle, we have a flip (x, y) gadget; the dotted line between x, y denotes that this particular edge may or may not exist. To the right, we have two $PCGs$. The dashed lines denote flip gadgets, the dotted lines denote that these particular edges may or may not exist. The continuous lines denote always-on gadgets; these 4 always-on gadgets is how we connect $PCGs$.

connections in total, where each connection uses 4 always-on gadgets, as depicted in Figure 1). In order to connect $CG(i)$ (cell i) with $CG(i + 1)$ (cell $i + 1$) we connect $A_j(i)$ with $A_j(i + 1)$, and $A_j(i)$ with $B_j(i + 1)$, as shown in Figure 2. A CG is said to have value 0 if all 4 of its $PCGs$ are set to 0 and 1 if all $PCGs$ are set to 1. We guarantee that no other case can occur in $G^{(t)}$, $t \in \mathbb{N}$, although this is not guaranteed for the intermediate configurations $G^{(t+0.5)}$, $t \in \mathbb{N}$.

To conclude the construction of $G^{(0)}$, each cell of Rule 110 corresponds to a CG in $G^{(0)}$, and neighboring cells have their corresponding CGs connected. Finally, we set the value of its CG (that is the value of its 4 $PCGs$) equal to the initial value of the corresponding cell.

Notice that all our gadgets are defined for a single time-step, namely for $t = 0$. One could imagine that in subsequent time-steps, nodes contained in the same gadget in $G^{(0)}$ are no longer connected in the same way (effectively destroying the gadget), or even that new gadgets are formed. The following lemma shows that this is not the case. Informally, it shows that no new gadgets are created, and that the only difference between graphs at different time steps concern edges that do not destroy the existing gadgets. For example, in the definition of a flip gadget, there is only one pair of nodes (its two special nodes) for which it does not matter whether they share an edge or not; the lemma shows that between nodes that belonged in the same flip gadget in $G^{(0)}$, only this special pair may change its connection (existence or not of an edge between them) through time.

Lemma 7. *If there exists a flip (x, y) -gadget connected to an $A_j(i)$ PCG in $G^{(0)}$, then the edge xy at time t exists if and only if $t \in \mathbb{N} \cup \{0\}$. Similarly, if there exists a flip (x, y) -gadget connected to a $B_j(i)$ PCG in $G^{(0)}$, then the edge xy exists if and only if $t \notin \mathbb{N} \cup \{0\}$. Finally, all other edges exist at any time step if and only if they exist in $G^{(0)}$, with the exception of edges between h, l nodes of a PCG .*

Proof. We prove our claim using induction on the time step t . The base case $t = 0$ holds by the construction of $G^{(0)}$. Suppose our claim holds for time step $t - 0.5$, we show that it also holds for time step t . We first prove our claim for

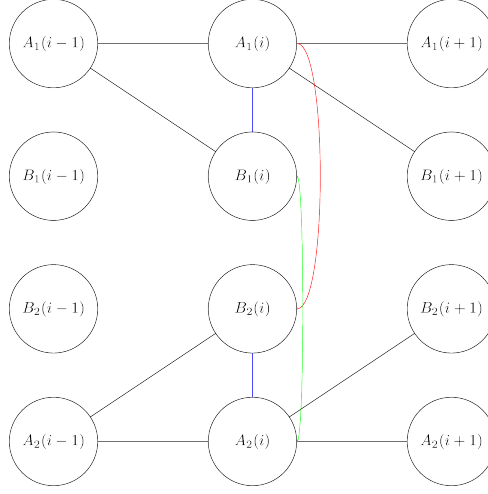


Fig. 2. Each circle represents a *PCG* and each line represents a connection between *PCGs* (4 always-on gadgets) as in Figure 1. Only connections relevant to $A_1(i), A_2(i), B_1(i), B_2(i)$ are shown. The 4 connections in the second column (again each one is 4 always-on gadgets) are internal connections of $CG(i)$. All other connections correspond to how $CG(i-1)$ is connected with $CG(i)$ and $CG(i)$ is connected with $CG(i+1)$. We prove that these connections are always preserved.

the pairs of nodes sharing an edge in $G^{(0)}$, except for the pairs *hl* of *PCGs*, as the Lemma makes no claim about them. Notice that it suffices to argue about always-on and flip gadgets, as this is the only way we added non-*hl* edges to $G^{(0)}$.

