Math. Struct. in Comp. Science (2012), *vol.* 22, *pp.* 788–815.
doi:10.1017/S0960129511000533

Adversarial scheduling in discrete models of social dynamics[†]

GABRIEL ISTRATE[‡], MADHAV V. MARATHE[§] and S. S. RAVI[¶]

[‡]*Center for the Study of Complexity, Babeş-Bolyai University,*
Str. Fântânele 30, cam. A-14,
Cluj-Napoca, RO-400294, Romania
and
e-Austria Research Institute, C. Coposu 4, cam 045B,
Timişoara, RO-300223, Romania
Email: gabrielistrate@acm.org
[§]*Network Dynamics and Simulation Science Laboratory,*
Virginia Bio-Informatics Institute,
1880 Pratt Drive Building XV, Blacksburg, VA 24061, U.S.A.
Email: mmarathe@vbi.vt.edu
[¶]*Computer Science Dept., S.U.N.Y. Albany,*
Albany, NY 12222, U.S.A.
Email: ravi@cs.albany.edu

Received 2 November 2010

In this paper we advocate the study of discrete models of social dynamics under *adversarial scheduling*. The approach we propose forms part of a foundational basis for a *generative approach to social science* (Epstein 2007). We highlight the feasibility of the adversarial scheduling approach by using it to study the *Prisoners's Dilemma Game with Pavlov update*, a dynamics that has already been investigated under random update in Kittock (1994), Dyer *et al.* (2002), Mossel and Roch (2006) and Dyer and Velumailum (2011). The model is specified by letting players at the nodes of an underlying graph G repeatedly play the Prisoner's Dilemma against their neighbours. The players adapt their strategies based on the past behaviour of their opponents by applying the so-called win–stay lose–shift strategy. With random scheduling, starting from any initial configuration, the system reaches the fixed point in which all players cooperate with high probability. On the other hand, under adversarial scheduling the following results hold:

— A scheduler that can select *both* game participants can preclude the system from reaching the unique fixed point on most graph topologies.
— A non-adaptive scheduler that is only allowed to choose *one* of the participants is no more powerful than a random scheduler. With this restriction, even an adaptive scheduler is not significantly more powerful than the random scheduler, provided it is 'reasonably fair'.

[†] This work has been supported by NSF PetaApps Grant OCI-0904844, DTRA R&D Grant HDTRA1-0901-0017, NSF Netse CNS-1011769, and NSF SDCI OCI-1032677, by BMWF, and by a grant from and a project on Postdoctoral programs for sustainable development in a knowledge-based society, contract POSDRU/89/1.5/ S/60189, cofinanced by the European Social Fund through the Sectorial Operational Program for Human Resource Development 2007-2013.

1. Introduction

The study of Computational Social Sciences (Lazer 2009) has greatly benefited in recent years from viewing social dynamics from a discrete dynamical systems perspective. This has opened the social realm to the class of models usually investigated in the complex systems literature: cellular automata (Hegselmann and Flache 1998) and their asynchronous counterpart, sequential dynamical systems (Mortveit and Reidys 2007; Barrett *et al.* 2005), spin glasses and related models from the statistical physics literature (see, for example, Castellano *et al.* (2009)), discrete game-theoretic models (see, for example, Wilhite (2006)) and a myriad of computational and computational agent-based simulations in the social sciences (Ballot and Weisbuch 2000; Tesfatsion and Judd 2006; Gilbert and Troizch 2005). The discrete nature of these models has brought aspects of social dynamics into the realm of computability theory, making them natural candidates for studying issues such as computability (see, for example, Velupillai (2000)) or computational complexity.

The central problem of such models is, of course, their *validation and validation*: How does one understand and interpret properties of such systems, be they obtained through mathematical analysis or computational simulations, and how do we make sure that the insights we derive from such models are 'correct'? This question is as much philosophical as it is practical, given that game theory and agent-based simulations are emerging as tools for guiding political decision-making (see, for example, Barrett *et al.* (2005), Epstein *et al.* (2004) and Eubank *et al.* (2004); the TRANSIMS project is also relevant[†]).

A possible answer to the issue described above is that results characterising the dynamical properties of such models provide insights (and possible causal explanations) for features observed in 'real-world' social dynamics. To give just one example, the primary intuition behind the (mathematical) concept of *stochastic stability* in evolutionary game theory (Young 1998) is that a small amount of 'noise' (or, equivalently, small deviations from rationality) in game dynamics can solve the so-called equilibrium selection problem by helping the system 'focus' on one particular equilibrium. Thus, deviations from rationality provide a mechanism that can compensate for the apparent underspecification of rational social behaviour provided by game-theoretic equilibrium concepts.

In a similar spirit to the possible answer outlined above, Epstein (see Epstein (1999; 2007); see also Axtell and Epstein (1996)) has advocated a generative approach to (agent-based) social science. The goal is to *explain* a given social phenomenon by *generating* it using multiagent simulations[‡]. Related concerns have been recently voiced throughout so-called *analytical sociology* (Hedström and Bearman 2009), with a particular emphasis on *mechanism-based explanations* (Hedström 2005; Hedström and Swedberg 2006). Mechanism-based explanations citeglennan1996mechanisms,bunge1997mechanism are some of the more promising sociological approaches to 'dissecting the social' (Boudon 1998; Elster 1998; Machamer *et al.* 2000; Craver 2006). Presumably, a more disciplined version of the generative approach would aim to identify plausible social mechanisms underlying

[†] See `http://code.google.com/p/transims/` for details.

[‡] Compare this with Epstein (2007, Chapter 1), : 'if you didn't grow it [the social phenomenon, n.n.], you didn't explain its emergence'.

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

the social phenomena of interest, with mathematical results/computer simulations serving as 'virtual laboratories'.

The generative approach to social science is not without critics, such as a rather large number of game theorists and other mathematically inclined social scientists who are skeptical of the social simulation approach. However, even an economist generally sympathetic to computational experiments could write (Mirowski 2002, page 531): 'Forty years on, the first complaint of a member of the audience for an economic simulation is: Why didn't you report that variant run of your simulation? Where is the sensitivity analysis? How many simulations does it take to make an argument? One often finds that such specific criticisms are often parried by loose instrumentalist notions, such as "we interpret the question, can you explain it? as asking, can you grow it ?" ((Axtell and Epstein 1996), p. 177). On those grounds there would never have been any pressing societal need for molecular biology [...]'.

In any case, whatever ones opinion is of Epstein's program and, more generally, of what validating a discrete social model means, it is hard to deny the fact that generation is at least *necessary* for mathematical/computational accounts of social phenomena, and making the analysis of generative models more scientific is desirable. Such a task will certainly involve making sure that the conclusions we derive are robust to small variations in model specification.

The goal of this paper is to focus on one particular aspect of the robustness of agent-based simulations and evolutionary game-theoretic models: *agent scheduling*, that is, the order in which agents update their strategies. This is clearly a significant aspect of social dynamics, one that crucially differentiates agent societies from discrete physical systems: agents have free will and can decide to act in response to perceived characteristics of social dynamics. As such, the activation order is often complex, most of the time being endogenously coupled to the specification of the dynamics.

In contrast, many of the mathematical models in the literature take the update order as exogenous. In fact, for reasons of mathematical tractability, the most two popular alternatives seem to be:

— **synchronous update:** *every* player can update at every moment (in either discrete or continuous time) – this is the model implicitly used by 'large population' models in evolutionary game theory and by cellular automata.

— **random activation:** agents are vertices of a (hyper)graph. At each step, we choose a (hyper)edge uniformly at random and all players corresponding to this hyperedge are updated using the local update function.

Instead of postulating one of these two update mechanisms, we advocate the study of social dynamics under an approach we call *adversarial scheduling*, which is inspired in part by the theory of *self-stabilisation of distributed systems* from the computer science literature. This approach takes the endogenising of the scheduler to its limits in a controlled way, identifying the structural reasons that make the original result hold. The validity of the original result then involves deciding whether the endogenous schedule satisfies these structural reasons 'in the real world'.

As an example of adversarial scheduling, we will study its application to the Iterated Prisoner's Dilemma game with a win–stay lose–shift strategy. This dynamics (originally motivated by the colearning model in Shoham and Tennenholtz (1997)) has received substantial attention in the game-theoretic literature(Kittock 1994; Dyer *et al.* 2002; Mossel and Roch 2006; Dyer and Velumailum 2011).

We are, of course, far from being the first to recognise the crucial role of activation order on the properties of social dynamics However, what we advocate is a (somewhat) more systematic approach, based on the following principles:

(i) *Start with a 'base case' result P, under a particular scheduling model.*
(ii) *Identify several structural properties of the scheduling model that impact the validity of P*. Ideally, these properties should be selected by a careful examination of the proof of P, which should reveal their importance.
(iii) *Of these properties, identify those that (alone or in combinations) are* **necessary (sufficient)** *for the validity of P*. Correspondingly, *those that can be abstracted out without affecting the validity of P*.
(iv) The process outlined so far can be continued by recursively applying steps (i)-(iii). In doing this *we may need to reformulate the original statement in a way that makes it hold under larger classes of schedulers*, thus making it more robust. The precise reformulations normally arise from inspecting the cases when the proof of P fails in an adversarial setting.

We do *not* pretend that the above program is always feasible. On the other hand, since starting this research, we have found ever more central examples of game-theoretic models where such a program can be accomplished. More importantly, we plan to argue in a future paper that the program can be realised to a certain extent for some computational agent-based models as well, which is the motivating goal of this research.

1.1. *Organisation of the paper*

The present paper does not do justice (nor does it pretend to) to the myriad philosophical issues pertaining to our work (for example, emergence in social systems, the specification and role of social mechanisms, the micro–macro link in social theory). We plan to discuss these issues further in future research. Instead, this paper is primarily mathematically-oriented, presenting a proof-of-concept of our approach Its intended benefits are multiple. Our (modest) goal is to show that **these benefits do not come at the expense of mathematical tractability**: at least in some cases, adversarial scheduling (as outlined in the four point approach above) is feasible and can lead to interesting results.

