Swarm Intell (2011) 5:225–255
DOI 10.1007/s11721-011-0058-8

Recovery properties of distributed cluster head election using reaction–diffusion

Lidia Yamamoto · Daniele Miorandi · Pierre Collet · Wolfgang Banzhaf

Received: 22 October 2010 / Accepted: 12 August 2011 / Published online: 30 September 2011

Abstract Chemical reaction–diffusion is a basic component of morphogenesis, and can be used to obtain interesting and unconventional self-organizing algorithms for swarms of autonomous agents, using natural or artificial chemistries. However, the performance of these algorithms in the face of disruptions has not been sufficiently studied. In this paper we evaluate the use of reaction–diffusion for the morphogenetic engineering of distributed coordination algorithms, in particular, cluster head election in a distributed computer system. We consider variants of reaction–diffusion systems around the activator–inhibitor model, able to produce spot patterns. We evaluate the robustness of these models against the deletion of activator peaks that signal the location of cluster heads, and the destruction of large patches of chemicals. Three models are selected for evaluation: the Gierer–Meinhardt Activator–Inhibitor model, the Activator–Substrate Depleted model, and the Gray–Scott model. Our results reveal a trade-off between these models. The Gierer–Meinhardt model is more stable, with rare failures, but is slower to recover from disruptions. The Gray–Scott model is able to recover more quickly, at the expense of more frequent failures. The Activator–Substrate model lies somewhere in the middle.

Keywords Reaction–diffusion · Activator–inhibitor · Pattern formation · Cluster head

L. Yamamoto (✉) · P. Collet
FDBT-LSIIT, University of Strasbourg, Pôle API, Bd. Sébastien Brant, 67412 Illkirch, France
e-mail: lidia.yamamoto@unistra.fr

P. Collet
e-mail: pierre.collet@unistra.fr

D. Miorandi
iNSPIRE Team, CREATE-NET, v. alla Cascata 56/D, Povo, 38123 Trento, Italy
e-mail: daniele.miorandi@create-net.org

W. Banzhaf
Department of Computer Science, Memorial University of Newfoundland, St. John's, NL A1B 3X5, Canada
e-mail: banzhaf@cs.mun.ca

1 Introduction

Morphogenetic engineering (Doursat 2008), or *embryomorphic engineering* explores new design and programming techniques based on the way biological development is able to produce complex organisms without central control. Morphogenetic engineering requires a collection of autonomous agents that self-organize into a proper architecture that achieves the desired result. Therefore it can be seen as a promising technique for swarm intelligence.

Morphogenetic engineering gets inspiration from natural *morphogenesis*, the process by which the embryo develops from a single-cell egg into an adult form with different organs and tissues. Morphogenesis relies on chemical signals that trigger the activation or suppression of certain genes, causing cells to differentiate into the proper cell types. Such chemical signals may travel through cells by passive diffusion or by active transport. While traveling, they might encounter other chemicals and react with them, forming new substances with potentially different roles in the cell signaling process. Hence morphogenesis is necessarily subject to an underlying *reaction–diffusion* process, that is, a process by which chemicals react and diffuse in space. Indeed, reaction–diffusion was proposed by Turing (1952) as the first mathematical explanation for some common morphogenetic patterns found on the skin of animals, such as leopard spots and zebra stripes.

Although far from being fully responsible for all the complex morphogenetic processes encountered in nature, reaction–diffusion is one of the simplest and most basic components of morphogenesis. Therefore understanding and harnessing the programmability of reaction–diffusion is a basic step toward achieving the vision of morphogenetically steering complex swarms of autonomous agents.

In this paper we evaluate the use of reaction–diffusion as a cell signaling system for the morphogenetic engineering of one particular class of distributed coordination algorithms: *distributed cluster head election*. The distributed cluster head election problem (Yu and Chong 2005; Erciyes et al. 2007; Soro and Heinzelman 2009) consists in selecting (or "electing") a number of nodes (processors or agents) out of a network of interconnected nodes, that act as leaders for a group of agents in the region around them. These region leaders are called *cluster heads*. As the leader of a local group of processors, the cluster head node may provide a variety of services to its neighborhood, such as aggregating information from nearby nodes in a sensor network, or saving energy by switching off redundant nodes. For reasons of efficiency and scalability, it is interesting to have an entirely decentralized election process, relying solely on local communication among the agents. At the end of the process, all agents must have one leader assigned to them, and in the case of changes in network topology, the system must reconfigure itself accordingly. The problem of cluster head election can be regarded as equivalent to obtaining a spatial pattern characterized by regularly distributed spots, where each spot corresponds to a cluster head. Reaction–diffusion processes are known to be able to produce similar patterns (such as leopard spots). These patterns have been used in the implementation of instances of decentralized cluster head election algorithms, as discussed below.

Various mathematical models have been proposed to model reaction–diffusion processes and explain pattern formation. A basic one is the *activator–inhibitor* model, where two chemicals, an activator and an inhibitor, interact in an antagonistic way, resulting in Turing patterns in space (Turing 1952), such as spots and stripes on the skin of animals. Activator–inhibitor models are recurrent components in the study of morphogenesis. They offer an abstract model to explain many different morphogenetic phenomena, including the regular spacing of cactus thorns and bird feathers, regeneration of the shape after damage, the production of sequences of repeated elements such as

insect body segments, the assembly of photoreceptor cells in insect eyes, or the positioning of leaves in growing plants (Bar-Yam 2003; Meinhardt 1982; Murray 2003; Deutsch and Dormann 2005; Koch and Meinhardt 1994). They have also been used as inspiration for designing algorithms able to produce textures and landscapes in computer graphics, and for autonomous, decentralized, distributed coordination algorithms, for instance in swarm robotics (Shen et al. 2004), amorphous computing (Abelson et al. 2000; Rauch 2003), wireless sensor networks (Durvy and Thiran 2005; Neglia and Reina 2007; Lowe et al. 2009), and autonomous surveillance systems (Yoshida et al. 2005; Hyodo et al. 2007). Durvy and Thiran (2005) and Neglia and Reina (2007) have proposed the use of spot patterns to solve instances of cluster head election problems in sensor networks. Shen et al. (2004) have used these patterns to guide shape formation for collective task solving in swarm robotics. In this paper, we complement this research with an experimental evaluation, based on numerical simulations, of the robustness properties that a possible chemical implementation of cluster head election schemes using activator–inhibitor models could present.

When spot patterns are produced by an activator–inhibitor process, the spot locations typically indicate peaks in activator concentration. From an algorithmic perspective, one can imagine a two- or three-dimensional space filled with processors or cells that are sensitive to chemical concentrations and their gradients. Chemicals can then be used as signals to assign tasks to processors, to trigger certain responses, to change their behavior, and so on. In our case, the location of an activator peak is interpreted as a candidate processor for executing a cluster head function. Using the reaction–diffusion pattern formation process in this way provides a decentralized way to elect cluster heads via chemical signaling. The resulting system can be regarded as a distributed algorithm that solves an instance of a cluster head election problem. Note that the implementation of the chemical signaling process might be handled by the processors themselves, as in amorphous computing, or might be performed by real chemicals present in the environment. The former simulates reaction–diffusion and as such requires sufficient computation power in the processors and the implementation of an amorphous diffusion process as described by Rauch (2003), while the latter delegates such calculations to the environment in an embodied computation fashion (Pfeifer et al. 2005), where the processors make use of the information encoded in chemicals' concentration level to take decisions. We believe that the results presented in this paper fit the embodied case more closely, while also covering the amorphous case to a certain extent.

For the purpose of evaluating the robustness of distributed cluster head election, we select three well-known reaction–diffusion models in the activator–inhibitor category, in a range of parameters for which they produce spot patterns. The first model studied is the Gierer–Meinhardt Activator–Inhibitor model, in which an activator chemical autocatalyses its own production, and an inhibitor chemical inhibits the activity of the catalyst. The second one is the Activator-Depleted Substrate model, in which the activator needs to feed on a substrate that becomes depleted during the autocatalysis process. The third model is the Gray–Scott model, a special case of the second model that has been shown to produce a rich variety of interesting patterns, including spots that grow and divide. The first two models are extensively described by Meinhardt (1982) and Koch and Meinhardt (1994), while the third one can be found in Gray and Scott (1990), Pearson (1993), Mazin et al. (1996). Due to the special features of the Gray–Scott model, we have decided that it deserved a separate evaluation, although it is a special case of the second model.

It is important to design a distributed cluster head election algorithm that is robust to disruptions of several kinds, including the deletion of chemicals. Meinhardt (1982) and others have pointed out that activator–inhibitor models are robust to perturbations in the resulting patterns. For instance, removing an activator peak results in this peak being regenerated

or another peak being formed at a nearby place. However, little has been reported about the actual quantitative performance of implementations of activator–inhibitor models under perturbations. We seek to cover this gap by experimentally evaluating the robustness of the three models in the presence of attacks that remove existing activator peaks that signal the location of cluster heads, and attacks that destroy entire regions containing chemicals. The main contribution of this paper is a comprehensive quantitative evaluation of the robustness of artificial chemistry simulations of the three above cited models under such perturbations.

In the interest of a potential embodied implementation, we approach the use of reaction–diffusion for computation from a chemical computing (Dittrich 2005) perspective: chemical reactions can be regarded as computation steps, and the overall computation process can be viewed as a movement in the state space of a dynamical system (Stepney 2010), which in our case is represented by the system of chemical reactions combined with the set of processors that use chemical signaling as source of information. For this purpose, we focus on the chemical implementation of the reaction–diffusion systems[1] covered here, that is, we explicitly derive the set of chemical reactions that lead to the desired patterns, as opposed to the more widespread approach of using the corresponding differential equations directly. Our goal is to ensure compatibility with a potential chemical implementation, either in an artificial chemistry (Dittrich et al. 2001), or in natural chemical computing such as reaction–diffusion processors (Adamatzky et al. 2005). The advantage of an artificial chemistry implementation is an easier algorithmic modeling and proof of properties by applying analysis techniques from dynamical systems, for instance, as suggested by Meyer and Tschudin (2009). On the other hand, it can be computationally expensive to simulate a chemistry on traditional hardware. In contrast, by implementing such algorithms in chemical media ("wetware") these high simulation costs can be avoided. However, natural chemical computing is still in its infancy, and the implementation of our algorithms using real chemistry is beyond the scope of this paper.

Our results reveal a trade-off between the models studied. The chemical derivation of the Gierer–Meinhardt model, although more stable, seems not as robust as expected, since activator peaks are slower to recover from disruptions. The Gray–Scott model is able to recover more quickly, at the expense of more frequent failures. The Activator–Substrate model lies somewhere in the middle.

This paper is an extension of a previous work (Yamamoto and Miorandi 2010). In the present paper we have added the Gray–Scott model for comparison, showing that it has very good self-healing properties. Moreover, we also compare the three chosen reaction–diffusion models against the non-chemical algorithm by Basagni (1999), which we judged as representative of the state of the art in cluster head election. In addition we have evaluated the robustness of the three reaction–diffusion models on large grids and in an amorphous computing scenario, both with the help of a GPU (Graphics Processing Unit). We also discuss the potential application domains more extensively.

The remainder of this paper is organized as follows. In Sect. 2 we introduce the basics of reaction–diffusion systems. In Sect. 3 we detail the mathematical model of the two activator–inhibitor systems we will be dealing with. Section 4 presents the experimental setup we used for validation purposes. The outcomes of our experiments are presented and discussed in Sects. 5, 6, and 7: Sect. 5 looks at the dynamics of the process of recovery from

[1] It is common to model a reaction–diffusion process by a system of chemical reactions, and/or by a system of differential equations. For this reason, with some abuse of language, we will use the terms "reaction–diffusion process/model/system" interchangeably; a similar argument applies to the various activator–inhibitor processes/models/systems considered in this paper.

disruptions. Section 6 includes fault-tolerance measurements on large grids performed on a GPU. Section 7 shows the recovery behavior of the three models in an amorphous computing scenario. Section 8 describes a number of potential applications of activator–inhibitor models and discusses some related approaches. Section 9 concludes the paper by pointing out directions for extending the presented work.

2 Chemical kinetics and reaction–diffusion processes

The *Law of Mass Action* states that, in a well-stirred reactor, the average speed (or rate) of a chemical reaction is proportional to the product of the concentrations of its reactants (Atkins and de Paula 2002).

The concentration dynamics of all molecules in the system can be described by a system of ordinary differential equations (ODE), where each equation describes the change in concentration of one particular molecular species. This ODE system can be expressed in matrix notation as follows:

$$\frac{d\mathbf{c}(t)}{dt} = M\mathbf{v}(t) \tag{1}$$

where $d\mathbf{c}(t)/dt$ is the vector of differential equations expressing how the concentration c_i of each of the species C_i changes in time t; M is the stoichiometric matrix of the system, which expresses the net changes in number of molecules for each species in each reaction; and $\mathbf{v}(t)$ is a vector of rates for each reaction. The rates may follow the law of mass action, or other laws such as enzyme or Hill kinetics.