Let us first focus on the nodes that, at $G^{(0)}$, are contained in the same always-on (x, y) -gadget. We argue that for any two such nodes x', y' , the edge between them exists on time step t , except possibly for the xy edge; more formally, the unordered pair $\{x', y'\}$ is assumed to be different from $\{x, y\}$. By definition of the always-on gadget and the inductive hypothesis, x' and y' have exactly 20 common neighbors in $G^{(t-0.5)}$, and thus they continue sharing an edge in $G^{(t)}$. Concerning the x, y nodes of the gadget, we take cases depending on whether they also happen to be the two special endpoints of a flip (x, y) gadget in $G^{(0)}$ or not. In the former case, by the inductive hypothesis, they have between 40 and 41 common neighbors in $G^{(t-0.5)}$, depending on the existence of edges not defined by our induction hypothesis. Thus, these edges always flip their status at t , as the lemma dictates. In the latter case they have between 20 and 24 common neighbors in $G^{(t-0.5)}$, depending on the existence of edges not defined by our induction hypothesis. Thus, these edges continue to exist in $G^{(t)}$.

We are only left to argue about pairs of nodes with no edge connecting them in $G^{(0)}$. For a non-existent edge to become existent, it must be that its two endpoints have at least 40 common neighbors, by the potential function. But, by the inductive hypothesis and the construction of $G^{(0)}$, this only happens for

endpoints x, y for which there exists a flip (x, y) -gadget (we already argued about such cases) and for endpoints h, l of some PCG (for which case our lemma does not claim anything). Thus, no other edge is ever created. \square

Our next step is to discuss how hl edges of $PCGs$ change. The number of common neighbors of an hl pair of an $A_j(i)$ is $CN^{(t)}(h, l) = 70$, for all integer time steps t and valid i, j , as it has 5 neighboring $PCGs$ (each contributing 2), and 60 auxiliary nodes within the PCG (by Lemma 7). For non-integer time steps $t + 0.5, t \in \mathbb{N} \cup \{0\}$, by Lemma 7, the 60 auxiliary nodes are not connected with h and l , and so $CN^{(t)}(h, l) = 10$. Similarly, the number of common neighbors of an hl pair of a $B_j(i)$ is $CN^{(t)}(h, l) = 66$, for all non-integer t and valid i, j , and $CN^{(t)}(h, l) = 6$ for integer t .

Furthermore, for all t , it holds that $CE^{(t)}(A_j(i)(h), A_j(i)(l)) = 8 + A_j^{(t)}(i - 1) + B_1^{(t)}(i) + B_2^{(t)}(i) + A_j^{(t)}(i + 1) + B_j^{(t)}(i + 1)$, as the edges between common neighbors are the internal edges of connected $PCGs$, plus the connection between $A_j^{(t)}(i - 1)$ and $B_j^{(t)}(i)$ (4 edges), plus the connection between $A_j^{(t)}(i + 1)$ and $B_j^{(t)}(i + 1)$ (4 edges). Similarly, for a $B_j(i)$ we have that $CE^{(t)}(B_j(i)) = 4 + A_j^{(t)}(i - 1) + A_1^{(t)}(i) + A_2^{(t)}(i)$.

Lemma 8. *It holds that $A_j^{(t)}(i) = B_j^{(t)}(i) = cell^{(t)}(i)$ for $j \in \{1, 2\}$ and all $i, t \in \mathbb{N}$.*

Proof. It holds that $A_j^{(0)}(i) = B_j^{(0)}(i) = cell^{(0)}(i)$ by the initialization of our construction. Suppose that $A_j^{(t)}(i) = B_j^{(t)}(i) = cell^{(t)}(i)$ for an integer $t \geq 0$. By using induction we show that the lemma holds for time $t + 1$.