In the next section, we review the basic model and its properties under random scheduling. Our results and the related literature are presented in Section 3. In Section 4 we discuss the role of fairness in scheduling. Results for edge-schedulers (those that are able to choose both players in the Prisoners' Dilemma) are presented in Section 5. In Sections 6 and 7, we consider the case of node-schedulers, which are only able to choose one partner in the game. In Section 8, we present some experimental results on convergence speed. Finally, in Section 9, we discuss the results and present our conclusions.

2. Preliminaries

We begin by defining two classes of graphs that we will frequently consider in this paper:

— The **line graph** L_n, $n \geqslant 1$ consists of n vertices $v_1, \ldots v_n$ and edges (v_i, v_{i+1}), $1 \leqslant i \leqslant n-1$. We will use *Line* to denote the set of all line graphs.
— A **star graph** $Star_n$, $n \geqslant 1$, is the complete bipartite graph $K_{1,n}$. We will use *Star* to denote the set of all star graphs.

2.1. *Basic model: the Prisoner's Dilemma with Pavlov dynamics*

In this section we describe the basic mathematical model for the Prisoner's Dilemma with Pavlov dynamics (PDPD). It first arose as a very simple case of Shoham and Tennenholtz' colearning framework, which is a model for the emergence of social norms (conventions) in artificial societies. In the interests of simplicity, we will omit a detailed discussion of the motivation and the detailed game-theoretic formulation of the model (see Kittock (1994), Mossel and Roch (2006) and Dyer *et al.* (2002) for information on these issues) and simply specify it as a discrete dynamical system.

We are given an undirected graph $G(V, E)$, $|V| = n$ and $|E| = m$. Each vertex $v \in V$ represents an agent. Each agent has a label from the set $\{0, 1\}$. These labels denote the strategies that the players follow: 0 can also be equivalently viewed as *cooperation* and 1 can be viewed as *defection*. Without loss of generality, we assume that G is connected, otherwise the dynamics will reduce to independent dynamics on the connected components of G. We will also assume that the graph contains at least two edges.

Time will be discrete. At time $t = 0$, all nodes are assigned a label from $\{0, 1\}$. At each subsequent step, certain nodes/agents change their label (strategy) according to the rules given below. We will use $x_t(v)$ to denote the label of node v. At each step $t + 1$ an edge $e = (u, v)$ is selected according to some rule and the states of u and v are updated as follows:

$$x_{(t+1)}(u) \leftarrow (x_t(u) + x_t(v)) \pmod 2$$
$$x_{(t+1)}(v) \leftarrow (x_t(u) + x_t(v)) \pmod 2$$

We will use $\mathbf{X}_t$ to denote the vector $(x_t(v_1), \ldots x_t(v_n))$ representing the states of the nodes $v_1, \ldots v_n$. This will sometimes be referred to as the *global configuration*, and we will sometimes omit the subscript t for ease of exposition when its value is clear from context. With this terminology, each step of the dynamics can be viewed as a global update function F. It takes as input an element $e = (v_i, v_j) \in E$ and a global configuration $\mathbf{X} = (x(v_1), \ldots x(v_n))$, and returns the next global configuration $\mathbf{Y} = (y(v_1), \ldots y(v_n))$, which is computed as follows:

— $\forall v_k$, such that $k \neq i$ and $k \neq j$, $y(v_k) = x(v_k)$; and
— $y(v_i) = y(v_j) = (x(v_i) + x(v_j)) \pmod 2$.

In this case, $\mathbf{Y}$ is said to be reachable from $\mathbf{X}$ in one step. A global configuration $\mathbf{X}$ is said to be a fixed point if $\forall e \in E$, we have $F(\mathbf{X}, e) = \mathbf{X}$. It is easy to see that the dynamical system studied here has a unique fixed point $\mathbf{0} = (0, \ldots, 0)$. Following the

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

dynamical systems literature, a configuration $\mathbf{X}$ is called a *Garden of Eden configuration* if the configuration is not reachable from any other configuration. For the rest of this paper, we will use $\mathbf{X}, \mathbf{Y}, \ldots$ to denote global configurations. An instance of PDPD can thus be represented as (G, f), where G is the underlying interaction graph and f is the local function associated with each node. Since f will always be fixed, in the rest of this paper, we will simply specify an instance by G.

2.2. *The base-case result*

The following property of the PDPD is easily seen to hold under random matching: for all interaction graphs G with no isolated vertices, *the system converges with probability* $1 - o(1)$ *to the 'all zeros' configuration*, which we will denote by $\mathbf{0}$ from now on. With a slight abuse of convention, we will refer to this event as *self-stabilisation*. This is the property we will study in an adversarial setting.

We will also be interested in the *convergence time* of the dynamics. Under random scheduling, Dyer *et al.* (2002) proved that the number of steps needed to self-stabilise is $O(n \log n)$ on C_n (the simple cycle on n nodes) and exponential in n on K_n (the complete graph on n nodes). The convergence time was further investigated by Mossel and Roch (Mossel and Roch 2006). Recently, the dynamics was generalised in Dyer and Velumailum (2011) to 'rational Pavlov dynamics', and their results provide evidence for the existence of a phase transition between polynomial and exponential mixing time on a cycle.

2.3. *Types of schedulers*

A *schedule* $\mathbf{S}$ is specified as an infinite string over E, that is, $\mathbf{S} \in E^*$. Given a schedule $\mathbf{S} = (e_1, e_2, \ldots, e_t, \ldots)$, the graph G and an initial configuration $\mathbf{I}$, the system evolves as follows. At time $t = 0$ the system is in state $\mathbf{I}$. At time t, we pick the tth edge from $\mathbf{S}$, and call this edge e_t. If the configuration at the beginning of time t is $\mathbf{X}$, then the configuration $\mathbf{Y}$ at the beginning of time $t + 1$ is given by $\mathbf{Y} \leftarrow F(\mathbf{X}, e_t)$. The iterated global transition function F^* is defined as

$$F^*(\mathbf{I}, (e_1, e_2, \ldots, e_t, \ldots)) \equiv F^*(F(\mathbf{I}, e_1), (e_2, \ldots, e_t, \ldots)).$$

We say that a system G *self-stabilises* for a given initial configuration $\mathbf{I}$ and a schedule $\mathbf{S} = (e_1, \ldots, e_t, \ldots)$[†] if $\exists t \geqslant 1$ such that the system starting in $\mathbf{I}$ reaches the (unique fixed-point) configuration $\mathbf{0}$ after t time steps, that is, $\mathbf{0} \leftarrow F^*(\mathbf{I}, (e_1, \ldots, e_t))$. G is said to self-stabilise for a schedule $\mathbf{S}$ if G eventually reaches a fixed point when started at any initial configuration I, that is, $\forall I,\ F^*(\mathbf{I}, \mathbf{S}) \to \mathbf{0}$. Conversely, a schedule $\mathbf{S}$ can preclude self-stabilisation of G if $\exists I$ such that $F^*(\mathbf{I}, \mathbf{S})$ never reaches $\mathbf{0}$. Given a set of schedules, a scheduler is simply an algorithm (possibly randomised) that chooses a schedule. The schedulers considered here are all polynomial time algorithms and the set of feasible schedules is described below.

[†] Note that e_i and e_j in the sequence need not be distinct.

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

Schedulers can be adaptive or non-adaptive. An adaptive scheduler decides on the next edge or node based on the current global configuration. A non-adaptive scheduler decides on a schedule in advance by looking at the graph (and possibly the initial configuration). This schedule is then fixed for the rest of the dynamic process. One particularly restricted class of *non-adaptive* schedules is a fixed permutation of nodes/edges repeated periodically and independent of the initial state of the system. For node updates, this is the model employed in *sequential dynamical systems* (Barrett *et al.* 2003; Barrett *et al.* 2005). There are several distinctions that can be made with respect to the power of a scheduler. The first concerns the number of players that the scheduler is able to choose. There are two possibilities:

— An *edge-daemon* (or edge-scheduler) is able to choose *both* players of the interacting pair. In other words, an edge daemon constructs **S** by selecting edges $e_i \in E$ in some order.
— A *node-daemon* (or node-scheduler) can choose only one of the players. We can let this player choose its partner. A natural model is to consider the case where the partner is chosen uniformly at random among the neighbours of the first player. In such a case, we say that the node-scheduler model has *random choice*, and that such a scheduler is a random-choice node-scheduler.

3. Summary of results and related work

Our results can be summarised as follows:

— Unsurprisingly, some degree of fairness is necessary to obtain self-stabilisation in an adversarial setting.
— The power given to the scheduler makes a big difference in determining whether the system does or does not self-stabilise:
  (i) if the scheduler can *exogenously* choose *both* participants in the game, then (Theorems 4.5 and 5.1) it can preclude convergence on most graphs, even when bounded by fairness constraints.
  (ii) On the other hand, schedulers that allow a limited amount of *endogeneity in agent interactions*[†], by only choosing *one* of the participants, are no more powerful than the random scheduler (Theorem 5.2) when non-adaptive, and are not significantly more powerful (Theorem 6.1) when adaptive but 'reasonably fair'.

We also investigate experimentally (in Section 8) the convergence time of the colearning dynamics for a few non-adaptive schedulers for a case where the convergence time for the random scheduler is known rigorously.

Our approach is naturally related to the *theory of self-stabilisation of distributed systems* (Dolev 2000). Multi-agent systems, like the ones considered in the evolutionary game dynamics, have many of the characteristics of a distributed system: a number of entities (the agents) capable of performing certain *computations* (changing their strategies) based

[†] The importance of this property has been recognised (Vriend 2006) in the agent-based simulation literature.