In a reaction–diffusion process, substances not only react but may also diffuse in space. This is expressed by a system of partial differential equations (PDE) describing the change in concentrations of substances caused by both reaction and diffusion effects combined:

$$\frac{\partial\mathbf{c}(x,y,t)}{\partial t} = \mathbf{f}\big(\mathbf{c}(x,y,t)\big) + D\nabla^2\mathbf{c}(x,y,t) \tag{2}$$

The vector $\mathbf{c}(x,y,t)$ now refers to the concentration level c_i at time t of each chemical C_i at position (x,y) in space. The reaction term $\mathbf{f}(\mathbf{c}(x,y,t))$ describes the reaction kinetics, as in (1), but now expressed for each point in space. The diffusion term $D\nabla^2\mathbf{c}(x,y,t)$ tells how fast each chemical substance will diffuse in space. D is a diagonal matrix containing the diffusion coefficients, and ∇^2 is the Laplacian operator.

3 Activator–inhibitor models

Activation-inhibition models describe situations in which two competing processes take place over space and time. The first one (*short-range activation*) tends to self-enhance the process within the local neighborhood. A competing force (*long-range inhibition*), weaker but with a longer spatial range, tends to decrease the activation effect in the surrounding space. Under certain conditions, the interaction between short-range activation and long-range inhibition can lead to the formation of asymmetric spatial patterns, such as spots and stripes resembling those found on the skin of animals. Turing (1952) was the first to present a mathematical model of morphogenesis based on reaction–diffusion dynamics, including activator–inhibitor dynamics.

In chemistry, activators and inhibitors are molecules that may diffuse over space. Activators trigger autocatalytic reactions that increase their own concentration (self-enhancement),

but such effect has limited spatial range. On the other hand, activators also trigger inhibitory reactions that cause their own concentrations to decrease. Such a "negative impact" process has a reduced intensity when compared to self-enhancement, but has much larger spatial range.

Numerous alternative activator–inhibitor models are available in the literature (Bar-Yam 2003; Turing 1952; Meinhardt 1982; Murray 2003; Koch and Meinhardt 1994). They all achieve similar short-range activation and long-range inhibition effects, but differ in the chemicals needed, the way they interact, and the characteristics of the resulting patterns. In this paper we focus on two of these models: the Gierer–Meinhardt Activator–Inhibitor Model, and the Activator-Depleted Substrate Model, both from Meinhardt (1982). We describe them below.

3.1 The Gierer–Meinhardt activator–inhibitor model

One of the most widely used activator–inhibitor model is named after Gierer and Meinhardt (Meinhardt 1982), and is described by the following reaction–diffusion equations:

$$\frac{\partial a}{\partial t} = \frac{\sigma a^2}{h} - \mu_a a + \rho_a + D_a \nabla^2 a \tag{3}$$

$$\frac{\partial h}{\partial t} = \sigma a^2 - \mu_h h + \rho_h + D_h \nabla^2 h \tag{4}$$

In this model, a represents the concentration of a short–range autocatalytic substance (activator A) and h the concentration of is its long-range antagonist H, the inhibitor.

The concentration a tends to self-enhance (growth proportional to σa^2), but such growth is slowed down by the inhibitor by a factor $1/h$. The inhibitor is produced by the molecular collision of two activator molecules, and this reaction contributes to its growth by a factor σa^2. Further, both activator and inhibitor concentrations decay proportionally to their respective values. The constants μ_a and μ_h describe the rates at which each substance decays. The terms ρ_a and ρ_h represent a constant inflow of substances A and H, respectively. Molecules can move across nearby cells following the concentration gradient, hence the diffusion terms $D_a \nabla^2 a$ and $D_h \nabla^2 h$, where D_a and D_h are constant diffusion coefficients.

An important condition for the formation of asymmetric patterns in this model is $D_h \gg D_a$ (the inhibitor diffuses much faster than the activator). Another condition is that $\mu_h > \mu_a$ (the inhibitor drains more quickly than the activator). A full mathematical description of the necessary and sufficient conditions for pattern formation in activator–inhibitor systems can be found in Murray (2003).

Figure 1 (left) illustrates the typical pattern formation process resulting from this activator–inhibitor model: starting from a homogeneous mix of chemicals that is slightly perturbed, the system progressively self-organizes into spot patterns, where the spots are regions of high activator concentration. Figure 1 also shows the other models studied, which will be described later in this section. The figure was obtained using closed (reflective, non-periodic) boundary conditions on a regular grid of size 32×32 points.

This model is widespread in the literature (Bar-Yam 2003; Turing 1952; Meinhardt 1982; Murray 2003; Koch and Meinhardt 1994), but little has been discussed about its actual chemical implementation, either in natural or artificial chemistries. This is important if one would like to engineer reaction–diffusion systems using reaction–diffusion computers (Adamatzky et al. 2005), or by evolving the corresponding chemical reaction graphs (Deckard and Sauro 2004).

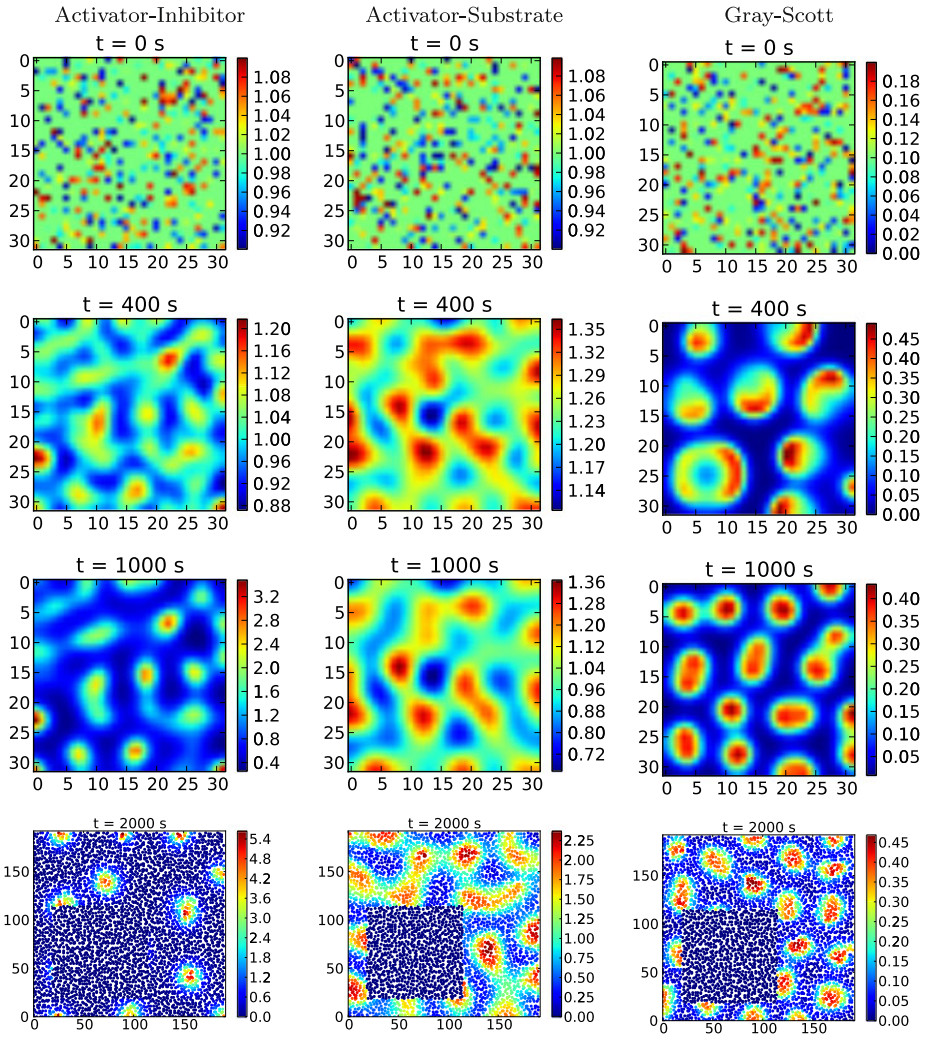

Fig. 1 Pattern formation in time, for the three models compared. Time flows from top to bottom. The plots show the level of activator concentration as a heatmap, according to the scale on the right side of each plot. The x- and y-axes indicate the grid positions

What set of chemical reactions can lead to (3) and (4)? In particular, the term $1/h$ in (3) does not seem to stem directly from mass action kinetics, neither from other well-known kinetic laws such as enzyme kinetics or Hill kinetics. Bar-Yam (2003) has pointed out that the term $1/h$ comes from the effect of a third substance, a catalyst C, which is assumed to be in steady state.

Whereas an ODE system can be directly constructed from a system of chemical reactions, currently there is no firm and generic method to do the reverse operation, that is, to infer the corresponding chemical reactions from a given ODE. We have reverse-engineered equations

(3) and (4) using the catalyst hint from Bar-Yam (2003), and obtained the following result:

$$2A \xrightarrow{\sigma} 2A + H \tag{5}$$

$$A \xrightarrow{\mu_a} \emptyset \tag{6}$$

$$H \xrightarrow{\mu_h} \emptyset \tag{7}$$

$$\emptyset \xrightarrow{\rho_a} A \tag{8}$$

$$\emptyset \xrightarrow{\rho_h} H \tag{9}$$

$$C + 2A \xrightarrow{k_1} C + 3A \tag{10}$$

$$C + H \xrightarrow{k_2} H \tag{11}$$

$$C \xrightarrow{\mu_c} \emptyset \tag{12}$$

$$\emptyset \xrightarrow{\rho_c} C \tag{13}$$

Note that the first five reactions (5)–(9) stem directly from (3) and (4), while the other reactions (10)–(13) involve a "hidden" chemical C. C acts as a catalyst in the production of the activator A, however, the inhibitor H consumes C, and it is the depletion of C through H that causes the inhibitory effect on A. This can be shown via the following analysis. We start by deriving the differential equations corresponding to the full set of reactions (5)–(13), using the Law of Mass Action:

$$\frac{da}{dt} = k_1 c a^2 - \mu_a a + \rho_a \tag{14}$$

$$\frac{dh}{dt} = \sigma a^2 - \mu_h h + \rho_h \tag{15}$$

$$\frac{dc}{dt} = -k_2 c h - \mu_c c + \rho_c \tag{16}$$

Equation (15) corresponds directly to the reaction part of (4), so we do not need to look at it further. Now we focus on how to obtain the reaction part of (3) from (14) and (16). For simplification we assume that $\mu_c = 0$, so that the inhibitor is the only responsible for a depletion of C. At steady state, the concentration of C stays stable, thus:

$$\frac{dc}{dt} = -k_2 c h + \rho_c = 0 \quad \Rightarrow \quad c = \frac{\rho_c}{k_2 h} \tag{17}$$

Substituting (17) in (14) we obtain

$$\frac{da}{dt} = \frac{k_1 \rho_c a^2}{k_2 h} - \mu_a a + \rho_a \tag{18}$$

By setting $\sigma = k_1 \rho_c / k_2$ we obtain the reaction part of (3) as needed. Note therefore that (3) is only accurate when the system is in a steady state. Perturbations may cause the catalyst concentrations to fluctuate, and therefore the approximation in (3) becomes a poor model of the system's dynamic behavior in these cases. This might partly account for this model's lower tolerance to disruptions uncovered in our simulation experiments reported in Sects. 5 and 6.