First of all, we prove that $A_j^{(t+0.5)}(i) = cell^{(t+1)}(i)$. If $cell^{(t)}(i) = 0$, then it holds that $cell^{(t+1)}(i) = cell^{(t)}(i + 1) = A_j^{(t)}(i + 1) = B_j^{(t)}(i + 1)$, due to our inductive hypothesis. Furthermore, due to our inductive hypothesis it holds that $A_j^{(t)}(i) = B_1^{(t)}(i) = B_2^{(t)}(i) = 0$. Thus, since $CN^{(t)}(A_j(i)(h), A_j(i)(l)) = 70$ and $|E^{(t)}(A_j(i)(h), A_j(i)(l))| = 0$ (there is no edge between the h, l nodes in $A_j(i)$) the potential between the pair of nodes is $\mathcal{E}^{(t)}(A_j(i)(h), A_j(i)(l)) = CE^{(t)}(A_j(i)(h), A_j(i)(l)) + \beta - 10$. To find the potential of the pair of nodes $A_j(i)$ we compute:

$$\begin{aligned} CE^{(t)}(A_j(i)(h), A_j(i)(l)) &= \\ 8 + A_j^{(t)}(i - 1) + B_1^{(t)}(i) + B_2^{(t)}(i) + A_j^{(t)}(i + 1) + B_j^{(t)}(i + 1) &= \\ 8 + cell^{(t)}(i - 1) + 2cell^{(t)}(i + 1) \end{aligned}$$

Thus, it follows that the potential of $A_j(i)(h)$ and $A_j(i)(l)$ is $\beta + cell^{(t)}(i - 1) + 2cell^{(t)}(i + 1) - 2$, which is at least β if and only if $cell^{(t)}(i + 1) = 1$. Thus, in the case where $cell^{(t)}(i) = 0$ we proved that indeed it holds that $A_j^{(t+0.5)}(i) = cell^{(t+1)}(i)$.

We use a similar reasoning for the case where $cell^{(t)}(i) = 1$. In particular, since $CN^{(t)}(A_j(i)) = 70$ and $|E^{(t)}(A_j(i))| = 1$ (there is an edge between the h, l

nodes in $A_j(i)$ the potential between the pair of nodes is $\mathcal{E}^{(t)}(A_j(i)(h), A_j(i)(l)) = \beta + 12 - CE^{(t)}(A_j(i))$. We compute:

$$\begin{aligned} CE^{(t)}(A_j(i)(h), A_j(i)(h)) = \\ 8 + A_j^{(t)}(i-1) + B_1^{(t)}(i) + B_2^{(t)}(i) + A_j^{(t)}(i+1) + B_j^{(t)}(i+1) = \\ 10 + cell^{(t)}(i-1) + 2cell^{(t)}(i+1) \end{aligned}$$

Thus, it follows that the potential of $A_j(i)(h)$ and $A_j(i)(l)$ is $\mathcal{E}^{(t)}(A_j(i)) = \beta + 2 - cell^{(t)}(i-1) - 2cell^{(t)}(i+1)$, which is less than β if and only if $cell^{(t)}(i-1) = cell^{(t)}(i+1) = 1$. This proves that $A_j^{(t+0.5)}(i) = cell^{(t+1)}(i)$.

Since $CN^{(t+0.5)}(A_j(i)(h), A_j(i)(l)) = 10$, we have $A_j^{(t+1)}(i) = cell^{(t+1)}(i)$. Therefore $A_j^{(t+1)}(i) = A_j^{(t+0.5)}(i)$. Similarly, since $CN^{(t)}(B_j(i)(h), B_j(i)(l)) = 6$, it holds that $B_j^{(t+0.5)}(i) = B_j^{(t)}(i)$.

The potential of $B_j(i)$ at $t+0.5$ is (recall that $CN^{(t)}(B_j(i)(h), B_j(i)(l)) = 66$):

$$\begin{aligned} \mathcal{E}^{(t+0.5)}(B_j(i)(h), B_j(i)(l)) = CE^{(t+0.5)}(B_j(i)(h), B_j(i)(l)) + \beta - 6 = \\ \beta + 2A_j^{(t+0.5)}(i) + A_j^{(t+0.5)}(i-1) - 2 \end{aligned}$$

This is at least β if and only if $A_j^{(t+0.5)}(i) = 1$, which proves that $B_j^{(t+1)}(i) = cell^{(t+1)}(i)$. \square

The following corollary is a straightforward consequence of this lemma.