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

on *local information*. In fact, randomised models of this type (including the model we study in this paper) have been considered recently in the context of self-stabilisation (Fribourg *et al.* 2006). Also, Angluin *et al.*'s *population protocol* (see Aspnes and Ruppert (2007) for a survey) is relevant to our work, though the focus of that theory is slightly different, its main interest being computation by direct interaction in spite of unpredictable patterns of scheduling. The system we study can indeed be viewed as a special case of a population protocol. This connection with game theory recently became even stronger – see Jaggard *et al.* (2000) and Bournez *et al.* (2009). Remarkably, Bournez *et al.* (2009) studied computation in the population protocol model using two-person symmetric games, and considered the case of *Pavlov protocols*, motivated in part by our base-case scenario from Dyer *et al.* (2002).

There are, however, a number of differences. First, in self-stabilisation the computational entities (processors) are capable of executing a wide-range of activities (subject to certain constraints, such as the requirement that all processors run the same program in the context of so-called *uniform self-stabilising systems*). The aim of such systems is to achieve a certain goal (legal state) in spite of transient errors and malicious scheduling. In contrast, in our setup, there is no 'goal', the computations are fixed, and restricted to steps of the evolutionary dynamics. The only source of uncertainty arises from the scheduling model. A second difference is in the nature of the update rule. Usually, in self-stabilising systems there is a difference between *enabled* processors, which intend to take a step, and those that actually take it. Such a notion is not as natural in the context of game-theoretic models.

As mentioned earlier, the dynamics can be easily recast in the context of the Prisoner's Dilemma: let 1 encode 'defection' and 0 encode 'cooperation'. Then the update rule corresponds to the so-called *win–stay lose–shift* (Posch 1997), or Pavlov's strategy. This rule specifies that agents defect on the next move precisely when the strategy they used in the last interaction was different from the strategy used by the other player. It was the object of much attention in the context of the Iterated Prisoner's Dilemma (Axelrod 1984; Nowak and Sigmund 1993; Axelrod 1997). Related versions of the dynamics have an even longer history in the Psychology literature, where they were proposed to model the emergence of cooperation in situations where players do not have precise knowledge of the payoffs of the game in which they are participating, and might even be unaware they are playing a game. Sidowski (1957) proposed the 'minimal social situation' (MSS), which is a two-person experiment representing an extremely simple form of interaction between two agents. MSS was first viewed as a game in Thibaut and Kelley (1959), where it was called 'Mutual Fate Control'. An explanation for the empirical findings in Thibaut and Kelley (1959) was proposed in Radloff *et al.* (1962), which raised the possibility that players act according to Pavlov dynamics. MSS was generalised to multiplayer games in Coleman *et al.* (1991) and Colman (2005), which gave a mathematical characterisation of the emergence of cooperation.

Last, but not least, we should note that in addition to the social science interpretations, our model is naturally related to the spin systems studied in statistical physics, and particularly to kinetic population models (Krapivsky *et al.* 2010); in fact, it can be mapped directly onto such a model (we thank Eli Ben-Naim for this comment). In the same spirit,

Mossel and Roch (2006) remarked that the IPD dynamics behaves heuristically like the closely related *contact process*. So our results can be interpreted as studying the impact of update dynamics in such (physically motivated) systems.

4. Fairness in scheduling

A necessary restriction on the schedulers we will be concerned with is *fairness*. In self-stabilisation this is usually taken to mean that each node is updated infinitely often in an infinite schedule. We will also consider notions of *bounded fairness*, for which the following is a natural definition.

Definition 4.1. Let $b \geqslant 1$. A scheduler that can choose one item among a set of m elements is *(worst-case) b-fair* if for every agent x, no other agent is scheduled more than b times between two consecutive schedulings of x.

It is easy to see that a 1-fair edge-scheduler chooses a fixed permutation of edges and uses this as a periodic schedule. A 1-fair node daemon selects a fixed permutation of nodes, and for each node, selects a random neighbour and repeats the same permutation (with possibly different partners) periodically.

To investigate the properties of random schedulers, a natural starting point is to consider the fairness of probabilistic schedulers. For such schedulers, the worst-case fairness in Definition 4.1 is far too restrictive.

Definition 4.2. A probabilistic scheduler is *weakly fair* if for any node x and any initial schedule y, the probability that x will eventually be scheduled, given that the scheduler selected nodes according to y, is positive.

The random scheduler is weakly fair. Not every scheduler is weakly fair, and a scheduler need not be weakly fair to make the system self-stabilise. On the other hand, the base-case result does not extend to the adversarial setting when weak fairness is not required.

Theorem 4.3. The following are true:

(i) There exists an edge-scheduler that is not weakly fair and that makes the system self-stabilise no matter what its starting configuration is.
(ii) For any graph G, there exists an edge/node-scheduler that is not weakly fair and that prevents the system from self-stabilising on some initial configuration.

A restatement of Theorem 4.3 is that weak fairness is necessary to preclude some 'degenerate' schedulers, like the ones we construct for the proof of part (ii).

Proof.

(i) Consider an edge-scheduler that works as follows:
   — Choose an edge e that has not yet self-stabilised, that is, at least one of its endpoints is **1**.
   — Turn nodes of e to **0** by playing e twice.
   — The scheduler never schedules e subsequently.

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

(ii) Consider a node(edge)-scheduler that repeatedly schedules the same node (edge). It is easy to see that the system does not self-stabilise unless the graph consists of a single edge (a star in the case of a node-scheduler when the centre node is the scheduled one). □

Definition 4.4. A probabilistic scheduler is $O(f(n))$-*node fair with high probability* if the following condition is satisfied while the system has not reached the fixed point:

— For any schedule W, with last scheduled node x, and every node y and every $\epsilon > 0$, there exists $C_\epsilon > 0$ such that, with probability at least $1 - \epsilon$, node y will be scheduled at most $C_\epsilon \cdot f(n)$ times before x is scheduled again.

A scheduler is *boundedly node fair with high probability* if it is $O(f(n))$-node fair with high probability for some function $f(n)$.

We should emphasise the fact that Definition 4.4 also applies to edge-schedulers, where a node x is considered to occur at stage t if some edge containing x is scheduled at that step. With these definitions we now have the following theorem.

Theorem 4.5. Let S be a (node or edge) weakly fair probabilistic scheduler such that for any initial configuration, the probability that the system self-stabilises tends to one. Then the scheduler is boundedly node fair.

Proof. We will consider node and edge-schedulers at the same time. Let $\epsilon > 0$ and $T = T(\epsilon, n)$ be an integer such that, no matter what configuration I we start the system in, the probability that the system does *not* self-stabilise (taken over all the scheduler's random choices) is at most ϵ.

We consider any configuration I' of the system after a node x has been scheduleded, and assume that I' is not the absorbing state $\mathbf{0}$. Run the system for T steps. The probability that the system does not self-stabilise is at most ϵ. On the other hand, if some node y is not played at all during the T steps, then the system has no chance to self-stabilise. It follows that the maximum number of times a given node can be scheduled before x is scheduled again is at most $T - 1$. □

5. The power of edge-schedulers

From now on we will restrict ourselves to boundedly fair schedulers. In this section we show that edge-schedulers are too powerful. Indeed, it is easy to show that there exist graphs on which even 1-fair edge-schedulers can prevent self-stabilisation. The following two results provide a modest improvement, showing that even 2-fair edge-daemons on any graph are too strong.

Theorem 5.1. Let G be an instance of PDPD. Then there exists an initial configuration I and a 2-fair edge-scheduler $\mathbf{S}$ that precludes self-stabilisation on G starting in configuration I.

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

Proof. Consider a sequence of edges $e_0, \ldots e_k$ (with repetitions allowed) such that all the following properties hold:

— every edge of G appears in the list;
— for every $i = 0, \ldots, k$, we have e_i and e_{i+1} have *exactly one* vertex in common (where $e_{k+1} = e_0$); and
— every edge appears in the sequence at most twice.

We will show that an enumeration $F(G)$ with these properties can be found for *any* connected graph with more than one edge. Such a sequence specifies in a natural way (through its periodic extension) a 2-fair edge daemon. It is easy to see that the only states that can lead to the fixed point are the fixed point and states leading to it in one step: such a state has exactly two (adjacent) ones. But such a state cannot be reached from any other state according to the previously described scheduler, since the edge that would have been 'touched' immediately before has unequal labels on its extremities, which cannot be the case after updating it.

We now have to show how to construct the enumeration $F(G)$. First we give the enumeration for the case when the graph G is a tree. In this case we perform a walk on G, listing the edges as follows: suppose the root r is connected *via* vertices $v_1, \ldots, v_k$ to subtrees $T_1, \ldots, T_k$. We then define recursively

$$F(G) = (v, t_1)F(T_1)(t_1, v)(v, t_2)F(t_2)(t_2, v)\ldots(t_{k-1}, v)(v, t_k)F(T_k)(t_k, v), \tag{1}$$

where, if the list of edges thus constructed contains two consecutive occurrences of the same edge, we eliminate the second occurrence.

We now consider the general case of a connected graph G, and let $S(G)$ be a spanning tree of G. Edges of G belong to two categories:

(i) Edges of the spanning tree $S(G)$.
(ii) Edges in $E(G) \setminus E(S(G))$.

Define $F(G)$ as the list of edges obtained from $F(S(G))$ as follows: whenever $F(S(G))$ first touches a new vertex w in G, insert the edges in $E(G) \setminus E(S(G))$ adjacent to w (in some arbitrary order); continue then with $F(S(G))$. We now claim that each edge e is listed at most twice in $F(G)$. To prove this, consider the two cases:

— $e \in E(S(G))$:
The statement follows in this case from the recursive definition (1).
— $e \in E(G) \setminus E(S(G))$:
The statement follows in this case by construction since a non-tree edge is visited only when one of its endpoints is first touched in $E(S(G))$. □

Even the most restricted edge-schedulers, 1-fair edge-schedulers, are able to preclude self-stabilisation on a large class of graphs. To see this, we define:

(i) $\mathcal{G}_1$ to be the class of graphs G that contain a cycle of length at least four.
(ii) $\mathcal{G}_2$ to be the class of graphs G that contain *no* cycles of length at least four and m, the number of edges of G is even.
(iii) $\mathcal{G}_3$ to be the class of trees with $n = 4k$ vertices.