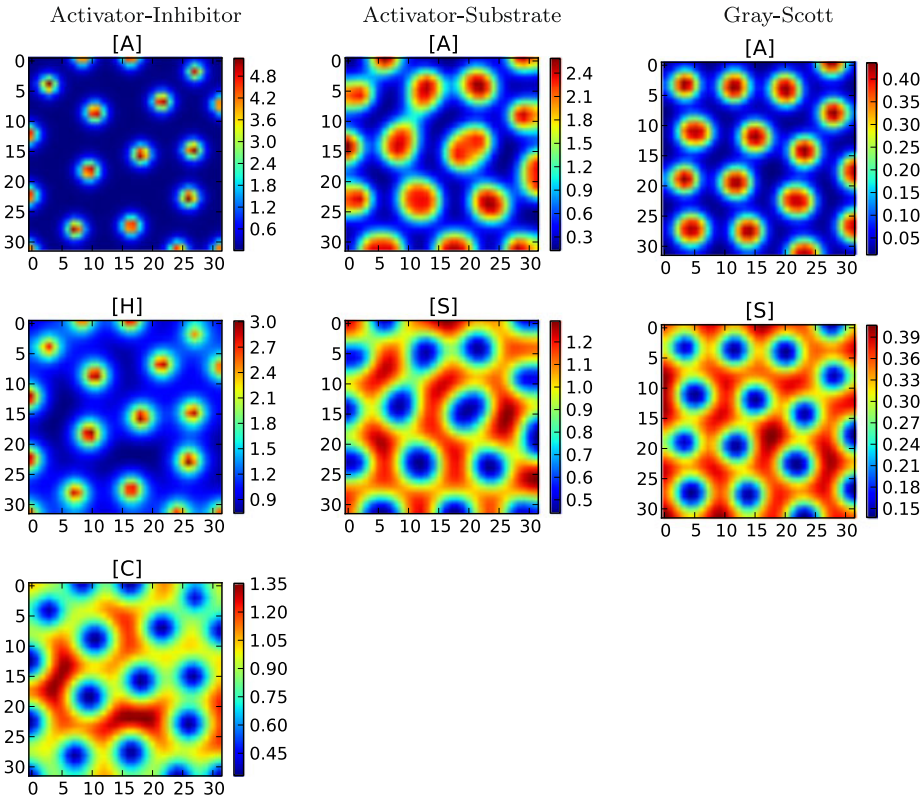

Fig. 2 Concentrations of Activator (A), Inhibitor (H), Substrate (S), and Catalyst (C) for the three models, at the end of the simulation of Fig. 1 at $t = 4000$ s

Figure 2 (left) shows the concentrations of activator, inhibitor and catalyst at the end of the simulation that generated Fig. 1 (left). We can see how the activator and inhibitor peaks coincide, the inhibitor peaks being more modest, while the catalyst valleys correspond to regions of high activator concentration.

3.2 The activator-depleted substrate model

Several variations over the basic activator–inhibitor model have been proposed in the mathematical biology literature (Dormann 2000; Meinhardt 1982; Murray 2003), some encompassing a larger number of equations, representing situations in which more complex interactions among molecules take place.

An interesting alternative approach in our context is the *Activator-Depleted Substrate model* (Meinhardt 1982), or Activator–Substrate for short. Instead of modeling the explicit presence of a molecular species able to slow down the activation process, the Activator–Substrate model achieves a similar antagonistic effect through the depletion of a substrate S, which gets consumed during the production of the activator A. The resulting reaction–diffusion equations read:

$$\frac{\partial a}{\partial t} = \sigma_a s a^2 - \mu_a a + \rho_a + D_a \nabla^2 a \tag{19}$$

$$\frac{\partial s}{\partial t} = -\sigma_s s a^2 - \mu_s s + \rho_s + D_s \nabla^2 s \tag{20}$$

The substrate S gets consumed during the autocatalytic production of activator A. Both activator and substrate are produced everywhere at constant rate ρ_a and ρ_s, respectively, and decay at rate μ_a (resp. μ_s). Figure 1 (middle) shows the typical pattern formation process using this model. Note that, in contrast to the previous model, here spots slowly change, and sometimes grow and divide, that is, they can replicate. This behavior has been thoroughly studied in the Gray–Scott model (discussed in Sect. 3.3 below) which is a special case of Activator–Substrate model.

Note that if it was not for the possibility to have different σ values for A and S (σ_a and σ_s), then it would have been straightforward to derive the corresponding chemical reactions. In order to allow for different σ_a and σ_s, we have come up with the following solution:

$$S + 2A \xrightarrow{\sigma_a} 3A \tag{21}$$

$$S + 2A \xrightarrow{k_s} 2A \tag{22}$$

$$\emptyset \xrightarrow{\rho_a} A \tag{23}$$

$$\emptyset \xrightarrow{\rho_s} S \tag{24}$$

$$A \xrightarrow{\mu_a} \emptyset \tag{25}$$

$$S \xrightarrow{\mu_s} \emptyset \tag{26}$$

where $k_s = \sigma_s - \sigma_a$. With this we obtain the reaction part of (19) and (20) from the law of mass action. Reactions (21) and (22) imply that, when two molecules of A and one of S react, one extra molecule of A may be formed in some cases (reaction (21)), while in others (reaction (22)) the substrate molecule will simply be lost or degraded into something that cannot be used by the system. So the second reaction can be seen as a kind of "error" or inefficiency in the process of autocatalysis of A, which can be nevertheless exploited to construct useful patterns.

Figure 2 (middle) shows the concentrations of activator and substrate at the end of the simulation that generated Fig. 1 (middle). The substrate valleys correspond to regions of high activator concentration. Note the wider and smoother activator peaks when compared to Fig. 2 (left).

3.3 The Gray–Scott model

The Gray–Scott Model (Gray and Scott 1990; Pearson 1993) is a variant of an Activator–Substrate model described by the following reactions:

$$U + 2V \xrightarrow{k=1} 3V \tag{27}$$

$$V \xrightarrow{F+K} P \tag{28}$$

$$\emptyset \xrightarrow{F} U \tag{29}$$

$$U \xrightarrow{F} \emptyset \tag{30}$$

where F and K are parameters of the system, and P is a product that is removed from the system. It is easy to see that these reactions are variants of the Activator–Substrate model

with $U = S$, $V = A$, $\sigma_a = 1$, $k_s = 0$, $\rho_a = 0$, $\rho_s = F$, $\mu_a = F + K$, and $\mu_s = F$. The corresponding differential equations are:

$$\frac{\partial u}{\partial t} = -uv^2 + F(1-u) + D_u \nabla^2 u \tag{31}$$

$$\frac{\partial v}{\partial t} = uv^2 - (F+K)v + D_v \nabla^2 v \tag{32}$$

The Gray–Scott model is a variant of an earlier model originally proposed to explain glycolysis, a set of biochemical reactions that breaks glucose to release energy for various cellular processes. The well-stirred Gray–Scott model is known to exhibit bistability and oscillations for a range of parameters (Gray and Scott 1990). Pearson (1993) discovered a rich variety of patterns in the spatial Gray–Scott model, including spots, stripes, hexagons, labyrinths, waves, and self-replicating spots. Since then, extensive analysis of the dynamic properties of the model has been carried out, and the conditions for the formation of the various types of pattern are well studied (Mazin et al. 1996). The interesting patterns arise in a narrow region of F and K values, near the transition between a trivial homogeneous steady state where the autocatalyst is depleted, and regions of bistable and unstable behaviors.

The phenomenon of spot multiplication is known to be driven by strong perturbations of the homogeneous steady state where U dominates and V is depleted. Spots can start with a region of sufficiently high activator concentration. The activator autocatalyses itself and diffuses to the neighborhood, causing the spot to grow, until the substrate U is depleted at the center of the spot, causing the center to fade until the spot splits into two parts, and the process is repeated from there, until the whole region is filled with spots. Overcrowding can then occur, killing several spots and leaving a blank region that other spots can repopulate. The system can also stabilize into interesting beehive-like structures where spots are hexagonally separated.

In this paper we will look at how self-replicating spots in the Gray–Scott model can be used to solve the distributed cluster head computation problem with fast recovery from pattern disruptions caused by the removal of chemicals. For this purpose, we chose a set of parameters for the Gray–Scott that produces such replicating spots and then stabilizes into a regular structure. An example is shown in Fig. 1 (right): the spots form very early and start to "replicate", that is, grow, divide in two or more smaller spots that grow again, until a stable regular structure is reached, as can be seen in Fig. 2 (right).

4 Experiments

We have performed a series of simulation experiments to evaluate the ability of the three reaction–diffusion models described to function as signaling mechanisms for distributed cluster head election algorithms. We compare the three models described in the previous section: the original Gierer–Meinhardt Activator–Inhibitor model (henceforth simply referred to as "Activator–Inhibitor"[2]), the Activator-Depleted Substrate model (or simply "Activator–Substrate"), and the Gray–Scott model. We simulate a chemical implementation of these reaction–diffusion systems, according to their corresponding chemical reactions (5)–(13)

[2]Capitals are used to distinguish this specific model from the multiple variants of activator–inhibitor models explained in Sect. 3.

Table 1 Parameters used in the experiments

Activator–Inhibitor			Activator–Substrate			Gray–Scott
$\sigma = 0.02$	$\mu_a = 0.01$	$\rho_a = 0$	$\sigma_a = 0.01$	$\mu_a = 0.01$	$\rho_a = 0$	$F = 0.01$
$k_1 = 0.01$	$\mu_h = 0.02$	$\rho_h = 0$	$\sigma_s = 0.02$	$\mu_s = 0$	$\rho_s = 0.02$	$K = 0.04$
$k_2 = 0.1$	$\mu_c = 0$	$\rho_c = 0.1$	$D_a = 0.008$	$D_s = 0.2$	$k_s = 0.01$	$D_v = 0.033$
$D_a = 0.005$	$D_h = 0.2$					$D_u = 0.2$

and (21)–(26). This is in contrast to the traditional approach using (3), (4), (19) and (20) directly, and then integrating them without looking at the actual reactions involved.

We specify the chemical reactions, together with their rates, in a text format such as "`A + 2 B -> C, k=0.1`". Each line expresses a chemical reaction. The lines are then parsed, and the corresponding stoichiometric matrices are constructed, together with their reactant and product multisets. This information is then used to drive a generic integrator (based on the simple explicit Euler method) automatically, according to (1) and (2). This system is intuitive and versatile, and can simulate any kind of reaction–diffusion system with little effort. Besides ensuring chemical compatibility, another interesting feature of such a system is to make sure that the formula and behavior obtained stem directly from the chemical reactions, without hidden assumptions or simplifications.

A peak is a point with maximum activator concentration, that is, the local concentration at location (x, y) is higher than all the concentrations in the von Neumann neighborhood (north–south–east–west cells) around the peak. Moreover the concentration at the peak must be above a threshold for it to be considered as a cluster head. In this evaluation study we consider only the peaks elected as cluster heads, therefore we use the terms peak and cluster head interchangeably. The peak concentration threshold is $a_{\min} = 3.0$ for the Activator–Inhibitor model, $a_{\min} = 2.0$ for the Activator Substrate model, and $a_{\min} = 0.3$ for the Gray–Scott model.

The parameters for the three models can be found in Table 1. These parameters were chosen in order to compare the three models studied under similar conditions, in which they form equidistant spot patterns with a similar density of spots. For the first two models, except for the extra coefficients k_i, all other constants are set to the same values as Koch and Meinhardt (1994). For the third model, the parameters are selected within the range reported by Pearson (1993) and Mazin et al. (1996). We choose the diffusion coefficients D_a (resp. D_v) for the activator empirically, such that the average number of peaks obtained and their average spacing are approximately the same in all three models, so that they can be compared. The diffusion coefficient for the inhibitor D_h (resp. substrates D_s and D_u) is the same for all the models: $D_h = D_s = D_u = 0.2$.

The diffusion process is discretized in space, as follows:

$$\nabla^2 c(\mathbf{p}, t) \approx \sum_{\mathbf{q} \in \text{Neighbors}(\mathbf{p})} c(\mathbf{q}, t) - c(\mathbf{p}, t) \tag{33}$$

for a given chemical C at point $\mathbf{p}$ with concentration c at time t. On regular grids, we use a von Neumann neighborhood, so the set Neighbors($\mathbf{p}$) of neighbors of a given point $\mathbf{p}$ are its north, south, left and right neighbors. Two types of boundary condition are possible: periodic and closed. In the case of periodic boundary conditions, the grid is considered as a torus. In the case of closed boundaries, the concentrations at non-existing neighbors at the border are taken as the concentrations at point $\mathbf{p}$ itself.

Three sets of experiments are performed: in the first set, reported in Sect. 5, small-scale experiments on a regular grid of size 32×32 are performed to look at the dynamics of the recovery process in response to successive peak and region disruptions. The results are compared against the non-chemical state-of-the art cluster head election algorithm by Basagni (1999). The second set of experiments, reported in Sect. 6, looks at fault-tolerance metrics for varying grid sizes, up to 1024×1024. The simulations of such larger grids are computationally intensive, therefore a GPU implementation is used. The third set of experiments, reported in Sect. 7, shows the recovery behavior of the three models studied in an amorphous computing scenario.

The simulations of Sects. 5 and 6 (for the reaction–diffusion models on regular grids) were performed using periodic boundary conditions. Qualitatively identical results are obtained using mass-conserving diffusion over closed boundaries.