Corollary 1. *It holds that $cell^{(t)}(i) = CG^{(t)}(i)$.*

The above construction simulates Rule 110. The only problem is that it takes two time steps to simulate a single time step of Rule 110, meaning that even if Rule 110 converges, our construction infinitely flips between two different configurations, due to the flip gadgets, and as a result it does not stabilize. To overcome this problem, we use the aforementioned construction and make changes that allow us to remove the intermediate steps in the simulation, that is the steps $t + 0.5, t \in \mathbb{N} \cup \{0\}$.

Theorem 7. *The (α, β) -Dynamics is Turing-Complete.*

Proof. By Lemma 7 and Corollary 1 it follows that Rule 110 would be correctly simulated by the particular (α, β) -Dynamics constructed above, if the transitional non-integer time steps were missing, and thus the convergence of an instance of Rule 110 would mean the stabilization of the constructed (α, β) -Dynamics. To achieve this, we simulate the two steps of the constructed (α, β) -Dynamics in one step based on the observation that the defined potential for each pair of nodes x, y depends only on the graph induced by the nodes at distance at most 1 from either x or y . As a result, if nodes x and y at time step t could 'guess' what this induced graph would look like in the transitional, non-integer,

time step $t + 0.5$, they could immediately use this to deduce their potential in time step $t + 0.5$.

We are left to argue about how x and y get information about this induced graph. Notice that a node u may get connected with another node v at any time step t' only if $d^{(t'-0.5)}(u, v) \leq 2$. Thus, in order for x and y to be able at time step t , to know this induced graph at time step $t + 0.5$, it suffices to compute the connections at time $t + 0.5$ between all nodes u for which $\min\{d^{(t)}(x, v), d^{(t)}(y, v)\} \leq 2$. In turn, in order to compute such a potential, they need to have information about nodes at distance 1 from these nodes that lie at distance at most 2. In conclusion, it suffices to access all nodes at distance at most 3 at time t ; notice that by Lemma 7 and the construction of $G^{(0)}$, there is a constant number of such nodes, for any pair xy and time t .

Therefore, the new (α, β) -Dynamics starts with the same $G^{(0)}$ and computes the new potential between any two nodes x, y in two conceptual steps. In the first step, it uses the old potential function, and information from nodes at distance at most 3 from either of them, to compute how the graph induced by all nodes u for which $\min\{d^{(t)}(x, u), d^{(t)}(y, u)\} \leq 2$ would look like at time $t + 0.5$. Then, by applying the old potential function on this computed graph, it computes the final potential between x and y , effectively simulating the transitional time step. Therefore, the potential function only acquires information from nodes at a constant distance (at most 3) from either x or y , as required. It is also clear that it is network-agnostic, or in other words that it is designed without access to the topology of $G^{(0)}$.

To see that this new potential function is computationally symmetric, notice that the auxiliary graph is computed both by x and by y by accessing the same information and using the same computationally symmetric potential function, meaning both x and y end up with the same auxiliary graph. Then, they apply the same computationally symmetric function on this graph, meaning that they acquire the same value.

Finally, we have shown that at any time step, each node only has a constant number of neighbors. Therefore, the auxiliary graph also has a constant number of nodes, and we only need a constant number of words to represent the auxiliary graph. The computation of each such edge in the auxiliary graph, as well as the final computation, uses the old potential function; all these computations are using the same working memory. Thus, the new potential function respects the restriction of having a working memory at most (asymptotically) logarithmic in size, compared to the input memory (which contains the neighborhoods of u and v), since the old potential function does as well. The time needed is also polynomial in the input size, as the same holds for the time needed to compute the old potential function.

□

7 Extensions

We briefly discuss two straightforward extensions of (α, β) -Dynamics and provide related examples. To begin with, we can add static information to nodes/edges

(e.g., weights). This information is encoded by the potential function and does not change with time. The degree-like function defined in Section 5 can be used to assign a time-independent importance factor (e.g. a known centrality measure in $G^{(0)}$) while letting $g(u)$ be the sum of these factors of nodes in $N_{G^{(t)}}(u)$. To demonstrate it, we provide a small example with a toy model inspired by Structural Balance Theory [18] of networks with friendship and enmity relations [5]. This example is more reminiscent of population dynamics rather than distributed protocols. Assume that the network of agents corresponds to people (nodes) with friendship relations (edges). Each agent v is defined by how nice she is $n(v)$, how extrovert she is $x(v)$ as well as by the set of her enemies $\mathcal{EN}(v)$. We wish to design a model that captures how friendships change in this setting when enemies do not change⁵ as well as when friendships are lost in case of very few common friends, while friends are made in the opposite case.