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

Theorem 5.2. Let G be a connected graph in $\mathscr{G}_1 \cup \mathscr{G}_2 \cup \mathscr{G}_3$. Then there exists an initial configuration on G and a *1-fair* edge-schedule $\mathbf{S}$ that is able to preclude self-stabilisation on G for ever.

In other words, connected graphs for which the system self-stabilises for all 1-fair schedulers have an odd number of edges and all their cycles (if any) have length 3.

Proof. We define $\Delta_{i,j} = (d_{k,l}^{(i,j)})$, as an $n \times n$ matrix over $\mathbf{Z_2}$ with

$$d_{k,l}^{(i,j)} = \begin{cases} 1 & \text{if } (k,l) = (i,j) \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Suppose we represent configurations of the system as vectors in $\mathbf{Z}_2^n$. Taking one step of the dynamics on an arbitrary configuration $\mathbf{X}$ with the scheduled edge being (i,j) leads to configuration $\overline{\mathbf{X}} = A_{i,j} \cdot \mathbf{X}$, where matrix $A_{i,j}$ is given by

$$A_{i,j} = I_n + \Delta_{i,j} + \Delta_{j,i}. \tag{3}$$

Indeed, the only non-diagonal elements of matrix $A_{i,j}$ that are non-zero are in positions (i,j) and (j,i). This means that all elements of a configuration $\mathbf{X}$ in positions other than i, j are preserved under multiplication with $A_{i,j}$. It is easy to see that labels in positions i, j change according to the specified dynamics.

Consider a graph G with m edges, $E(G) = \{(i_1, j_1), \ldots, (i_m, j_m)\}$. The action of a 1-fair edge schedule $\mathbf{S}$ (specified by permutation π of $\{1, \ldots, m\}$) on a configuration $\mathbf{X}$ corresponds to multiplication of $\mathbf{X}$ by

$$\pi(S) = A_{i_{\pi[1]}, j_{\pi[1]}} \cdot A_{i_{\pi[2]}, j_{\pi[2]}} \cdot \ \ldots \ \cdot A_{i_{\pi[m]}, j_{\pi[m]}}. \tag{4}$$

When an edge-scheduler cannot prevent self-stabilisation, the system starting from any initial configuration reaches a fixed point. In other words, $\forall \mathbf{I} \in \mathbf{Z}_2^n \ \exists k \in \mathbf{N}$ such that $[\pi(\mathbf{S})]^k \cdot \mathbf{I} = \mathbf{0}$. Since the number of vectors $\mathbf{I}$ is finite, this is equivalent to saying that $\exists k_1 \in \mathbf{N}, \forall \mathbf{I}, \ [\pi(\mathbf{S})]^{k_1} \cdot \mathbf{X} = \mathbf{0}$. Equivalently, this means that $[\pi(\mathbf{S})]^k = \mathbf{0}$; in other words, matrix $\pi[\mathbf{S}]$ is nilpotent. Thus, what we want to show is the following.

Lemma 5.3. For any graph G there exists a schedule $\mathbf{S}$ such that the corresponding matrix $\pi[\mathbf{S}]$ is not nilpotent.

We now consider an arbitrary ordering of vertices in G and let π be the permutation corresponding to the induced lexicographic ordering of edges of G (where an edge is viewed as an ordered pair, with the vertex of lower index appearing first).

It is easy to see that

$$\Delta_{i,j} \cdot \Delta_{k,l} = \begin{cases} \Delta_{i,l} & \text{if } j = k \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Indeed, if $\Delta_{i,j} = (a_{m,n}^{(i,j)})_{m,n \geqslant 1}$, then the only way for some element $c_{m,n}$ of the product $\Delta_{i,j} \cdot \Delta_{k,l}$ to be non-zero is that there exists at least one term $d_{m,p}^{(i,j)} \cdot d_{p,n}^{(k,l)}$ that is non-zero. But this is only possible for $(i,j) = (m,p)$ and $(p,n) = (k,l)$; in other words, for $m = i$, $n = l$ and $p = j = k$, which immediately yields Equation (5).

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

Let us now consider the integer matrix $\Pi[S]$ obtained by interpreting Equations (3) and (4) as equations *over integers*. Because integer addition and multiplication commute with taking the modulo 2 value, matrix $\pi[S]$ can be obtained by applying reduction modulo 2 to every element of $\Pi[S]$.

Let $P_{i,j} = \Delta_{i,j} + \Delta_{j,i}$. From the definition of matrix $\pi(S)$ in Equation (4) and the definition of matrices $A_{i,j}$ in Equation (3), we see that $\pi[S]$ is a sum of products, with each term in a product corresponding to either a $P_{i,j}$ or to the identity matrix. Thus

$$\Pi[S] = I + \sum_{\emptyset \neq S \subseteq \{1,\ldots,m\}} \left(\prod_{k \in S} P_{i_{\pi[k]}, j_{\pi[k]}} \right). \tag{6}$$

Consider the directed graph $\overline{G}$ obtained from G by replacing every edge $\{i, j\}$ of G by two *directed* edges (i, j) and (j, i). Label every edge $e \in E(G)$ by the (unique) integer k such that $e = \{i_{\pi[k]}, j_{\pi[k]}\}$, and apply the same labelling to the two oriented versions of edge e in $\overline{G}$. Then Equation (5) shows that the non-zero products of matrices Δ are in bijective correspondence with directed paths of length two with increasing labels when read from the start to the end node of the path. Inductively generalising these observations to all sets S, we see that products in (6) are non-zero exactly when they specify a *directed path in* $\overline{G}$ (that is, a path in G) from a vertex k to a vertex l with increasing labels when read from k to l, in which case they are equal to $\Delta_{k,l}$.

Therefore, $\Pi[S] = I + C$, where $C = (c_{i,j})$ is given by

$$c_{i,j} = \begin{cases} \#\text{ of paths from } i \text{ to } j \text{ with increasing labels} & \text{if such paths exist} \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

The matrix $\pi[\mathbf{S}]$ is, of course, obtained by reducing modulo 2 the elements of $\Pi[S]$. A well-known result in linear algebra[†] is that the the characteristic polynomial of a non-zero nilpotent $n \times n$ matrix A is x^n. Thus, one strategy to show that a given matrix $\pi[\mathbf{S}]$ is not nilpotent is to make sure that for some p, $0 \leqslant p \leqslant n$, the sum s_p of its principal minors of order p is non-zero. This is equivalent to making sure that the sum of the corresponding minors of the associated *integer* matrix is odd. The proof consists of three cases, which we will consider in turn.

Case (a) $G \in \mathscr{G}_1$**:**

We will prove the following lemma.

Lemma 5.4. There exist two permutations σ_1 and σ_2 with corresponding matrices over integers $A_1 = \Pi[\sigma_1]$ and $A_2 = \Pi[\sigma_2]$ such that

$$trace(A_2) \equiv (trace(A_1) + 1) (\text{mod } 2).$$

[†] This result justified as follows. A classical result states that the characteristic and the minimal polynomial of a matrix have the same roots (with different multiplicities). But it is easy to see that the minimal polynomial of a nilpotent matrix is x^k for some $k \leqslant n$.

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

Given Lemma 5.4, the proof of Lemma 5.3 for Case (a) follows since matrices $\pi[\sigma_1]$ and $\pi[\sigma_2]$ cannot both be nilpotent. This is true since $trace(A_1)$ and $trace(A_2)$ have different parities. We will prove Lemma 5.4 using a multistep argument, combining the conclusions of Lemmas 5.5-5.7 below. Consider first the following 'basic' graphs: K_4, $K_3 \triangle K_3$ (the graph obtained by merging two triangles on a common edge), and C_n, $n \geqslant 4$.

Lemma 5.5. The conclusion of Lemma 5.4 is valid for the 'basic' graphs.

Proof. By the previous result on the value of coefficients $c_{i,j}$, the value of the trace of a matrix A can be easily computed from the number of cycles with increasing labels. Also, note the following:

— Any cycle C contributes a one to *at most* one $c_{i,i}$, for some vertex i appearing in C. This is because of the restriction on the increasing labels, which may be satisfied for at most one node of the cycle.

— Moreover, any triangle contributes a 1 to *exactly one* $c_{i,i}$. Therefore, in considering the trace of matrix A, triangles add the same quantity irrespective of the permutation, and can thus be ignored.

This observation leads to a straightforward solution when the underlying graph is a simple cycle C_n, $n \geqslant 4$, or the graph $K_3 \triangle K_3$. Note that these graphs contain a unique cycle C of length at least 4. Consider an ordering of the edges of this cycle, corresponding to moving around the cycle. We will create two labellings corresponding to this ordering. The first assigns labels 1 to $|C|$ in that order. The other labelling assigns labels $1, 2, \ldots, |C|-2, |C|, |C|-1$ in that order. It is easy to see that the first ordering contributes a 1 to exactly one diagonal element, while the second does not contribute a 1 to any element. Hence the traces of the corresponding matrices differ by exactly 1.