4.1 Benchmarking against a non-reaction–diffusion algorithm

In order to provide a benchmark for the performance of the analyzed mechanisms against non-diffusion-based schemes, we have performed a comparison with a state-of-the-art clusterhead election algorithm. For this purpose, we have selected the algorithm by Basagni (1999), a distributed clustering algorithm widely used in the ad hoc networking field. This algorithm is based on a variant of the lowest-ID algorithm, first introduced by Lin and Gerla (1997). Each node has a "weight" attribute that may represent the level of available resources (computing power, batteries, etc.), or may simply be a randomly generated identifier. Each node can be in one of three states: idle, associated or clusterhead. If a clusterhead is present in the neighborhood, nodes in the idle state associate with it. Nodes promote themselves to clusterhead if (i) there is no active clusterhead in the neighborhood (ii) there is no idle node in the neighborhood with a larger weight. Basagni's algorithm automatically adapts the number of clusterheads in order to ensure that each node is associated with a clusterhead. Therefore, given a connected topology, the algorithm always converges to a configuration where each node is associated with exactly one clusterhead. The algorithm includes some procedures aimed at minimizing the number of messages transmitted. Despite being initially conceived for ensuring that each node is at most at distance one hop from its clusterhead, it can be straightforwardly generalized for handling arbitrary maximal distance values. Given as input k the maximum distance between a node and its clusterhead, the algorithm then elects a variable number of clusterheads (depending on the network topology), in order to ensure that each node is at most k hops from the closest clusterhead.

We have reimplemented the k-hop version of Basagni's algorithm in Matlab. We used $k = 6$ in our experiments, which is the same value used in the experiments of Sects. 5 and 6. In order to make the comparison fair, we had to limit the maximum number of clusterheads to 18, a value in the range of those obtained by the three reaction–diffusion models studied. This has the important consequence that the algorithm can no longer ensure convergence to a situation in which all nodes are associated, that is, served by a nearby clusterhead. This simulates a situation in which the amount of clusterheads is a limited resource, while the adaptive case would require that as many nodes as needed be elected as clusterheads, which would be inconsistent with the studied reaction–diffusion models, for which the amount of peaks formed depends on their (fixed) parameter values.

Our implementation of Basagni's algorithm works in rounds. Each round corresponds to the time taken by each node to propagate its status information (whether it is associated, clusterhead or non-associated) to its k-hop neighborhood. To compare the dynamics of Basagni's scheme with those obtained by the diffusion-based mechanisms considered here,

we set the duration of each round equivalent to 250 s of simulated time in the diffusion-based models. The duration of a round was chosen such that the convergence times in the absence of perturbation were similar to those obtained in the reaction–diffusion models. Moreover a sufficient number of rounds was needed to recover from perturbations introduced every 1000 seconds as will be explained in the next section.

Note that the simulated time scales used are arbitrary, since they are determined by the values of the kinetic rate parameters and the units used. As long as we are not simulating real chemistry, it is possible to adjust the time scales of the different algorithms so that they become comparable. In a real implementation on top of traditional silicon hardware, the time scales would probably be of the order of milliseconds, not hundreds of seconds like in our case, which is more similar to what would be obtained with a wet implementation on real chemistry.

5 Dynamics of the recovery process

In this section we compare the ability of the three models to recover from perturbations that delete chemicals at given locations. For benchmarking purposes, as explained in Sect. 4.1, Basagni's algorithm is also included in the comparison. The simulations are performed on a regular grid of 32×32 cells with periodic boundary conditions, using an integration timestep of $\Delta t = 0.1$ s. Two scenarios are tested for each model: with and without perturbation. In the scenario without perturbation, the simulation runs without interference until it finishes at 10,000 simulated seconds. In the perturbed scenario, at $t = 4000$ s, after the system has reached a stable pattern, perturbation events are introduced every 1000 seconds.

Two types of perturbation event are introduced: peak and region deletion. In the peak deletion scenario, each perturbation event happens as follows: 50% of the existing peaks are selected at random for disruption. If selected, a peak loses 90% of its activator concentration, and the same percentage is removed from the Moore neighborhood (8 surrounding cells) around the peak. This scenario is intended to evaluate the ability of the schemes to recover from activator peak deletions: since they signal the location of cluster heads, these peaks are the most important signals in the system, therefore it is important that they are regenerated when disrupted.

In the region deletion scenario, a whole rectangular patch covering 25% of the surface of the grid is deleted: all chemicals in this region are removed entirely. This scenario simulates severe damages, such as those caused by stepping on full regions of chemicals on the computation surface, and is intended to evaluate the ability of the schemes to repopulate these areas after the damage. For Basagni's algorithm, the nodes selected for deletion are brought back to the idle state.

Figure 3 shows an example of activator peaks elected as cluster heads using the algorithm. The elected peaks are indicated by crosses. The other dots indicate catalyst and substrate peaks that play no cluster head role. The Gray–Scott model does not appear in Fig. 3 because it results in a similar peak configuration to that shown for the Activator–Substrate model (of which it is a special case). The difference in the Gray–Scott case is the faster spot dynamics (growth, motion, fusion, shrinking).

The performance of each algorithm is measured by the average distance between nearest peaks, the total number of elected peaks, and the number of cells that are served by a peak. The first two metrics are self-explanatory; the third measures the number of cells that are located at a distance shorter than a threshold of the nearest peak, that is, they can receive a service from this peak, such as the aggregation of some sensor information. We calculate the

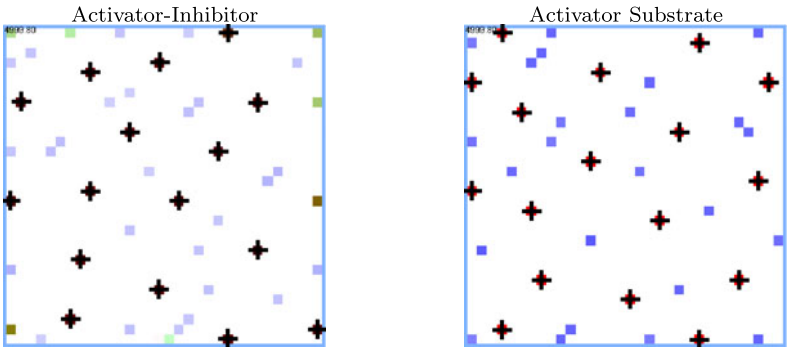

Fig. 3 Cluster head election example for two of the models studied, at $t = 5000$ s. Activator peaks (marked with *crosses*) play a cluster head role. Other *dots* indicate catalyst or substrate peaks. The Gray–Scott model (not shown) exhibits a peak configuration similar to the Activator–Substrate case, except that peaks tend to form earlier and move faster

average distance between nearest peaks by looking at the four nearest peaks of each peak, and calculating the average of this distance for all peaks.

Figure 4 shows the average distance between nearest peaks, comparing the various models. We can see that this distance remains stable when there are no disruptions. Under peak perturbations, the Activator–Inhibitor model is less stable, presenting bigger fluctuations than the other models. Note, however, that while the peak distance in the Activator–Inhibitor and Basagni's models remain totally constant in the absence of perturbations, in the other two models this distance fluctuates slightly, even in the absence of perturbations. This indicates that peaks tend to move a bit, as the underlying spots merge, grow and divide. This can be more easily seen by observing the number of peaks elected as cluster heads in Fig. 5. Under perturbation, the number of peaks in the Activator–Substrate and Gray–Scott models appear rather erratic, however, these fluctuations may actually turn out to be advantageous, as they reflect a quick adaptation process going on to reoccupy the space left in the disturbed areas. Indeed, this behavior can be easily observed when watching animations of both the Activator–Substrate and the Gray–Scott systems.

Figure 6 depicts the average distance from any cell to its nearest peak, reflecting the amount of cells that are served: if a cell is at a distance of less than $d_{\max} = 6$ cells from the nearest peak, then this cell is considered as not served. The threshold $d_{\max}$ has been set to a little more than half of the average distance between nearest peaks, such that without disruptions, any cell should have a good chance to find a peak within this threshold distance. Under such constraint, configurations leading to regularly spaced peaks have an advantage. Note that this threshold is set very low on purpose to introduce abnormally high error rates in order to stress the system to the most and make the failures more apparent such that they can be more easily detected and measured.

Except for Basagni's model (discussed in more detail in Sect. 5.1 below), the peak and cell distance plots on Figs. 4 and 6 resemble each other, consistently with the fact that the peak distance has a direct influence on the distance between a cell and its nearest peak. Cells within this service distance $d_{\max}$ are served by a nearby peak. The number of cells that are not served is displayed in Fig. 7. Here we can see that, under peak perturbation, the Activator–Inhibitor model clearly leaves a larger number of cells without service than the other models, and that under region perturbation, the Activator–Inhibitor model is less reactive. Interestingly, in the absence of perturbation, the service ratio of the Gray–Scott model

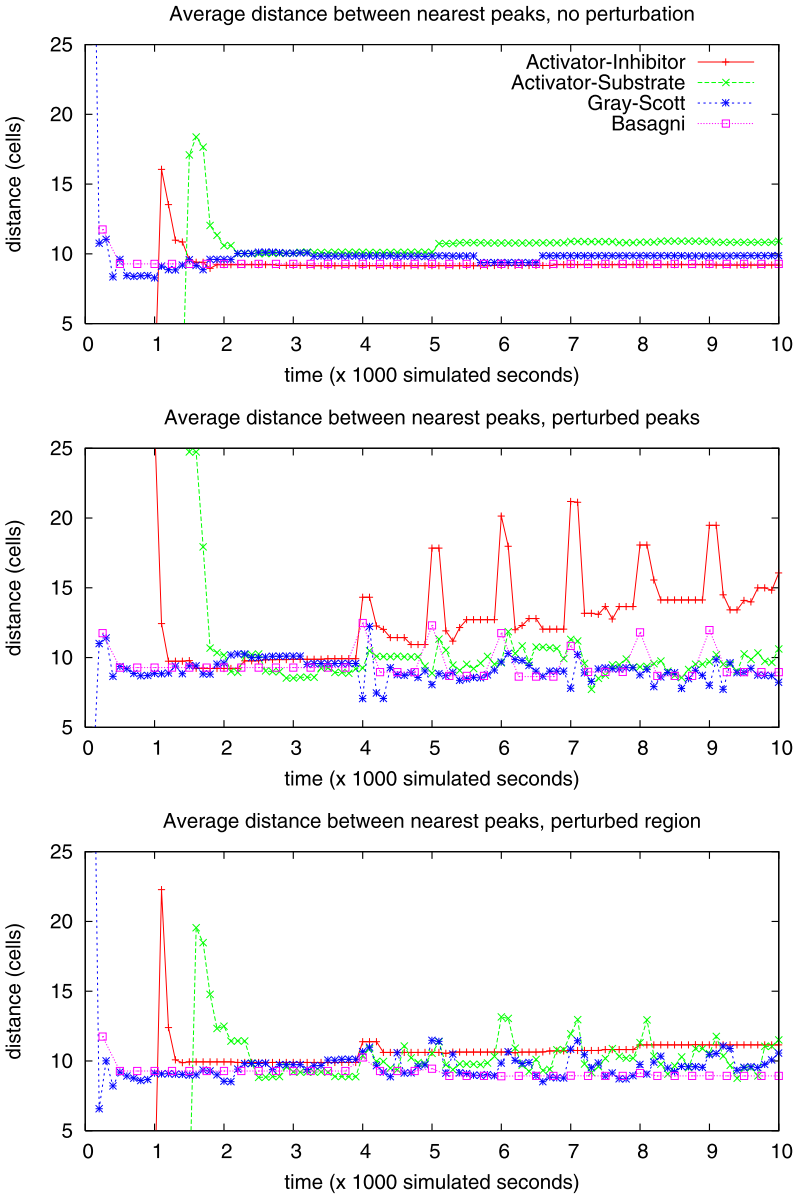

Fig. 4 Average distance between nearest peaks, for each of the four models compared, with and without perturbation

reaches nearly 100%, while the others always leave about 5% of the cells without service. Note that although 5% non-served might appear as a large value, as explained earlier, this is due to the artificially low distance threshold set on purpose to make the failures more apparent for comparison purposes. For a higher distance threshold (Yamamoto and Miorandi 2010) the amount of non-served cells drops to zero for all models in the unperturbed case.