To define the social dynamics we need to define the interaction sets and the potential function that essentially describe our toy model. The interaction sets captures the interactions between the agents enforced by the model. This toy model is only for the purpose of highlighting our convergence results and we do not claim to realistically capture certain social phenomena. The interaction sets are defined as follows: (a) if two agents u and v are enemies then they never become friends (no pairwise interaction between them in $C^{(t)}$, for any t), (b) if two agents u and v are not connected by an edge in $G^{(t)}$ (they are not friends) but their distance is at most the sum of their extrovertedness, then they interact - that is, if at time t it holds that $1 < \text{dist}(u, v) \leq x(u) + x(v)$ then there is an edge uv in $C^{(t)}$, (c) if two agents are connected by an edge in $G^{(t)}$, then there is a pairwise interaction between them in $C^{(t)}$ if their number of common friends is $\leq \gamma$. If their common friends are $> \gamma$ then their friendship is strong and it will not be affected at this round, and thus no edge in $C^{(t)}$ is introduced. This concludes the description of \mathcal{S} .

As for the potential function, we define the potential between u and v in $G^{(t)}$ to be $\mathcal{E}(u, v) = (n(u) + \sum_{w \in N(u)} n(w)) + (n(v) + \sum_{w \in N(v)} n(w))$, capturing our intuition that friendships are created or stopped based on how nice the two agents and their neighbors are. This is a computationally symmetric function and thus the protocol is consistent. The function g corresponds to the sum of the niceness of a node plus the niceness of its neighbors and thus it is degree-like. The function f is proper since it is a simple sum between u and v w.r.t. the output of the function g in each node. Thus, (α, β) -Dynamics on this social network stabilizes by Theorem 6 (the proof holds without any modification, even in this somewhat extended version of (α, β) -Dynamics). Theorem 6 also allows us to add any rules with respect to \mathcal{S} like imposing a maximum number of friends, allowing for additional random connections (to achieve long-range interaction), etc. Similarly, we can change the definition of potential and still prove stabilization as long as the assumptions of Theorem 6 are valid. If these assumptions are violated, as it would be in the case of a potential function that

⁵ The permanence of enmity is in fact not exactly compatible with structural balance theory on networks.

applies to a subset of neighbors (e.g., common neighbors between u and v), then a new analysis is required to prove stabilization, if stabilization can be reached. Finally, the interaction sets allows us to remove the assumption of permanence on enmity by allowing under certain conditions particular pairwise interactions, thus dynamically changing the set $\mathcal{EN}(v)$.

Another straightforward generalization is to allow for general stateless protocols \mathcal{A} targeting at providing algorithmic solutions for specific problems. An example of such a generalization is given below for constructing a spanning star. We show in simple terms the stateless approach when compared to state-dependent approaches for constructing a network (e.g., Network Constructors model [23,24]). In some sense, we already provide such an example of explicit network construction in the case of the α -core. We assume that in each time step t a pairwise interaction uv is chosen uniformly at random and we set $C^{(t)} = \{uv\}$. In [23] they provide a simple protocol that uses states on the nodes, which, starting from the null graph, constructs the spanning star in optimal $\Theta(n^2 \log n)$ expected time. We discuss a protocol \mathcal{A} that computes a spanning star starting from any network. It is reminiscent of the random copying method [20] for generating power law networks. It would be interesting to find out whether hub-and-spoke networks (essentially star networks) can be generated by some similar social process. In this case, the probability of choosing pairwise interactions should be related to the degree of the involved nodes.