For graph K_4, we first label the diagonal edges by 5 and 6. There are three cycles of length 4 in the graph K_4 – one that uses no diagonal edges, and the other two using them both. For the outer cycle consisting of no diagonal edges, we consider the two orderings described in the previous case on the outer cycle C_4. As before, this shows that the traces of the corresponding matrices differ exactly by 1. Next, note that irrespective of the labelling of the non-diagonal edges, the two cycles containing the diagonal edges *cannot* be traversed in increasing label order, so they do not contribute to the trace of the associated matrix. Therefore, the result follows for graph K_4 as well. This completes the proof of Lemma 5.5. □

Lemma 5.6. Let G be a graph and G_2 be a subgraph of G induced by a subset of the vertices in G. If the conclusion of Lemma 5.5 holds for G_2, then it holds for G.

Proof. We extend a permutation of the edges in G_2 to a permutation of the edges in G through a fixed labelling of the edges in $E(G) \setminus E(G_2)$ such that the following conditions hold:

(i) The index of any edge with both end points in G_2 is strictly smaller than the index of all edges not in this class.

(ii) The index of any edge with *exactly* one end point in G_2 is strictly larger than the index of any edge not in this class.

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

The trace of the resulting matrix is determined by the cycles with strictly increasing labels. There are several types of cycles like this:

(i) Cycles in $G \setminus G_2$.
Whether such a cycle can be traversed in increasing label order *does not depend on the precise labelling on edges of* G_2 as long as the conditions of the extension are those described earlier.

(ii) Cycles containing some edges in G_2, as well as additional edges from $G \setminus G_2$.
Because of the restriction we placed on the labellings, the only such cycles that can have increasing labels are the triangles with two vertices in G_2 and one vertex in $G \setminus G_2$. Since there is a unique way to 'read' a triangle in the increasing order of the edge labels, their contribution to the total trace is equal to the number of such triangles, and *does not depend on the precise labelling of edges in* G_2, as long as the restriction on the labelling is met.

(iii) Cycles entirely contained in G_2.

We now let σ_1, σ_2 be labellings on G_2 satisfying the conclusion of Lemma 5.4 and let $\overline{\sigma_1}, \overline{\sigma_2}$ be extensions to G satisfying the stated restriction. The conclusion of the previous analysis is that a difference in the parity of the traces of matrices corresponding to the labellings σ_i on G_2 translates directly into a difference in the parity of the traces of matrices corresponding to labellings $\overline{\sigma_i}$ on G. □

Finally, we reduce the case of a general graph to that of a *base case* graph using the following result.

Lemma 5.7. Let G be a graph that contains a cycle of length $\geqslant 4$ and is minimal (that is, any induced subgraph H of G does *not* contain a cycle of length $\geqslant 4$). Then G is one of the 'basic' graphs from Lemma 5.5.

Proof. Let G be minimal with the property that it contains a cycle of length $\geqslant 4$ and n be the number of nodes in G, and let C be a cycle of length $\geqslant 4$ in G. Because of minimality, C contains all the vertices of G (otherwise G would not be minimal since one could eliminate nodes outside C). Thus C is a Hamiltonian cycle. If no other edge is present, we get the cycle C_n. Moreover, no other edge can be present unless $n = 4$ (otherwise G would contain a smaller cycle of length $\geqslant 4$ and thus would not be minimal). In this case the two possibilities are K_4 and $K_3 \triangle K_3$. □

Case (b) $G \in \mathscr{G}_2$**:**

Consider the ordering $<_{sum}$ on the edges of G such that $\{i, j\} <_{sum} \{k, l\}$ when either

$$i + j < k + l$$

or

$$i + j = k + l \text{ and } \min\{i, j\} < \min\{k, l\}.$$

In this case, $c_{k,k} = 0$ for every k except when k is the *middle-index vertex* of a triangle (that is, a triangle with vertex labels i, j, k such that $i < k < j$).

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

We infer that $s_1 = trace(A)$ is congruent (mod 2) to n plus the number of triangles in G, that is, to $m + 1$ (mod 2) (where m is the number of edges of G).

Case (c) $G \in \mathscr{G}_3$**:**

Consider the sum s_2 of principal minors of size 2 of A. In a tree, there can be only one path between a pair of nodes. Since we are counting paths with increasing labels, the only way for $a_{i,j} = a_{j,i} = 1$ to hold is that vertices i and j are adjacent. But in this case the corresponding minor is zero. It follows that s_2 is the number of sets of different non-adjacent vertices in G, that is

$$s_2 = \binom{n}{2} - (n-1) = \frac{(n-1)(n-2)}{2} = 1 \bmod 2$$

if $n = 4k$. □

One might suspect that Theorem 5.2 extends to all graphs, thus strengthening the statement of Theorem 5.1 to 1-fair daemons. However, this is not the case. To see this, let the line graph L_6 be an instance of PDPD. Then *for all* 1*-fair edge-schedulers* **S** *and for all initial configurations I the system self-stabilises starting at I*. We verified this statement using computer simulations by running PDPD for all 6! 1-fair daemons. A result that rendered this experiment computationally feasible is the *state-reduction* technique highlighted in the proof of (ii): to prove self-stabilisation, we only needed to consider those initial configurations with exactly one 1. Thus we had to run $6 \times 6!$ simulations. It is an open problem to find *all* graphs for which this happens. However, Theorem 5.2 shows that the class of such graphs is really limited.

6. Non-adaptive node-schedulers

As we saw in the previous section, even 1-fair edge-schedulers are able to prevent self-stabilisation. What if we only allow the scheduler to choose *one* of the nodes? In this section we study PDPD when adversaries are 1-fair node-schedulers. Because only one of the nodes of the scheduled edge is chosen by the adversary and the other is chosen randomly, the self-stabilisation of the system is a stochastic event.

Theorem 6.1. Let $Star_n$ be an instance of PDPD. Then S_n self-stabilises with probability 1 against *any* 1-fair scheduler.

Proof. We can assume, without loss of generality, that the first node to be scheduled is the centre (labelled 0) and the rest of the nodes are scheduled in the order $1, 2, \ldots, n$. Indeed, if the centre was scheduled later in the permutation of nodes, it is enough to prove self-stabilisation from the configuration that corresponds to first running the system up to just before the centre is scheduled, and then viewing the run as initialised at the new configuration, and with a new periodic schedule (that now starts with node 0). As for the order in which the other nodes get scheduled, by relabelling the nodes, we may assume without loss of generality that this is $1, 2, \ldots, n$.

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

Let $a_0, a_1, \ldots, a_n$ be the labels of nodes at the beginning of the process. It is useful to consider first a deterministic version of the dynamics in question specified as a *game* between two players:

— The first player chooses one node to be scheduled. It is required that the sequence of nodes chosen by this player forms a periodic sequence π. The goal of the first player is to prevent self-stabilisation.
— Given a node choice by the first player, the second player responds with a choice of the second node to be scheduled. Unlike the first player, the sequence of nodes chosen by the second player can potentially vary between successive repetitions of the permutation π. The goal of the second player is to make the system converge to state **0**.

The game above is an example of *scheduler-luck games* in the self-stabilisation literature (Dolev *et al.* 1995). We will provide a strategy for the second player that (when applied) will turn any configuration into the 'all zeros' configuration. But a winning strategy for the second player in the scheduler-luck game will be played with positive probability in any round of the scheduler. Thus with probability going to one (as the number of rounds goes to infinity) this strategy will be played at least in one round, making the system converge to state **0**.

The crux of the strategy is to use the 'partner node' of node 0 carefully, when it is scheduled, to create a segment of nodes $1, 2, \ldots i$ (with i non-decreasing, and eventually reaching n) with labels zero at the beginning of a round of scheduling.

This is simple to do at the very beginning: if node 0 plays node 1 (when 0 is scheduled), then the labels of the two node will be identical, thus when node 1 is scheduled (and plays node 0 again) the label of node 1 will be zero.

If at the beginning of a round the label of node 0 is 1, we make it play (when scheduled at the beginning of a round) the node of smallest positive index $(i + 1)$ that is still labelled 1. This will turn the labels of both nodes to 0. Further scheduling of nodes 1 to $i + 1$ will not change this, and at the end of the round, nodes 1 to $i + 1$ will still be labelled 0.

If, on the other hand, at the beginning of the round node 0 is labelled 0, we make it keep this label (so it does not affect the zero labels of nodes 1 to 1) by making it play (when scheduled) against another node labelled 0 (say node 1).

To complete the argument, we still need to show that for any configuration $x_0, \ldots x_n$, different from the 'all zeros' configuration, we will reach a configuration where the first case applies in a finite number of rounds, and thus the length of the 'all zero' initial segment increases.

Indeed, we assume that $x(0) = 0$ and that it stays that way throughout the process. Then, using $\mathbf{Y_t} = (x(1)_t, \ldots x(n)_t)^T$ to denote the labels of the nodes 1 to n at the beginning of the tth round, it is easy to see that the dynamics of the system is described by the recurrence

$$\mathbf{Y_{t+1}} = B \cdot \mathbf{Y_t},$$

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

where $B = (b_{i,j})$ is an $n \times n$ over $\mathbf{Z}_2$ specified by

$$b_{i,j} = \begin{cases} 1 & \text{if } i \geqslant j \\ 0 & \text{otherwise.} \end{cases}$$

We now consider B as a matrix over $\mathbf{Z}$, rather than $\mathbf{Z}_2$. It is easy to show by induction that $B^k = (b_{i,j}^{(k)})$ where

$$b_{i,j}^{(k)} = \begin{cases} \binom{i-j+k-1}{k-1} & \text{if } i \geqslant j \\ 0 & \text{otherwise.} \end{cases}$$

The kth power over $\mathbf{Z}_2$ is obtained, of course, by reducing these values mod 2. In particular, if we define S_t to be the sum $x(1)_t + \ldots x(n)_t$, it is easy to see that $S_t = x(0)_{t+1}$, and thus, by our hypothesis, S_t has to be zero. On the other hand, a consequence of the previous result is that

$$S_t = \left[\sum_{i=1}^{n} \binom{n-i+t-1}{t-1} \cdot x_i \right] (\text{ mod } 2).$$