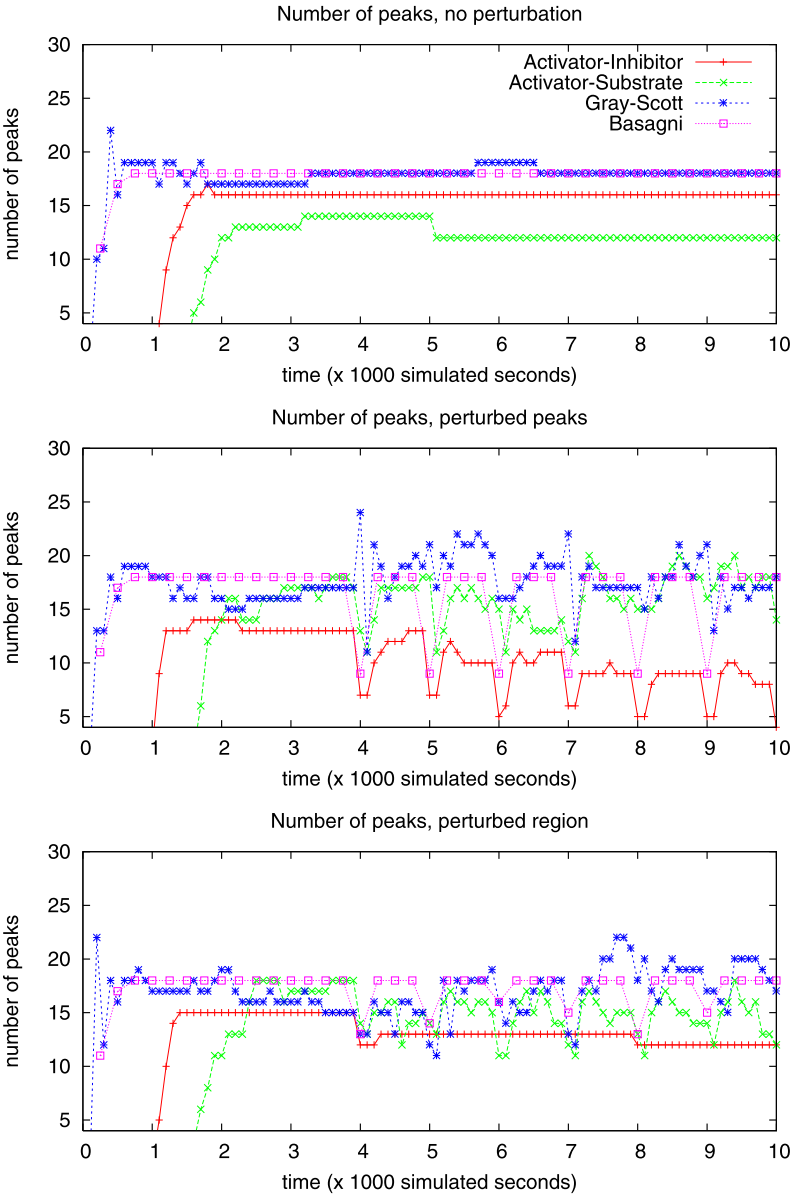

Fig. 5 Number of peaks elected as cluster heads, for each of the four models compared, with and without perturbation

5.1 Comparison with Basagni's algorithm

In this section we look more closely at the comparison between the three reaction–diffusion models studied and Basagni's non-diffusion-based clustering scheme. We make the following observations.

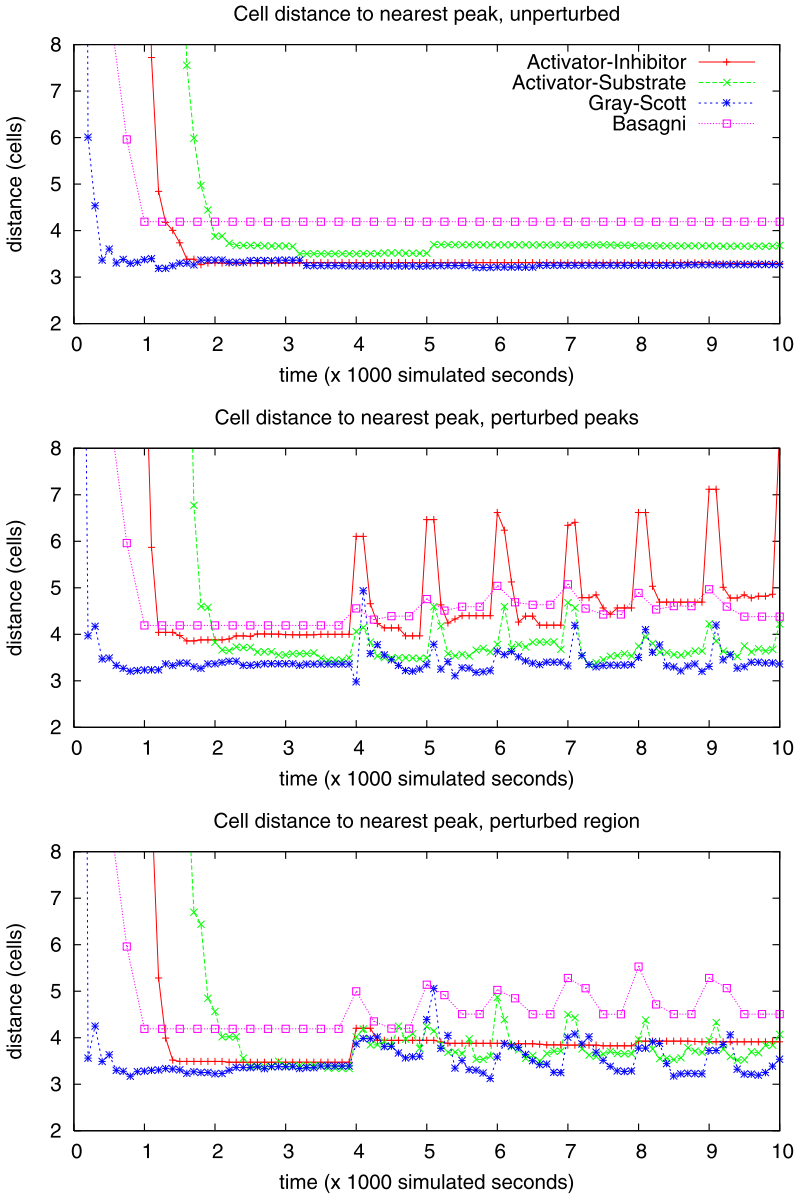

Fig. 6 Average cell distance to its nearest peak, for each of the four models compared, with and without perturbation

– The performance in the absence of perturbations (in terms of distance to peak and distance among neighboring peaks) is slightly better for the diffusion-based schemes when compared to Basagni's one. This comes from the fact that Basagni's mechanism, when limiting the number of peaks to be elected, behaves in a greedy fashion until it stops when it has reached the maximum number of peaks. In such circumstances, it may leave some regions of the network uncovered by peaks. The diffusion-based mechanisms, on

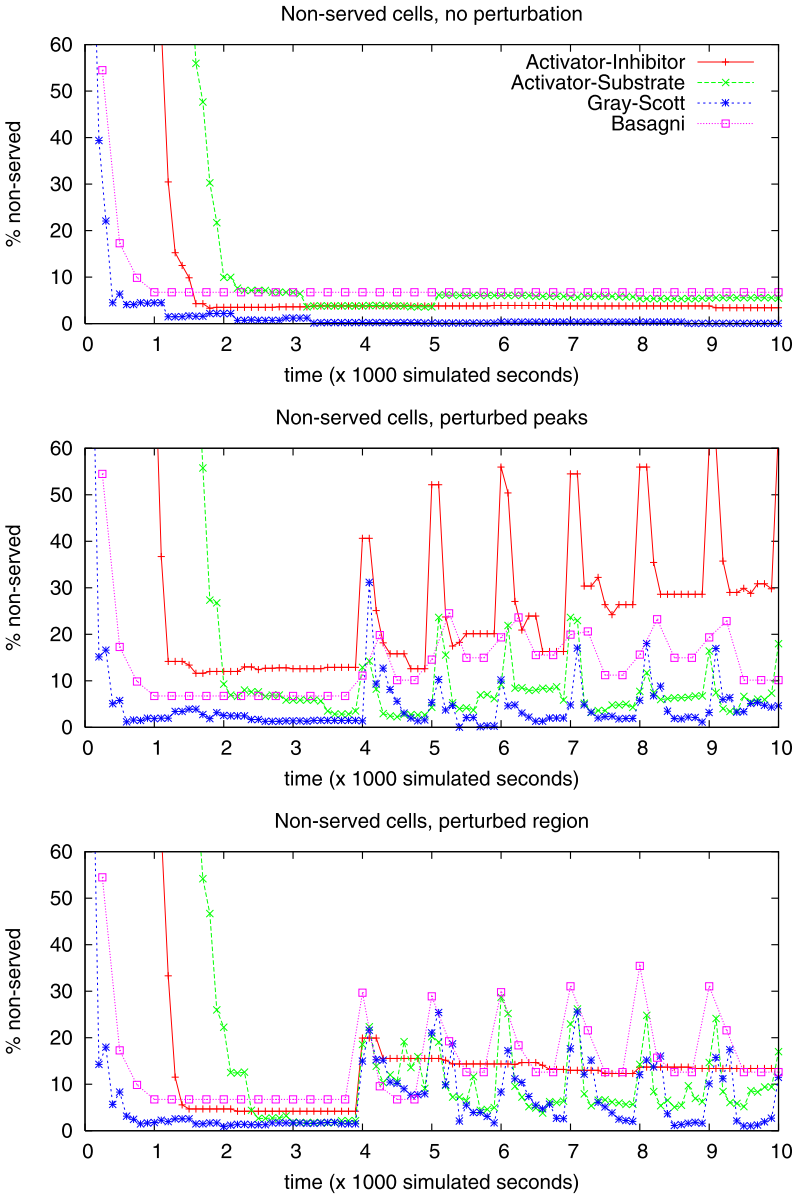

Fig. 7 Percentage of non-served cells, for each of the four models compared, with and without perturbation

the other hand, tend to spread peaks evenly in the region, thereby leading to slightly better performance.

– Basagni's scheme performs worse under peak deletion than under region deletion. This is because this algorithm suffers in the absence of valid clusterheads: recall that in the peak deletion case, 50% of the peaks are moved to the idle state, compared to 25% of the peaks in the region deletion case.

– Under carefully chosen comparable parameter settings, the behavior of Basagni's scheme in terms of recovery from perturbations that remove chemicals is similar to that offered by reaction–diffusion mechanisms in terms of both qualitative behavior as well as quantitative one (impact of perturbation on the considered metrics and recovery dynamics).

These results indicate that reaction–diffusion techniques can exhibit a recovery behavior similar to that of non-diffusion-based clustering techniques, with respect to the studied failure modes, provided that the parameters can be chosen such that their dynamics fall into comparable ranges.

Note that this tells nothing about the performance of these algorithms in terms of resource consumption (that is, number of messages exchanged between nodes, amount of memory or CPU time consumed, etc.). The accurate simulation of physical and chemical processes is computationally intensive, and reaction–diffusion is not an exception, as will become apparent in Sect. 6.1.

6 Fault tolerance measurements on larger grids

The results presented in the previous section were obtained on a small grid of size 32×32. While we expect that these results are qualitatively representative of what happens in these models in general, it is interesting to measure their fault tolerance behavior on larger grids, to find out whether the fault recovery ability of each model is affected by the size of the grid. In order to provide an answer to this question, each model must be repeatedly evaluated on several grid sizes. However, this is a computationally intensive task. In order to cope with it, we have reimplemented the reaction–diffusion PDE integration scheme, and the peak detection and cluster head election algorithms using GP-GPU (General-Purpose Graphics Processing Units) with NVIDIA's CUDA architecture.

6.1 Reaction–diffusion and cluster head election on a GPU

With GPU cards becoming more affordable and easier to program, parallel programming on top of GPUs becomes an attractive alternative to large-scale expensive computer clusters. However, not all tasks can fully benefit from the parallelism provided by GPUs: GPUs have a SIMD (single-instruction, multiple-data) architecture, in which each single processor is able to process multiple data items in parallel using the same instruction. Therefore, the tasks that are good for GPUs are those that must handle multiple data items using the same flow of instructions. PDE integration is exactly such a task: the same differential equations and diffusion rules apply to all the points of the grid.

Sanderson et al. (2009) present an extensive discussion and evaluation of GPU approaches to parallel PDE solving for advection–reaction–diffusion equations, a PDE system that also takes into account advection phenomena in which chemicals also move by fluid transport, such as pollutants in the air or water. Molnár et al. (2010) show an implementation of reaction–diffusion in three dimensions. Although the authors of both papers have published their source code, both are far more elaborate than what we need for our own experiments. Therefore we have decided to reimplement our own solution from scratch, based on our initial Breve–Python prototype that we had developed for the earlier version of this work (see Yamamoto and Miorandi 2010 for details).

Our own GPU-PDE implementation is simple: the set of reactions is parsed as before and converted to the corresponding stoichiometric reactant and product coefficient matrices, plus a vector of reaction rates and diffusion coefficients. The initial conditions for each model are

generated on the host computer, and copied to the GPU device, together with the parameters. The PDE integration routine is then launched on the GPU, where each thread is responsible for integrating the reaction–diffusion equations for one position on the grid of cells. The stoichiometric matrices, reaction and diffusion coefficients are placed in the local shared memory. This results in considerable performance improvements, as frequently reported in the GPU literature.