To describe the protocol let u and v be two nodes that interact at time t as determined by $C^{(t)}$. Assume w.l.o.g. that $d_G^{(t)}(u) > d_G^{(t)}(v)$. Then, the protocol dictates that all edges of v are to be moved to u . In case $d_G^{(t)}(u) = d_G^{(t)}(v) \neq 1$, we break symmetry (symmetry breaking was also needed in [23], which was implemented with the help of the scheduler) by tossing a fair coin in each node as to which node is going to transfer its neighbors⁶. The nodes communicate the result of their toss and if found equal no change happens in the current round, otherwise we again move all edges from the one node to the other. In both cases if no edge exists between u and v , an edge uv is added. Finally, if $d_G^{(t)}(u) = d_G^{(t)}(v) = 1$, then no change happens.

On the positive side, the difference of this protocol to the one given in [23] is that no state dynamics are used and we start from an arbitrary network. However, on the negative side, a pairwise interaction between u and v may affect all nodes up to distance 2 since the move of all these edges can affect nodes of distance 2 from u and v . Correctness is proved based on the observation that in each round either one node becomes a leaf transferring all its adjacent nodes to the other node, or nothing happens. The latter case can only happen infinitely often if the graph has become a star, otherwise by fairness, at some point of time an interaction that makes progress will happen. Because of this stalling, the time complexity analysis is more involved but we conjecture only by a poly-logarithmic factor away from the one in [23]. Although no such definition have

⁶ The coin toss can be implemented by denoting randomly in the active pair uv one node as the initiator and the other as the responder, as it usually happens in population protocols[2].

been provided for our model, the described protocol is self-stabilizing and silent. Looking at more detail in these protocols constitutes very interesting research direction.

8 Conclusion

(α, β) -Dynamics are stateless structural dynamics of a network. The protocol allows for two thresholds that affect the existence of the edges in the pairwise interactions determined by an interaction set at each time step. Since the dynamics are purely structural, the output of the protocol is another network, and thus (α, β) -Dynamics can be considered as a network transformation process. Such a process for example has been used in [35] to detect communities. In fact, the authors wondered whether conditional convergence could be proved. It is a matter of technical details to show that for regular networks one can choose α and β such that the protocol never stabilizes.

For future research, it would be very interesting to look at the notion of parallel time in (α, β) -Dynamics. Another interesting research direction is to see the effect of higher order structural interactions as well as look at how the model is affected when messages are restricted in size. Finally, inspired by the computation of the α -core in Section 3, a very interesting question is to look at more involved problems w.r.t. emergent behavior from simple protocols.

9 Acknowledgements

The authors would like to thank the anonymous reviewers for their useful insights and suggestions.

References

1. Afek, Y., Alon, N., Barad, O., Hornstein, E., Barkai, N., Bar-Joseph, Z.: A biological solution to a fundamental distributed computing problem. *Science* **331**, 183–5 (2011)
2. Alistarh, D., Gelashvili, R.: Recent algorithmic advances in population protocols. *SIGACT News* **49**(3), 63–73 (2018)
3. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M., Peralta, R.: Computation in networks of passively mobile finite-state sensors. *Distributed Computing* **18**(4), 235–253 (2006)
4. Angluin, D., Aspnes, J., Eisenstat, D., Ruppert, E.: The computational power of population protocols. *Distributed Computing* **20**(4), 279–304 (2007)
5. Antal, T., Krapivsky, P., Redner, S.: Social balance on networks: The dynamics of friendship and enmity. *Physica D: Nonlinear Phenomena* **224**(1), 130 – 136 (2006)
6. Arrighi, P., Dowek, G.: Free fall and cellular automata. In: *Developments in Computational Models. EPTCS*, vol. 204, pp. 1–10 (2015)
7. Barabási, A., Albert, R.: Emergence of scaling in random networks. *Science* **286**(5439), 509–512 (1999)