In particular,

$$\Delta x_t := S_{t+1} - S_t = \left[\sum_{i=1}^{n-1} \binom{n-i+t-1}{t} \cdot x_i \right] (\text{ mod } 2).$$

By induction and algebraic manipulation, we can generalise this to a higher order of iterated differences $\Delta^k x_t = \Delta(\Delta^{k-1} x_t)$ as

$$\Delta^k x_t = \left[\sum_{i=1}^{n-k} \binom{n-i+t-1}{t+k-1} \cdot x_i \right] (\text{mod } 2).$$

Let i_0 be the smallest index such that $x(i_0) = 1$. Then, by the previous relation,

$$\Delta^{n-i_0} x_t = \left[\sum_{i=1}^{i_0} \binom{n-i+t-1}{t+k-1} x_i \right] = \binom{n-i_0+t-1}{n-i_0+t-1} \cdot x_{i_0} = x_{i_0} = 1 \text{ (mod 2).}$$

But this contradicts the fact that $S_t = 0$ for every value of t and completes the proof. □

A 1-adaptive scheduler keeps repeating the nodes to choose according to a fixed permutation. Thus, for a fixed scheduler and interaction graph, we can talk about the probability of stabilisation in the limit. Also, for a given fixed scheduler, the event that this limit is one is a deterministic statement. Consequently, we can talk of the probability that this event happens when the interaction graph is sampled from a class of random graphs. As noted, for a random scheduler, the condition that G has no isolated vertices is necessary and sufficient to guarantee self-stabilisation with probability 1. This is also true for the adversarial model in the case of non-adaptive (1-fair) daemons. This is similar to the case of an edge daemon, where even non-adaptive daemons could preclude stabilisation.

Theorem 6.2. Let G be an instance of PDPD such that G has no isolated vertices. Then for *any* 1-fair node-scheduler and any initial configuration, the system G reaches state $\mathbf{0}$ with probability 1.

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

The results of Theorem 6.2 should be contrasted with the corresponding result for edge-schedulers, for which, as we showed, even non-adaptive daemons could preclude stabilisation.

The proof consists of the following three components:

(i) our earlier result that guarantees a winning strategy for the scheduler-luck game associated with the dynamics when the underlying graph is S_n;
(ii) the partition of a spanning forest of G into node disjoint stars; and
(iii) the fact that the existence of such a winning strategy is a *monotone graph property* with respect to edge insertions. This is formally stated in the following lemma.

Lemma 6.3. Suppose H is a graph such that a winning strategy W exists for the scheduler-luck game on H. Let $e \notin E(H)$ and $L = H \cup \{e\}$. Then W is also a winning strategy for graph L.

Proof. Given any node choice by the first player, the second player can choose the corresponding node according to strategy W (thus never scheduling the additional edge e). The outcome of the game is, therefore, identical on H and L. □

Proof of Theorem 6.2. In view of Lemma 6.3, it is enough to show the existence of a winning strategy for the second player in the scheduler-luck game on a graph G, when G is a tree. We decompose tree G into a set $\{S_1, \ldots, S_p\}$ of node-disjoint stars as follows:

— Root G at an arbitrary node r.
— Consider the star formed by the root and its children. Call it S_1.
— Remove the nodes in S_1 and all edges with one end point incident on nodes in S_1
— Recursively apply the procedure on each forest created by the above operation.

Now consider a 1-fair schedule π on graph G, which corresponds to a strategy of the first player in the schedule-luck game on G. For every star S_i, the projection π_i of the schedule on the nodes of S_i (which amounts to only considering scheduled nodes that belong to S_i) specifies a 1-fair schedule on S_i. According to Theorem 6.1, the second player has a winning strategy W_i for the scheduler-luck game on S_i when the first player acts according to the schedule π_i.

Next, we devise a strategy W for the scheduler-luck game on graph G, by 'composing' the winning strategies W_i. Specifically, if the node chosen by the first player belongs to star S_i, strategy W will employ W_i to choose the corresponding second node. Since on each star S_i the labels of the node will eventually be $\mathbf{0}$, W is a winning strategy for the second player in the scheduler-luck game on G. □

7. Adaptive node-schedulers

In the previous section we showed that non-adaptive schedulers cannot preclude self-stabilisation. In contrast, as the following theorem shows, 3-fair non-adaptive node-schedulers are still powerful enough to preclude self-stabilisation *with complete certainty*, and so are 2-*fair adaptive*[†] schedulers.

[†] Obviously, there are no 1-fair adaptive schedulers.

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

State	scheduled node	possible partner	resulting state
111	1	2	001
111	1	3	010
001	3	1	101
001	3	2	011
010	2	3	011
010	2	1	110
101	2	1,3	111
011	1	2,3	111
110	3	1,2	111

Fig. 1. A round of the 2-fair adaptive scheduler

Theorem 7.1. The following statements are true:

(i) Let the star graph $Star_n$ $(K_{1,n})$ be an instance of PDPD. Then there exists an initial configuration I and a 3-fair non-adaptive scheduler that precludes self-stabilisation on $Star_n$ starting in I.

(ii) Let the triangle K_3 be an instance of PDPD. Then there exists an initial configuration I and a 2-fair *adaptive* scheduler that precludes self-stabilisation on K_3 starting in I.

Proof.

(i) Consider the star graph $Star_n$ $(K_{1,n})$, with its centre labelled 0 and the rest of the nodes labelled $1, 2, \ldots, n$. We have to provide an example of a 3-fair scheduler that precludes self-stabilisation on some initial configuration. This initial configuration has two 1's, at nodes 1 and 2. The scheduler repeats the schedule $[0, 1, 1, 3, 4, \ldots n-2, 2, 1, n-1, n-1]$. After the scheduling of the initial subsequence $[0, 1, 1]$, the effect is that both nodes have label 0. Thus the scheduling of nodes $3, 4, \ldots, n-2$ does not change any label. With node 2, the label of node 0 will change to 1, thus changing in the next step the label of node 1 back to 1. Finally, scheduling the node $n-1$ twice turns back the label of node 0 to 0, thus yielding the initial configuration. It is easy to see that the scheduler is 3-fair.

(ii) Start with configuration I consisting of all ones. The scheduler will adaptively schedule the nodes in sequences of three so that at the end of such a *3-block*, the system is guaranteed to be in configuration I again. Figure 1 describes the strategy of the scheduler, assuming that node 1 is scheduled first.

The table illustrates a round (denoted S_1) of three scheduled nodes, with the first scheduled node being 1. We can define schedules S_2 and S_3 similarly, starting with 2 and 3, respectively. The columns in the table first list the global state, then the scheduled node in that state. The third column corresponds to the possible probabilistic choices of the partner node, with multiple (inconsequential) choices separated by a comma. A schedule of S_1 starts in state 111, first scheduling node 1. Depending on the outcome, he will get into state 001 or 010. Possible moves out of these states bring the system into state 101, 011 or 110, and one more move brings the system back to state 111

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

again. Note that a 3-block of S_1 consists of either a permutation of nodes (schedules 132, 123), or two nodes, with the initial and the final node in the block being identical (schedules 121, 131). Similar descriptions hold for 3-blocks of S_2, S_3.

The scheduler now proceeds to create an infinite schedule consisting of 3-blocks according to the following rule:

(a) If a given block B of S_a, the last schedule employed, is a permutation, then the start of the next block will also be a block of S_a.

(b) Otherwise, if block B is missing node z, the next block will come from S_z.

It is easy to see that the scheduler we have constructed is 2-adaptive and precludes self-stabilisation. □

Although a formal definition of the probability of self-stabilisation is more complicated in this case, we can still talk of the probability of self-stabilisation for adaptive daemons. However, as we have seen, the result of Theorem 6.1 is *no longer true*: on stars, 1-fairness is stronger than 2-fair adaptive scheduling. It would seem that this result shows that non-adaptiveness is important for self-stabilisation. However, we will see that the class of network topologies where this happens is reasonably limited. Indeed, we will next study self-stabilisation on Erdős–Renyi random graphs $G(n, p)$. We will choose p in such a way that with high probability a random sample from $G(n, p)$ has no isolated vertices[†]. In other words, we require that the necessary condition on the topology of G holds with probability $1 - o(1)$. We will say a graph G is *good* if for *any scheduler of bounded fairness* and any starting configuration I, we have G starting at I converges to $\mathbf{0}$ with probability $1 - o(1)$ as the number of steps goes to infinity.

Theorem 7.2. Let p be such that $np - \log n \to \infty$. Then with probability $1 - o(1)$, a random graph $G \in G(n, p)$ is good.

Of course, a natural question is whether such a weakening of the original result, from any graph topology satisfying a given condition to a generic random graph satisfying the same condition, is reasonable. We are, however, not the first to propose such an approach. Indeed, apart from a handful of cases, the network that a given social dynamics takes place on is not known in its entirety. Instead, a lot of recent work (see, for example, Pastor-Santorras and Vespigniani (2007), Newman *et al.* (2006) and Durrett (2006)) has resorted to the study of *generic* properties of random network models that share some of the observable properties of a *fixed* network (such as the Internet or the World Wide Web).

Proof. The plan of the proof is similar to that for 1-fair schedulers. We define a *round* of a b-fair scheduler to consist of a consecutive sequence of $b(n-1)+1$ steps.

(i) We prove that for graphs from a class **B** of 'base case' graphs (see Lemma 7.3 below), the second player has a winning strategy in the scheduler-luck game associated with

[†] A random sample from $G(n, p)$ has no isolated vertices with probability $1 - o(1)$ when $np - \log n \to \infty$ (Janson *et al.* 2000).

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

any scheduler of bounded fairness, where the game corresponds to a finite number of rounds of the scheduler.

(ii) We use the monotonicity of the existence of a strategy.

(iii) We show that with probability $1 - o(1)$, the vertices of a random sample graph G from the graph process can be partitioned such that all the induced subgraphs are isomorphic to one graph in G.