The experiments were run on a host containing three NVIDIA GTX 480 GPU devices. The speedup obtained with the GPU implementation was more than 100 times the CPU runtime, which is already at least another order of magnitude faster than our earlier Breve-Python prototype. One run of 100,000 iterations for the mesh size 1024×1024 still took about 20–30 minutes to complete on a single GPU card, but such a run would have been infeasible without our parallel solution.

6.2 Fault-tolerance metrics

A number of well-known metrics can be employed to measure the fault tolerance of a system:

- MTBF (Mean Time Between Failures): measures how long it takes on average before a component fails, that is, the interval between consecutive failures. In our case, a component corresponds to one grid cell, or one processor in the distributed system, and a failure corresponds to the lack of service for this cell, that is, the cell is in failure mode when it is at a distance larger than $d_{\max}$ from a peak. The MTBF in our case corresponds to the interval between two consecutive transitions from served to non-served state for a cell, averaged over all cells during the whole lifetime of the experiment.
- MTTF (Mean Time To Failure): measures the average time that the system remains operational until it fails, that is, its average uptime. In our case, this corresponds to the time each cell remains served until it falls to non-served state, averaged over all cells during the whole duration of the experiment.
- MTTR (Mean Time To Repair): measures how long it takes on average to repair a failure. The following relation holds: MTBF = MTTF + MTTR. In our case, the MTTR measures how long it takes for a cell to receive service again from a nearby peak, after it has fallen into non-served state; again, this quantity is averaged over all cells in the grid.
- Availability (percentage of uptime): measures the probability that the system is up and running correctly at any time t, which is equivalent to the percentage of uptime over the whole system lifetime. The availability $A(t)$ can be calculated as follows: $A(t) =$ MTTF / (MTTF + MTTR) = MTTF / MTBF.
- Unavailability (percentage of downtime): measures the proportion of time the systems is down over its lifetime, that is, $U(t) = 1 - A(t)$.

Achieving a high availability is of utmost importance for most manufactured items in industry, from light bulbs to airplane parts, including software. Sometimes, a low reliability can be compensated by a short time to repair, such that a high availability can be achieved in spite of frequent failures.

6.3 Measuring the fault tolerance of the three models considered

We have measured the availability and the MTBF for the 3 models studied in this paper, for increasing grid sizes. Given these two metrics, the other metrics can be easily computed, such as the downtime, MTTF and MTTR. Therefore these two metrics together are able to characterize the fault-tolerance behavior of the system in a quantitative way.

The fault-tolerance metrics were measured in the following way: at every time step Δt, each cell (GPU thread) checks whether it is up (peak or served cell) or down (otherwise). If it is up, the interval Δt is added to the total uptime t_u for the cell. If it is down but was up in the previous iteration, then the number of failures n_f for this cell is incremented. At the end of the simulation ($t_e = 10{,}000$ s), the pattern is reset, new initial conditions are generated and the simulation is repeated, with a total of $n_r = 10$ rounds of 10,000 s each. The availability is then computed as the average uptime per cell divided by the total accumulated simulation time, and then converted to percentage to give the percentage of uptime per cell. The MTBF is computed as the total simulation time divided by the cumulative amount of failures per cell:

$$A = \frac{\sum_{\forall \mathbf{p}} t_u(\mathbf{p})}{n_r t_e} \tag{34}$$

$$N_\mathrm{f} = 1 + \sum_{\forall \mathbf{p}} n_f(\mathbf{p}) \tag{35}$$

$$\hat{N}_\mathrm{f} = N_\mathrm{f}/N \tag{36}$$

$$\mathrm{MTBF} = \frac{n_r t_e}{\hat{N}_\mathrm{f}} \tag{37}$$

where N is the total number of cells in the grid, N_f is the total number of failures for the whole system during the complete duration of the experiment, and $\hat{N}_\mathrm{f}$ is the average number of failures per cell. This method enables us to calculate both quantities over long periods with only two cumulative variables t_{up} and n_f per cell. The value of N_f (35) is at least one, in order to avoid a division by zero in (37).

The failure mode is the same as the region disruption mode described in the previous section: after $t = 4000$ s (when the pattern is expected to be stable), at every interval of 1000 s, the activator and inhibitor (resp. substrate) chemicals are deleted in a square region covering 25% of the grid surface. This failure mode is expected to lead to recovery times that depend upon the grid size, since the larger the grid, the larger the surface removed, and the longer it will take for the remaining chemicals to repopulate the damaged area.

Note that an interval of 1000 s between perturbations does not imply MTBF = 1000 s, since not all cells are equally affected by the perturbation, and the recovery times for each cell vary depending on the algorithm used, and on how severely the cell was affected by the perturbation. If all cells would fail after a perturbation event, and afterwards recover within this 1000 s period before the next event, then we would indeed have MTBF = 1000 s, which can be regarded roughly as a worst-case lower bound on the MTBF value.

Figure 8 shows the measurement results for square grids with sizes varying from 32×32 to 1024×1024, for the case without disruption (left) and the case with square region disruption (right). The exact grid x and y dimensions used were 32, 64, 128, 256, 384, 512, 640, 768, 896, 1024. As explained in Sect. 6.2, the availability and MTBF values are average values measured per individual grid cell.

As expected, the undisrupted system is not affected by the grid size. Under successive region disruptions, the availability per cell decreases with the grid size for the three models, until it reaches a minimum at around $x = y = 384$. Beyond this size, the availability does not change much, indicating that the system is not able to fully recover from the damaged patch in the time interval of 1000 s before the next disruption occurs. Recall that the successive disruptions occur exactly at the same place, so the system is constantly trying to reconstruct this area. This can be confirmed by the snapshots of the system taken at the end of the run,

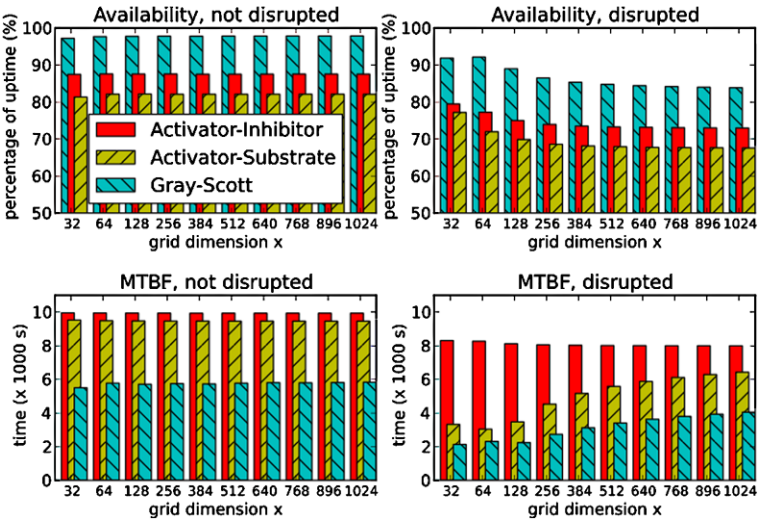

Fig. 8 Availability (percentage of uptime) and Mean Time Between Failures (MTBF) per grid cell, for the three models

shown in Fig. 9. For smaller grid sizes, spots start to occupy the damaged patch, more slowly for the Activator–Inhibitor model, faster for the Gray–Scott model (for the 64×64 mesh, no disruption can be noticed any more for this model, at the end of the simulation, that is, 1000 s after the last disruption event).

Here again, a trade-off between the models is revealed. The Gray–Scott model has the highest uptime, but the lowest MTBF, therefore it fails very frequently but also recovers very quickly. The Activator–Inhibitor model has the highest MTBF, remaining very stable even in the presence of disruptions. However, its uptime is much lower than the Gray–Scott model, showing that although it fails less frequently, when it fails it takes a long time to recover. The Activator–Substrate model has the poorest uptime, and although it has a high MTBF in the absence of disruptions, when disruptions occur its MTBF is severely affected, indicating that this system is very sensitive to region disruptions, which may cause a frequent collapse of peaks while the system struggles to find a new stable configuration. The apparently higher MTBF for higher grid dimensions is probably an artifact of cells being unable to recover on time from large patch deletions: when a whole region does not recover, no failure events occur there (since all nodes there are already down), so the failure interval may appear artificially large, due to a large portion of downtime comprised in the interval. This observation highlights the importance of taking both uptime and MTBF into account when quantifying fault tolerance.

Although the results shown were obtained using periodic boundary conditions, identical results are obtained using mass-conserving diffusion over closed boundaries. As the results in Fig. 8 show (recovery from deletion of large patches of chemicals) the models can tolerate small temporary leaks of mass through the borders. A significant loss of mass through open diffusion at the borders would pose a problem though, and in this case a chemical model would probably not be an appropriate solution.

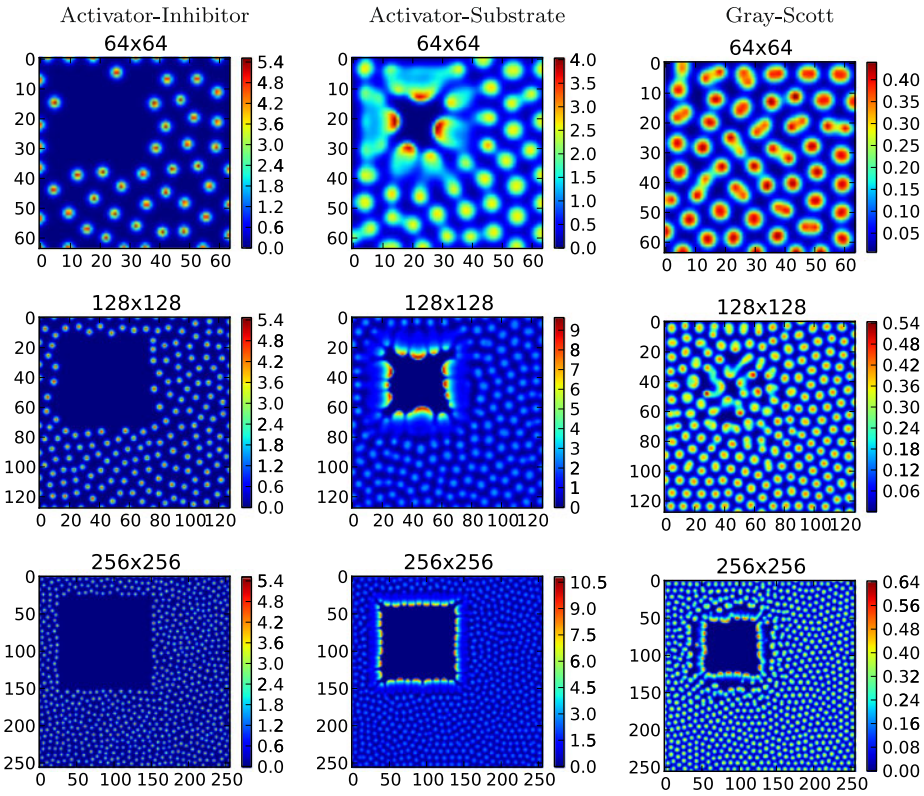

Fig. 9 Recovery behavior of the three models for varying grid sizes. *Top*: 64 × 64. *Middle*: 128 × 128. *Bottom*: 256 × 256. The plots show the levels of activator concentration, according to the scale on the right side of each plot. The x- and y-axes indicate the grid positions

7 Amorphous computing

In this section we show that the same qualitative behavior of the three models also holds for an amorphous computing scenario where a set of tiny processors or particles are placed at random positions on a surface. These processors compute the reaction–diffusion integrator. This result is expected since (Rauch 2003) has shown that the pattern formation process typically encountered on regular grids also occurs in amorphous computers.

The reaction–diffusion parameters used for these simulations are the same as those shown on Table 1, except for the Activator–Inhibitor model, where, for visibility purposes, the diffusion coefficients were multiplied by two, resulting in $D_a = 0.01$ and $D_h = 0.4$ (otherwise the activator peaks were too sharp and difficult to see on the amorphous model).

The method used to generate an amorphous topology is to apply a small random motion to the otherwise regularly spaced particles, akin to a Brownian motion. This decreases the probability of obtaining disconnected islands of particles, which would cause discontinuous regions of pattern formation. When a disconnected topology is detected, a new one is generated until a fully connected topology is obtained. We have also implemented the amorphous topology generation method suggested by Rauch (2003), but it still generated a significant proportion of disconnected topologies. A better method would have been to fix the topology

until it becomes valid, but this would have required more sophisticated algorithms, and in practice the Brownian motion method adopted was sufficient for our purposes.