8. Batagelj, V., Mrvar, A., Zaversnik, M.: Partitioning approach to visualization of large graphs. In: Graph Drawing. LNCS, vol. 1731, pp. 90–97. Springer (1999)
9. Becchetti, L., Clementi, A., Natale, E.: Consensus dynamics: An overview. SIGACT News **51**(1), 58–104 (2020)
10. Becchetti, L., Clementi, A., Natale, E., Pasquale, F., Trevisan, L.: Find your place: Simple distributed algorithms for community detection. In: SODA '17. p. 940–959 (2017)
11. Casteigts, A., Chaumette, S., Ferreira, A.: Characterizing topological assumptions of distributed algorithms in dynamic networks. In: Structural Information and Communication Complexity. pp. 126–140. Springer (2010)
12. Casteigts, A., Flocchini, P., Quattrociocchi, W., Santoro, N.: Time-varying graphs and dynamic networks. International Journal of Parallel, Emergent and Distributed Systems **27**(5), 387–408 (2012)
13. Cook, M.: Universality in elementary cellular automata. Complex Systems **15**(1) (2004)
14. Feldmann, M., Scheideler, C., Schmid, S.: Survey on algorithms for self-stabilizing overlay networks. ACM Comput. Surv. **53**(4) (jul 2020). <https://doi.org/10.1145/3397190>, <https://doi.org/10.1145/3397190>
15. Flajolet, P., Gardy, D., Thimonier, L.: Birthday paradox, coupon collectors, caching algorithms and self-organizing search. Discret. Appl. Math. **39**(3), 207–229 (1992), [https://doi.org/10.1016/0166-218X\(92\)90177-C](https://doi.org/10.1016/0166-218X(92)90177-C)
16. Gadouleau, M.: On the influence of the interaction graph on a finite dynamical system. Natural Computing **19**(1), 15–28 (2020)
17. Gärtner, B., Zehmakan, A.: (biased) majority rule cellular automata. CoRR **abs/1711.10920** (2017)
18. Heider, F.: The psychology of interpersonal relations. John Wiley and Sons (1958)
19. Ilachinski, A.: Structurally Dynamic Cellular Automata, pp. 29–71. Springer US (2018)
20. Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tompkins, A., Upfal, E.: The web as a graph. In: PODS '00. p. 1–10 (2000)
21. Litovsky, I., Métivier, Y., Sopena, E.: Graph relabelling systems and distributed algorithms. Handbook of Graph Grammars and Computing by Graph Transformation **3**, 1–56 (1999)
22. Malliaros, F., Giatsidis, C., Papadopoulos, A., Vazirgiannis, M.: The core decomposition of networks: theory, algorithms and applications. VLDB Journal **29**(1), 61–92 (2020)
23. Michail, O., Spirakis, P.: Simple and efficient local codes for distributed stable network construction. Distributed Computing **29**(3), 207–237 (Jun 2016)
24. Michail, O., Spirakis, P.: Network constructors: A model for programmable matter. In: SOFSEM '17. pp. 15–34 (2017)
25. Michail, O., Spirakis, P.: Elements of the theory of dynamic networks. Communications of the ACM **61**(2), 72–72 (Jan 2018)
26. Nakagaki, T., Yamada, H., Toth, A.: Maze-solving by an amoeboid organism. Nature **407**, 470 (09 2000)
27. O'Dell, R., Wattenhofer, R.: Information dissemination in highly dynamic graphs. In: DIALM-POMC '05. pp. 104–110 (2005)
28. Saidani, S.: Self-reconfigurable robots topodynamic. In: Proceedings of the 2004 IEEE Int. Conference on Robotics and Automation, ICRA. vol. 3, pp. 2883 – 2887 Vol.3 (01 2004)

29. Sayama, H., Laramée, C.: Generative Network Automata: A Generalized Framework for Modeling Adaptive Network Dynamics Using Graph Rewritings, pp. 311–332. Springer (2009)
30. Smith, D., Onnela, J.P., Lee, C., Fricker, M., Johnson, N.: Network automata: Coupling structure and function in dynamic networks. *Advances in Complex Systems* **14**(03), 317–339 (2011)
31. Szekeres, G., Wilf, H.: An inequality for the chromatic number of a graph. *Journal of Combinatorial Theory* **4**, 1–3 (1968)
32. Watts, D., Strogatz, S.: Collective dynamics of 'small-world' networks. *Nature* **393**(6684), 440–442 (1998)
33. Wolfram, S.: *A New Kind of Science*. Wolfram Media Inc. (2002)
34. Wolfram, S.: A class of models with the potential to represent fundamental physics (2020)
35. Zhang, Y., Wang, J., Wang, Y., Zhou, L.: Parallel community detection on large networks with propinquity dynamics. In: *ACM SIGKDD '09*. pp. 997–1006 (2009)