Lemma 7.3. The following are true:

(i) Let G be a graph with a perfect matching. Then for any scheduler S of bounded fairness and any initial configuration on G, the second player has a winning strategy for the scheduler-luck game corresponding to one round of the scheduler.

(ii) Let the line L_n, $n \geqslant 6$ be an instance of PDPD. Then for any node-scheduler $\mathbf{S}$ of bounded fairness and any initial configuration, the second player has a winning strategy in the scheduler-luck game associated with two consecutive rounds of $\mathbf{S}$.

Proof.

(i) A perfect matching M of G specifies a winning strategy in a scheduler-luck game: each node plays (when scheduled) against its partner in M. Since every node is scheduled at least once in a round of scheduling, every edge of M is played at least twice. Therefore, irrespective of the initial configuration, the final configuration is $\mathbf{0}$.

(ii) The lemma only needs to be proved in the case when n is odd (in the other case, the strategy based on perfect matchings applies). In this case the winning strategy is specified as follows:

  (a) In the first round, turn the leftmost L_4 portion of L_n into the all zero state by playing the matching based winning strategy.

  (b) In the second round, nodes in the leftmost L_3 will only choose to play against each other when scheduled, and thus stay at 0. The remaining nodes form a graph isomorphic to L_{n-3}, and in this round we use the perfect matching based strategy for this graph. □

Note that the statement of Lemma 6.3 extends to scheduler-luck games associated with node daemons of *bounded* fairness. The proof is similar (a strategy for the game on G is also a strategy for the game on a graph with a larger set of edges). Theorem 7.2 immediately follows if n is even: a classical result in random graph theory (see, for example, Janson *et al.* (2000, pages 82–85)) asserts that with probability $1 - o(1)$, G will have a perfect matching.

To complete the proof of Theorem 7.2, we only need to deal with the case when n is odd. For a graph G and a set of vertices V denote by $G|_V$ the subgraph induced by vertex set V.

Lemma 7.4. With probability $1 - o(1)$, G can be partitioned into $V = V_1 \cup V_2$ such that:

(i) $G|_{V_1}$ contains L_7 as an edge-induced subgraph.

(ii) $G|_{V_2}$ is a graph with a perfect matching.

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

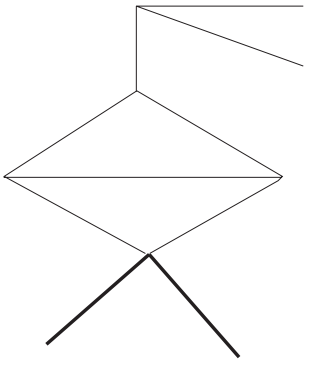

Fig. 2. A *cherry* in a graph (with bold lines)

Theorem 7.2 follows from Lemma 7.4, since for both $G|_{V_1}$ and $G|_{V_2}$ the second player has a winning strategy in the scheduler-luck game. A winning strategy for the corresponding game on G proceeds by using the winning strategy for $G|_{V_1}$, when the scheduled node is in V_1, and the winning strategy for $G|_{V_2}$ when the scheduled node is in V_2.

Proof of Lemma 7.4. The proof goes along lines similar to the proof of the existence of a perfect matching in a random graph (see Janson *et al.* (2000, pages 82–85)). The first step is to show that with high probability G does not contain a set of distinct vertices x_0, x_1, x_2, x_3, x_4 such that:

— $deg_G(x_0) = deg_G(x_4) = 1$.

— For every $i = 0, \ldots, 3$, we have x_i and x_{i+1} are adjacent.

This is easy to see, since the expected number of such structures is

$$O\left(n^5 p^4 (1-p)^{2(n-1)}\right) = O\left(n^5 p^4 e^{-2np}\right) = o(1),$$

since

$$p = \Theta\left(\frac{\log(n)}{n}\right).$$

We now consider a random graph G, conditioned on not containing such a structure, and a vertex x_0 in G of degree one. Since this information only exposes information on the edges with one endpoint at x_0, with probability $1 - o(1)$, graph $H = G \setminus \{x_0\}$ has a perfect matching. Let x_1 be the node in $G \setminus \{x_0\}$ adjacent to x_0 and let x_2 be the node matched to x_1 in H. With probability $1 - o(1)$, we get that x_2 has another neighbour x_3 in H, since otherwise G would contain a *cherry*, that is, two vertices of degree one at distance exactly 2 in G (see Figure 2), and a random sample from $G(n, p)$ only contains a cherry with probability $o(1)$ (see Janson *et al.* (2000, page 86)). Let x_4 be the node x_3 is matched to in H. Again, with probability $1 - o(1)$, we get that x_4 has a neighbour x_5 in H different from x_3 (otherwise the five vertices x_0 to x_4 would form a structure we have conditioned on not occurring in G). Finally, let x_6 be the node matched to x_5 in H. Then the restriction of G to the set $V_1 = \{x_0, \ldots, x_6\}$ contains a copy of L_7, and G restricted to $V_2 = V \setminus V_1$ contains a perfect matching (induced by the perfect matching on H). □

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

$\pi \mid n$	4	8	16	32	64	128
id	2.486	4.225	6.401	8.33	10.498	13.135
p3	2.469	4.039	5.807	7.662	9.639	11.718
rd	2.289	4.499	6.527	8.781	11.161	14.151
(13)	2.168	4.656	7.069	9.837	12.653	14.859
$\pi \mid n$	256	512	1024			
id	16.091	17.954	20.331			
p3	14.323	16.054	19.826			
rd	17.342	20.518	22.336			
(13)	18.504	20.346	20.392			

Fig. 3. Rounds on C_n under 1-fair scheduling.

This completes the proof of Theorem 7.2. □

8. Speed of convergence

The previous theorems have shown that results on the convergence to a fixed point can be studied in (and extended to) an adversarial framework. Perhaps what is not preserved as well in the adversarial framework are results on the *computational efficiency* of convergence to equilibrium. Such results include, for instance, the above mentioned $O(n \log n)$ bound of Dyer *et al.* (2002). The proof of this theorem displays an interesting variation on the idea of a potential function. It uses such a function, but in this case the value of the function only diminishes 'on the average', rather than for *every* possible move. Therefore, bounding the convergence time seems to critically use the 'global' randomness introduced in the dynamics by random matching, and does *not* trivially extend to adversarial versions. On the other hand, the proof of Theorem 5.2 only guarantees an exponential upper bound on the expected convergence time.

We have investigated experimentally the convergence time on C_n for some classes of 1-fair schedulers (permutations). Some of our results are presented in Figure 3, where we present the average number of *rounds*, rather than steps, over 1000 samples at each point. The symbol **id** denotes the identity permutation $(12 \dots n)$, **p3** is the permutation $\sigma[i] = 3i \pmod n$, **(13)** refers to permutations with the pattern $(13245768 \dots)$, and **rd** refers to the maximum average number of rounds, taken over 10 *random* permutations. In all cases, the convergence time is consistent with the above-mentioned $O(n \log n)$ result.

We have been unable to prove such a result formally (and leave it as an interesting open problem)†. It is even more interesting to study the dependency of the mixing time

† A promising approach is outlined in Fribourg and Messika (2005).

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

of the dynamics (Dyer *et al.* 2002) on the underlying network topology. While there are superficial reasons for optimism (for some models in evolutionary game theory, see, for example, Morris (2000), the impact of network topology on the convergence speed of a given dynamics is reasonably well understood) – see Mossel and Roch (2006) (especially the concluding remarks) for a discussion on the difficulties of connecting network topology and convergence speed for the specific dynamics we study.

9. Discussion of results and conclusions

We have advocated the study of evolutionary game-theoretic models under adversarial scheduling similar to the ones in the theory of self-stabilisation. As an illustration, we have studied the Iterated Prisoner's Dilemma with the win–stay lose–shift strategy.

Our results are an illustration of the adversarial approach as follows:

(i) **Start with some result P that is valid under random scheduling.**
The original statement is presented in Section 2.2.

(ii) **Identify several structural properties of a random scheduler that impact the validity of P.**
The random scheduler is:
— *fair*, even $O(n \log n)$ fair with high probability by the Coupon Collector Lemma;
— *endogenous*, since the next scheduled edge is not fixed in advance;
— *non-adaptive*, since the next edge to be scheduled does not depend on the configuration of the system.

(iii) **Identify those properties (or combinations of properties) that are *necessary/sufficient* for the validity of P.**
Theorem 4.3 shows that fairness is a *necessary condition* for the extension of the original result to adversarial settings. Next, the definition of node- and edge-schedulers illustrates another important property of random schedulers, namely, the *endogeneity of agent interactions*: an edge-scheduler completely specifies the dynamics of interaction. In contrast, node-schedulers provide perhaps the weakest possible form of endogeneity: the underlying social network is still fixed, but agents can choose one of their neighbours to interact with (or simply play a random one).
Theorems 6.1 and 6.2 show that, in contrast with the case of edge-schedulers, even this limited amount of endogeneity is sufficient to recover the original result for the random scheduler. Moreover, the proofs illuminate the role of endogeneity, which was somewhat obscured in the (trivial) original proof that the Pavlov dynamics (under random matching) converges with high probability to the 'all zeros' fixed point. This proof relies implicitly on the fact that from every state there exists a sequence of 'correct' moves that 'funnels' the system towards the fixed point. For *node*-schedulers, the existence of a such a set of correct moves is proved by explicit construction, and is more difficult in the adversarial setting. The existence of such a set of moves is precisely what the exogenous choice of agents is able to preclude.

https://doi.org/10.1017/S0960129511000533 Published online by Cambridge University Press

(iv) **Correspondingly, identify those properties that are inessential to the validity of P. In the process one can reformulate the original statement in a way that makes it more robust.**

Theorem 7.1 shows that if we allow schedulers to be adaptive, then *network topology* becomes important, and can invalidate the original result in an adversarial setting. However, adaptiveness (or, equivalently, the amount of fairness) is *inessential* if we require the convergence result to only hold *generically* with respect to the class of network topologies described by Erdős–Renyi random graphs.