The implementation of the reaction–diffusion integrator is very similar to the case of a regular grid: only the diffusion process changes, as it must now take into account the amorphous neighborhood. Each node now has neighbors placed at various distances from itself. Each node considers as its neighbors all the other nodes placed within a radius of less than $r = 5$ points of itself. For each node, the diffusion term of the reaction–diffusion equation is computed as explained by Coore and Nagpal (1998): the same formula as shown in (33) is applied, but now the points $\mathbf{p}$ are dispersed at random positions on the surface, and the set Neighbors($\mathbf{p}$) is the set of points within radius r of $\mathbf{p}$. There are no special boundary conditions, the nodes at the edges of the surface simply have less neighbors.

This diffusion method guarantees mass conservation, that is, the total amount of chemicals present on the surface before and after the diffusion step do not change (provided that the diffusion coefficient D_c and the time step Δt are small enough, such that the concentrations do not become negative). It also ensures that the system will converge to an equilibrium where the concentration of each diffused substances in each node (in a closed system and in the absence of further reactions) is the average concentration of the whole surface, whatever the topology used; and, moreover, that this equilibrium is stable (as shown by Meyer and Tschudin 2009).

Figure 10 shows the recovery behavior for the three models on an amorphous topology of 4096 nodes spread on a surface of size 192×192 points. Only the activator concentrations are shown. Time flows from top to bottom. On the top side, we see the formed pattern at $t < 2000$ s just before the patch disruption occurs. After that we see the disrupted pattern at $t = 2000$ s. The state of the recovering pattern at $t = 2400$ s is shown next. The state at the end of the simulation ($t = 4000$ s) appears at the bottom. Still in this case, we have a slower to faster reaction as we move from Activator–Inhibitor to Activator–Substrate, then to Gray–Scott.

8 Applications and related works

Reaction–diffusion systems naturally lend themselves to a distributed implementation, inherently scalable since they are based on local exchange of messages (which could be thought of as carrying molecules, thereby simulating the diffusion process) and local decision-making (which corresponds to the reaction process). Therefore they offer a natural metaphor for scenarios in which a number of autonomous entities need to interact, based on local communications only, in order to achieve a given desired global behavior. Adamatzky et al. (2005) introduce Reaction–Diffusion Computers as unconventional models of computation that exploit reaction–diffusion phenomena in real chemical media. Bandini et al. (2005) formalize the Reaction–Diffusion Machine (RDM), and prove that it is able to simulate a Turing Machine. Such results encourage further research in this domain. This new way to look at distributed computation may find many applications in swarm intelligence and distributed systems in general. In this section we review such applications, with particular focus on swarm robotics and networked devices.

8.1 Applications to swarm robotics

A robot swarm consists of a set of autonomous mobile robots that can collaborate in order to achieve a given global goal (Bonabeau et al. 1999). If coordination tasks within a swarm can

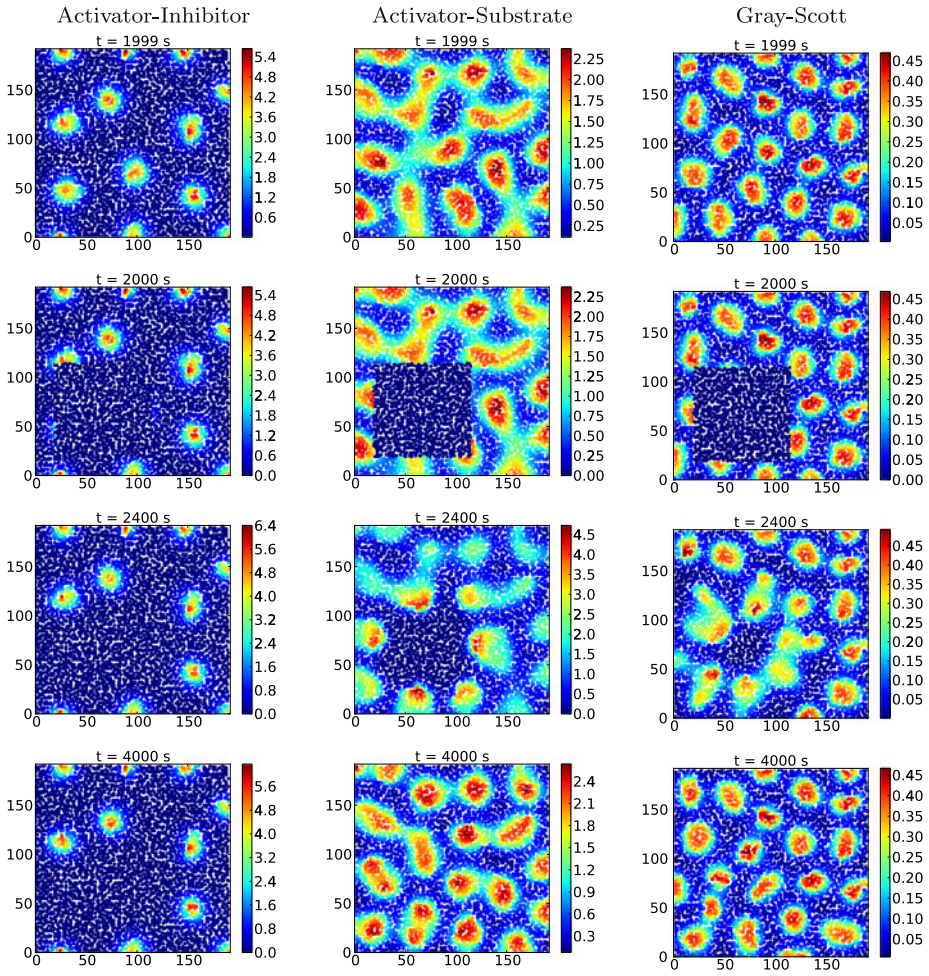

Fig. 10 Recovery behavior of the three reaction–diffusion models under amorphous computing. From top to bottom: $t = 1999$ s (before perturbation), $t = 2000$ s (when a perturbation is introduced), $t = 2400$ s (recovering), and $t = 4000$ s (end of simulation). The plots show the levels of activator concentration, according to the scale on the right side of each plot. The x- and y-axes indicate the grid positions (on a grid surface of size 192×192)

be mapped to a given configuration pattern, then the concentration patterns resulting from reaction–diffusion can be used to signal given locations and gradients to the swarm, which can then use these signals to achieve the target pattern in a decentralized way.

In the context of single robots, new robot controllers have been proposed that are either chemically inspired (Arena et al. 1999; Ziegler and Banzhaf 2001; Dale and Husbands 2010) or chemically based (Tsuda et al. 2007; Adamatzky et al. 2003; Ferrández et al. 2010). The chemically based ones aim at soft non-silicon robots or hybrid biological-electronic devices, their navigation in unknown and noisy environments, and their interface with natural cells and organisms, among other applications. Among these various chemically inspired and chemically based approaches, Arena et al. (1999), Dale and Husbands (2010), Adamatzky et al. (2003) are based on reaction–diffusion. For instance, Adamatzky et al. (2003) use the

traveling waves produced by the Belousov–Zhabotinsky reaction, while Dale and Husbands (2010) evolve the parameters of a one-dimensional Gray–Scott model to replace a neural network for control.

In the context of swarm robotics, Shen et al. (2004) have used activator–inhibitor dynamics to devise hormone-inspired self-organizing mechanisms for robotic swarms. According to such an approach, robots are modeled as cells which can diffuse signals (termed "digital hormones") within their neighborhoods. Robots react to the signals received by adapting their current trajectories accordingly. In such a way, complex spatial configurations can be achieved. Such an approach turns out to be extremely powerful, and to match well to applications whereby reactions depend on both the local topology as well as state information. Lately, Rubenstein et al. (2009) have extended the digital hormone model to address self-organization and self-assembly of multi-robot systems. Such approach is based upon the "Robotic Stem Cells" concept, representing a robot unit capable of performing a plurality of tasks without an a priori specification of the task to be performed at runtime. In such a way, self-healing and robustness can be obtained by getting robots to reconfigure themselves autonomously to preserve the swarm functionality in spite of unforeseen errors and/or attacks. In another swarm application, Fatès (2010) combine a simplified form of reaction–diffusion with chemotaxis in a swarm of agents solving a decentralized gathering problem.

The silicon implementations of many of the reaction–diffusion schemes cited above typically use very simplified simulations of the typical patterns formed by activation-inhibition. These simulations are usually based on Cellular Automata (CA), Cellular Neural Networks (CNN), or Gaussian curves. In the context of a real or artificial chemistry implementation of such schemes, relying on chemical signaling, the results presented in this paper could serve as useful guidelines for the choice of an appropriate scheme, taking into account their fault recovery properties. For instance, the stability of the activator–inhibitor model may be preferable in more static environments, while the activator–substrate and Gray–Scott models may offer interesting solutions in environments subject to frequent disruptions of chemicals.

8.2 Applications to networking

Networks of communicating devices also map well to the parallel nature of reaction–diffusion systems. Some typical application domains include routing, congestion control, and clustering problems as covered in this paper.

Durvy and Thiran (2005) proposed the use of activation-inhibition mechanisms to create transmission patterns in ad hoc wireless networks, in order to regulate access to the channel by exploiting spatial reuse.

Yoshida et al. (2008) have applied a reaction–diffusion system to congestion control in wireless mesh networks. In this system, the reaction part changes the activation level of a node depending on the current buffer occupancy. The buffer size has a positive impact on the activation level. The diffusion part is used to limit channel contention among neighboring units. Information on the current level of activator and inhibitor are communicated by means of broadcasting. In this way, nodes with a larger number of packets are given higher priority in accessing the channel, thereby effectively reducing congestion in the network.

Similar activation patterns can find applications in clustering problems in wireless sensor networks. This problem has been addressed by Neglia and Reina (2007). The authors have devised a set of parameters for the Gierer–Meinhardt model, such that activation patterns respect a constraint on the distance among maxima in the activation level. Simulation results confirm that their approach leads to the desired patterns, and moreover, show that it can outperform a probabilistic method whereby each node decides with a given probability (independently of its neighbors) to activate.

Activation patterns in wireless sensor networks were also considered by Henderson et al. (2004), where activation stripes are created to drive the movement of mobile robots (targeting mainly environmental monitoring applications). Their approach is based on the use of Thomas' model, a substrate-inhibition model widely studied in mathematical biology (Murray 2003). They also propose the use of a pure diffusion model (without reactions) to create regular patterns (in particular: stripes).

The use of an activator–inhibitor mechanism combined with anisotropic diffusion in a CNN model was considered by Lowe et al. (2009) to devise a distributed mechanism for building "data highways" in dense wireless sensor networks. They aim at finding efficient routing structures for many-to-one (or many-to-some) communications patterns in wireless networks. The resulting venation patterns are able to orient the highways toward the data sink(s).

The aforementioned applications are based on silicon hardware, and therefore must either resort to simplifications such as CA, CNN or Gaussians, or face the computational load of reaction–diffusion simulations, as we have shown in this paper. As an alternative, real chemical signaling could be embedded in the environment, upon which the electronic devices could rely to make coordinated decisions. In this case, or when the needed computational costs can be afforded and justified, the choice of the proper chemical implementation could be guided by the results presented in this paper, as discussed in Sect. 8.1 for the swarm robotics case.

8.3 Other relevant applications

Other algorithms that have been implemented using natural reaction–diffusion processors include image processing in vitro, and the calculation of spanning trees, shortest paths, and Voronoi diagrams (Adamatzky et al. 2005). There are also applications of natural reaction–diffusion to nanotechnology, such as the fabrication of structures and devices at very small scales (Grzybowski et al. 2005).

In the silicon domain, Yoshida et al. (2005) have applied activation-inhibition to the autonomous coordination of a distributed camera surveillance system, where the objective is to decrease the area of blind spots in the whole surveillance area. By means of numerical simulations, the authors show that their system presents interesting robustness properties with respect to removal/rearrangement of cameras. A similar application is considered by Hyodo et al. (2007), where a simple reaction–diffusion system is proposed to track the movement of targets and to adjust the video coding rate accordingly. With respect to our work, the same considerations presented in Sects. 8.1 and 8.2 apply to this case as well: depending on the expected failure rate, activator–inhibitor or activator–substrate systems could be used to provide the appropriate signaling when applicable. For instance, systems based on the Gray–Scott model look particularly appealing for fast bootstrap and recovery from disruptions in chemical concentrations.

9 Conclusions

Activation-inhibition models have been used as models of distributed computation, providing inspiration to solve coordination problems such as the placement of cluster heads and the formation of content delivery highways. However, their parametrization is not simple, and performance trade-offs have to be faced.