The results we have proved also highlight a number of techniques from the theory of self-stabilisation that might be useful in developing a general theory: the concept of scheduler-luck games, composition of strategies by partitioning the interaction topology, monotonicity and 'generic preservation' using threshold properties. Obviously, a reconsideration of more central game-theoretic models under adversarial scheduling is required (and would be quite interesting).

We conclude this section with one (simple) observation concerning the robustness of the convergence time under adversarial models. The $\Theta(n\log(n))$ mixing time is a particular case of the 'colearning' model in Shoham and Tennenholtz (1997), for which the $\Omega(n\log(n))$ simple lower bound on the convergence time is matched. This lower bound is based simply on the Coupon Collector problem. For a b-fair scheduler, this immediately translates into a $\Omega(b)$ lower bound for the adversarial model.

Acknowledgments

We are grateful to an anonymous referee for extremely insightful comments.

References

Aspnes, J. and Ruppert, E. (2007) An introduction to population protocols. *Bulletin of the European Association for Theoretical Computer Science* **93** 98–117.

Axelrod, R. (1984) *The Evolution of Cooperation*, Basic Books.

Axelrod, R. (1997) *The Complexity of Cooperation. Agent-Based Models of Competition and Cooperation*, Princeton Studies in Complexity, Princeton University Press.

Axtell, R. and Epstein, J. (1996) *Growing Artificial Societies: Social Science from the Bottom Up*, The MIT Press.

Ballot, J. and Weisbuch, G. (2000) Introduction: Why simulation in the social sciences. *Advances in Complex Systems* **3** (1-4) 9–16.

Barrett, C., Eubank, S. and Marathe, M. (2005) Modeling and simulation of large biological, information and socio-technical systems: An interaction based approach. In: Goldin, D., Smolka, S. and Wegner, P. (eds.) *Interactive Computation: The New Paradigm*, Springer Verlag 353–394.

Barrett, C., Hunt, H., Marathe, M. V., Ravi, S. S., Rosenkrantz, D. and Stearns, R. (2003) Reachability problems for sequential dynamical systems with threshold functions. *Theoretical Computer Science* **295** (1-3) 41–64.

Boudon, R. (1998) Social mechanisms without black boxes. In: Hedström, P. and Swedberg, R. (eds.) *Social mechanisms: An analytical approach to social theory*, Cambridge University Press.

Bournez, O., Chalopin, J., Cohen, J. and Koegler, X. (2009) Population protocols that correspond to symmetric games. (Preprint available at arXiv:0907.3126v1 [cs.GT].)

Bunge, M. (1997) Mechanism and explanation. *Philosophy of the Social Sciences* **27** (4) 410.

Castellano, C., Fortunato, S. and Loreto, V. (2009) Statistical physics of social dynamics. *Reviews of modern physics* **81** (2) 591–646.

Coleman, A., Colman, A. and Thomas, R. M. (1991) Cooperation without awareness: A multiperson generalization of the minimal social situation. *Behavioral Science* **35** 115–121.

Colman, A. (2005) Cooperation in multi-player minimal social situations: An experimental investigation. British Academy Larger Research Grants Scheme Grant No. LRG-37265, 2004–2005.

Craver, C. F. (2006) When mechanistic models explain. *Synthese* **153** (3) 355–376.

Dolev, S., Israeli, A. and Moran, S. (1995) Analyzing expected time by scheduler-luck games. *I.E.E.E. Transactions on Software Engineering* **21** (5) 429–439.

Dolev, S. (2000) *Self-stabilization*, M.I.T. Press.

Durrett, R. (2006) *Random Graph Dynamics*, Cambridge University Press.

Dyer, M., Greenhill, C., Goldberg, L., Istrate, G. and Jerrum, M. (2002) The Convergence of Iterated Prisoner's Dilemma Game. *Combinatorics, Probability and Computing* **11** 135–147.

Dyer, M. and Velumailum, M. (2011) The Iterated Prisoner's Dilemma on a cycle. (Available at arXiv:1102.3822v1 [cs.GT].)

Elster, J. (1998) *A plea for mechanisms*. In: Hedström, P. and Swedberg, R. (eds.) *Social Mechanisms: An Analytical Approach to Social Theory*, Cambridge University Press.

Epstein, J. M., Cummings, D., Chakravarty, S., Singa, R. and Burke, D. (2004) *Toward a Containment Strategy for Smallpox Bioterror. An Individual-Based Computational Approach*, Brookings Institution Press.

Epstein, J. (1999) Agent-based computational models and generative social science. *Complexity* **4** (5) 41–60.

Epstein, J. (2007) *Generative Social Science: Studies in Agent-based Computational Modeling*, Princeton University Press.

Eubank, S., Guclu, H., Kumar, V. S. A., Marathe, M. V., Srinivasan, A., Toroczkai, Z. and Wang, N. (2004) Monitoring and mitigating smallpox epidemics: Strategies drawn from a census data instantiated virtual city. *Nature* **429** (6988) 180–184.

Fribourg, L. and Messika, S. (2005) Brief Announcement: Coupling for Markov Decision Processes – Application to Self-Stabilization with Arbitrary Schedulers. In: *Proceedings of the Twenty-Fourth Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC'05)* 322.

Fribourg, L., Messika, S. and Picaronny, C. (2006) Coupling and self-stabilization. *Distributed Computing* **18** (3) 221–232.

Gilbert, N. and Troizch, K. (2005) *Simulation for social scientists* (second edition), Open University Press.

Glennan, S. S. (1996) Mechanisms and the nature of causation. *Erkenntnis* **44** (1) 49–71.

Hedström, P. (2005) *Dissecting the social: on the principles of analytical sociology*, Cambridge University Press.

Hedström, P. and Bearman, P. (2009) *The Oxford handbook of analytical sociology*, Oxford University Press.

Hedström, P. and Swedberg, R. (eds.) (2006) *Social Mechanisms: An Analytical Approach to Social Theory*, Cambridge University Press.

Hegselmann, R. and Flache, A. (1998) Understanding complex social dynamics: A plea for cellular automata based modelling. *Journal of Artificial Societies and Social Simulation* **1** (3) 1.

Jaggard, A., Schapira, M. and Wright, R. (2000) Distributed Computing with Adaptive Heuristics. *Proceedings of the Innovations in Computer Science Conference (ICS 2011)*, Tsinghua University Press 417–443.

Janson, S., Luczak, T. and Ruczinski, A. (2000) *Random Graphs*, Wiley.

Kittock, J. (1994) Emergent conventions and the structure of multi-agent systems. In: Nadel, L. and Stein, D. (eds.) *1993 Lecture Notes in Complex Systems: the proceedings of the 1993 Complex Systems Summer School* Santa Fe Institute Studies in the Sciences of Complexity, Volume VI, Addison Wesley Publishing Co.

Krapivsky, P. L., Redner, S. and Ben-Naim, E. (2010) *A kinetic view of statistical physics*, Cambridge University Press.

Lazer, D. (2009) Life in the network: the coming age of computational social science. *Science* **323** (5915) 721.

Machamer, P., Darden, L. and Craver, C. F. (2000) Thinking about mechanisms. *Philosophy of Science* **67** (1) 1–25.

Mirowski, P. (2002) *Machine dreams: Economics becomes a cyborg science*, Cambridge University Press.

Morris, S. (2000) Contagion. *The Review of Economic Studies* **67** (1) 57–78.

Mossel, E. and Roch, S. (2006) Slow emergence of cooperation for win–stay lose–shift on trees. *Machine Learning* **7** (1-2) 7–22.

Mortveit, H. and Reidys, C. (2007) *An Introduction to Sequential Dynamical Systems*, Springer Verlag.

Newman, M., Barabási, A. L. and Watts D. (eds.) *The Structure and Dynamics of Networks*, Princeton University Press.

Nowak, M. and Sigmund, K. (1993) A strategy of win–stay, lose–shift that outperforms tit-for-tat in the prisoner's dilemma game. *Nature* **364** 56–68.

Pastor-Santorras, R. and Vespigniani, A. (2007) *Evolution and Structure of the Internet: A Statistical Physics approach*, Cambridge University Press.

Posch, M. (1997) Win stay–lose shift: An elementary learning rule for normal form games. Technical Report 97-06-056, the Santa Fe Institute.

Radloff, R., Kelley, H., Thibaut, J. and Mundy, D. (1962) The development of cooperation in the minimal social situation. *Psychological Monographs* **76**.

Shoham, Y. and Tennenholtz, M. (1997) On the emergence of social conventions: modelling, analysis and simulations. *Artificial Intelligence* **94** (1-2) 139–166.

Sidowski, J. (1957) Reward and punishment in the minimal social situation. *Journal of Experimental Psychology* **54** 318–326.

Tesfatsion, L. and Judd, K. L. (eds.) (2006) *Handbook of Computational Economics. Volume 2: Agent-based computational economics*, North Holland.

Thibaut, J. and Kelley, H. (1959) *The social Psychology of Groups*, Wiley.

Velupillai, K. (2000) *Computable economics: the Arne Ryde memorial lectures*, Oxford University Press.

Vriend, N. (2006) ACE models of endogenous interaction. In: Tesfatsion, L. and Judd, K. L. (eds.) *Handbook of Computational Economics. Volume 2: Agent-based computational economics*, North Holland.

Wilhite, A. (2006) Economic activity on fixed networks. In: Tesfatsion, L. and Judd, K. L. (eds.) *Handbook of Computational Economics. Volume 2: Agent-based computational economics*, North Holland.

Young, H. P. (1998) *Individual Strategy and Social Structure: an Evolutionary Theory of Institutions*, Princeton University Press.