The results reported in this paper can serve as guidelines in the design of robust algorithms based on spot patterns obtained via activation-inhibition, on top of real or artificial

chemistries. The robustness of artificial chemistry simulations of three variants of activator–inhibitor models was evaluated against perturbations that delete peaks or regions of chemicals. One of the models is based on an inhibitor that consumes a catalyst necessary for the activator to grow, and the two other models are based on the depletion of a substrate also needed for activator growth. These models have been evaluated on regular grids and amorphous topologies, and have been compared with a state-of-the-art non-chemical cluster head election algorithm.

Our results show that there is a trade-off between the stable but potentially slow response to perturbations in the Activator–Inhibitor model at one extreme, and at the other extreme, the fast response to perturbations at the cost of more frequent failures in the Gray–Scott model. In the middle, the Activator–Substrate model exhibits slowly "moving peaks" which can lead to larger downtimes. The first model seems more appropriate in situations where, once the pattern is formed, it is not supposed to change very often. In contrast, the latter two models seem useful in more dynamic situations where frequent changes are expected, and the system should constantly react to them by quickly reconfiguring itself to a new valid configuration. Depending on the extent of dynamicity, the Gray–Scott model or the Activator–Substrate model may be preferred: the former leads to a higher availability, which is a very desirable property, while the latter has a lower failure rate.

The numeric simulations of all the three models exhibit similar qualitative behavior on regular grids or on amorphous topologies. Moreover under comparable parameter configurations, they provide robustness figures similar to those provided by Basagni's state-of-the-art non-chemical scheme (Basagni 1999). Still, one must be aware of the high computational cost involved in reaction–diffusion simulation using continuous concentration values such as those presented here. When precise details of concentration profiles are not needed, simplifications such as Gaussians (Shen et al. 2004) or Cellular Automata (Deutsch and Dormann 2005) might be more appropriate. However, they might also be more prone to perturbations: for instance, one wrong bit flip on a binary cellular automaton often causes the whole pattern to be disrupted, while perturbations to concentration values have comparatively little impact on a continuous system.

One might ask whether our results could be mere artifacts of our particular chemical implementations of the models covered. There might be other solutions leading to sets of reactions that do not suffer from the catalyst depletion problem found in our chemical solution for the Gierer–Meinhardt model. And there might be other sets of chemical reactions able to implement the Activator–Substrate model, or other parameter ranges where the peaks are more static. Or other interesting Gray–Scott configurations with lower failure rates. It is not obvious how to generalize the evaluation of given reaction–diffusion systems to all possible parameter settings, given that only a few parameter combinations lead to interesting patterns. Nevertheless, given the extensive evaluation provided, we believe that our results are fairly representative of the general characteristics of the models studied under valid parameter ranges for the desired cluster head computation task.

As future work, a microscopic simulation of the system would be interesting, taking into account stochastic effects that have not been considered here. Other failure modes should also be included, such as computation or communication failures, movement or removal of processors, insertion of parasite reactions, and so on. Moreover, it would be useful to evaluate other reaction–diffusion approaches and their robustness when used as engineering tools. It would also be interesting to study their behavior and performance in combination with other morphogenetic mechanisms. Another interesting topic for future work is to investigate the automatic evolution of reaction networks leading to the desired patterns.

Acknowledgements This work was supported by the European Union through the BIONETS Project EU-IST-FET-SAC-FP6-027748, www.bionets.eu, and by the French Region Alsace through the EVOL grant. It started when the first author was with the University of Basel in Switzerland.

References

Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Knight, T., Nagpal, R., Rauch, E., Sussman, G., & Weiss, R. (2000). Amorphous computing. *Communications of the ACM*, *43*(5), 74–82.

Adamatzky, A., de Lacy Costello, B., Melhuish, C., & Ratcliffe, N. (2003). Experimental reaction–diffusion chemical processors for robot path planning. *Journal of Intelligent & Robotic Systems*, *37*(3), 233–249.

Adamatzky, A., de Lacy Costello, B., & Asai, T. (2005). *Reaction–diffusion computers*. Elsevier Science, New York.

Arena, P., Fortuna, L., & Branciforte, M. (1999). Reaction–diffusion CNN algorithms to generate and control artificial locomotion. *IEEE Transactions on Circuits and Systems. I, Fundamental Theory and Applications*, *46*(2), 253–260.

Atkins, P., & de Paula, J. (2002). *Physical chemistry*. Oxford: Oxford University Press.

Bandini, S., Mauri, G., Pavesi, G., & Simone, C. (2005). Computing with a distributed reaction–diffusion model. In *Lecture notes in computer science: Vol. 3354. Machines, computations, and universality* (pp. 93–103). Berlin: Springer.

Bar-Yam, Y. (2003). *Dynamics of complex systems*. Reading: Westview Press.

Basagni, S. (1999). Distributed clustering for ad hoc networks. In A. Y. Zomaya, D. F. Hsu, O. Ibarra, S. Origuchi, D. Nassimi, & M. Palis (Eds.), *Proc. of I-SPAN* (pp. 310–315). Washington: IEEE Computer Society.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: From natural to artificial systems*. New York: Oxford University Press.

Coore, D., & Nagpal, R. (1998). Implementing reaction–diffusion on an amorphous computer. In *Proc. MIT student workshop on high-performance computing in science and engineering*. Boston: MIT Laboratory for Computer Science. Technical Report 737.

Dale, K., & Husbands, P. (2010). The evolution of reaction–diffusion controllers for minimally cognitive agents. *Artificial Life*, *16*(1), 1–20.

Deckard, A., & Sauro, H. M. (2004). Preliminary studies on the in silico evolution of biochemical networks. *ChemBioChem*, *5*(10), 1423–1431.

Deutsch, A., & Dormann, S. (2005). *Cellular automaton modeling of biological pattern formation: characterization, applications, and analysis*. Basel: Birkhauser.

Dittrich, P. (2005). Chemical computing. In *Lecture notes in computer science: Vol. 3566. Unconventional programming paradigms (UPP 2004)* (pp. 19–32). Berlin: Springer.

Dittrich, P., Ziegler, J., & Banzhaf, W. (2001). Artificial chemistries—a review. *Artificial Life*, *7*(3), 225–275.

Dormann, S. (2000). *Pattern formation in cellular automaton models*. PhD thesis, University of Osnabrück, Austria, Dept. of Mathematics/Computer Science.

Doursat, R. (2008). Organically grown architectures: creating decentralized, autonomous systems by embryomorphic engineering. In *Organic computing* (pp. 167–200). Berlin: Springer. Chap. 8.

Durvy, M., & Thiran, P. (2005). Reaction–diffusion based transmission patterns for ad hoc networks. In *Proc. of IEEE INFOCOM* (pp. 2195–2205). Washington: IEEE.

Erciyes, K., Dagdeviren, O., Cokuslu, D., & Ozsoyeller, D. (2007). Graph theoretic clustering algorithms in mobile ad hoc networks and wireless sensor networks. *Applied and Computational Mathematics*, *6*(2), 162–180.

Fatès, N. (2010). Solving the decentralised gathering problem with a reaction–diffusion-chemotaxis scheme. *Swarm Intelligence*, *4*, 91–115.

Ferrández, J. M., Lorente, V., Cuadra, J. M., de la Paz, F., Álvarez Sánchez, J. R., & Fernández, E. (2010). A hybrid robotic control system using neuroblastoma cultures. In *Lecture notes in computer science: Vol. 6076. Hybrid artificial intelligence systems* (pp. 245–253). Berlin: Springer.

Gray, P., & Scott, S. (1990). *Chemical oscillations and instabilities: nonlinear chemical kinetics*. Oxford: Oxford Science.

Grzybowski, B. A., Bishop, K. J. M., Campbell, C. J., Fialkowski, M., & Smoukov, S. K. (2005). Micro- and nanotechnology via reaction–diffusion. *Soft Matter*, *1*, 114–128.

Henderson, T. C., Venkataraman, R., & Choikim, G. (2004). Reaction–diffusion patterns in smart sensor networks. In *Proc. of IEEE international conference on robotics and automation* (Vol. 1, pp. 654–658). Washington: IEEE.

Hyodo, K., Wakamiya, N., & Murata, M. (2007). Reaction–diffusion based autonomous control of camera sensor networks. In *Proc. 2nd international conference on bio-inspired models of network, information, and computing systems (bionetics)*. Gent: ICST.

Koch, A. J., & Meinhardt, H. (1994). Biological pattern formation: from basic mechanisms to complex structures. *Reviews of Modern Physics*, *66*(4), 1481–1508.

Lin, C., & Gerla, M. (1997). Adaptive clustering for mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, *15*(7), 1265–1275.

Lowe, D., Miorandi, D., & Gomez, K. (2009). Activation-inhibition-based data highways for wireless sensor networks. In *Proc. 4th international conference on bio-inspired models of network, information, and computing systems (bionetics)*. Gent: ICST.

Mazin, W., Rasmussen, KE, Mosekilde, E., Borckmans, P., & Dewel, G. (1996). Pattern formation in the bistable Gray–Scott model. *Mathematics and Computers in Simulation*, *40*, 371–396.

Meinhardt, H. (1982). *Models of biological pattern formation*. London: Academic Press.

Meyer, T., & Tschudin, C. (2009). Chemical networking protocols. In *Proc. 8th ACM workshop on hot topics in networks (HotNets-VIII)* (online).

Molnár, F. Jr., Izsák, F., Mészáros, R., & Lagzi, I. (2010). Simulation of reaction–diffusion processes in three dimensions using CUDA. arXiv 1004.0480.

Murray, J. D. (2003). *Mathematical biology: spatial models and biomedical applications* (Vol. 2). Berlin: Springer.

Neglia, G., & Reina, G. (2007). Evaluating activator–inhibitor mechanisms for sensors coordination. In *Proc. 2nd international conference on bio-inspired models of network, information, and computing systems (bionetics)*. Gent: ICST.

Pearson, J. E. (1993). Complex patterns in a simple system. *Science*, *261*(5118), 189–192.

Pfeifer, R., Iida, F., & Bongard, J. (2005). New robotics: design principles for intelligent systems. *Special Number of Artificial Life on New Robotics, Evolution and Embodied Cognition*, *11*(1–2), 99–120.

Rauch, E. (2003). Discrete, amorphous physical models. *International Journal of Theoretical Physics*, *42*(2), 329–348.

Rubenstein, M., Sai, Y., Choung, C. M., & Shen, W. M. (2009). Regenerative patterning in swarm robots: mutual benefits of research in robotics and stem cell biology. *The International Journal of Developmental Biology*, *53*, 869–881.

Sanderson, A. R., Meyer, M. D., Kirby, R. M., & Johnson, C. R. (2009). A framework for exploring numerical solutions of advection–reaction–diffusion equations using a GPU-based approach. *Computing and Visualization in Science*, *12*(4), 155–170.

Shen, W. M., Will, P., Galstyan, A., & Chuong, C. M. (2004). Hormone-inspired self-organization and distributed control of robotic swarms. *Autonomous Robots*, *17*(1), 93–105.

Soro, S., & Heinzelman, W. B. (2009). Cluster head election techniques for coverage preservation in wireless sensor networks. *Ad Hoc Networks*, *7*(5), 955–972.

Stepney, S. (2010, in press). Nonclassical computation: a dynamical systems perspective. In *Handbook of natural computing* (Vol. II). Berlin: Springer. Chap. 52.

Tsuda, S., Zauner, K. P., & Gunji, Y. P. (2007). Robot control with biological cells. *Biosystems*, *87*, 215–223.

Turing, AM (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, *327*, 37–72.

Yamamoto, L., & Miorandi, D. (2010). Evaluating the robustness of activator–inhibitor models for cluster head computation. In *Lecture notes in computer science: Vol. 6234. Proc. ANTS, special session on morphogenetic engineering* (pp. 143–154). Berlin: Springer.

Yoshida, A., Aoki, K., & Araki, S. (2005). Cooperative control based on reaction–diffusion equation for surveillance system. In *Lecture notes in computer science: Vol. 3683. Knowledge-based intelligent information and engineering systems* (pp. 533–539). Berlin: Springer.

Yoshida, A., Yamaguchi, T., Wakamiya, N., & Murata, M. (2008). Proposal of a reaction–diffusion based congestion control method for wireless mesh networks. In *Proc. 10th international conference on advanced communication technology (ICACT)* (pp. 455–460). Washington: IEEE.

Yu, J. Y., & Chong, P. H. J. (2005). A survey of clustering schemes for mobile ad hoc networks. *IEEE Communications Surveys and Tutorials*, *7*(1), 32–48.

Ziegler, J., & Banzhaf, W. (2001). Evolving control metabolisms for a robot. *Artificial Life*, *7*(2), 171–190.