Evolutionary Graph Models with Dynamic Topologies on the Ubichip

Juan Camilo Peña[1], Jorge Peña[2], and Andres Upegui[3]

[1] UPB, Medellín, Colombia
juan.penasuarez@gmail.com
[2] Institut de Mathématiques Appliquées, UNIL, Lausanne, Switzerland
jorge.pena@unil.ch
[3] HES-SO/HEIG-VD, Yverdon, Switzerland
andres.upegui@heig-vd.ch

Abstract. The ubichip is a reconfigurable digital circuit with special reconfiguration mechanisms, such as dynamic routing and self-replication, for supporting the implementation of bio-inspired hardware systems. The dynamic routing mechanism allows to create and destroy interconnections between remote units in a distributed fashion, thus proving useful for implementing cellular systems featuring dynamic topologies. Evolutionary graph theory investigates genetic and cultural evolution processes using the mathematical formalism of both evolutionary game and graph theory. Populations are embedded in graphs representing interaction and imitation links. Payoffs are assigned and successful individuals are imitated with high probability. This paper describes the hardware implementation of a general evolutionary graph model in which the imitation network changes over time by exploiting the dynamic routing capabilities of the ubichip. As a particular example, we analyze the case of a coordination game played by agents arranged in a cycle in which imitation links are randomly created so as to simulate dynamic small-world networks.

Keywords: Reconfigurable circuit, dynamic routing, dynamic topology, evolutionary game theory, graph theory.

1 Introduction

The work presented in this paper has been developed in the framework of the european project Perplexus [1]. Perplexus aims to develop a scalable hardware platform made of custom reconfigurable devices endowed with bio-inspired capabilities that will enable the simulation of large-scale complex systems and the study of emergent behaviors in a virtually unbounded wireless network of computing modules. At the heart of these computing modules there is the ubichip [2], a custom reconfigurable electronic device featuring dynamic routing, distributed self-reconfiguration and a simplified connectivity. The ubichip thus offers an interesting set of mechanisms for supporting networks featuring different types of plasticity, like adding, removing, and replacing nodes and links in a directed

G.S. Hornby et al. (Eds.): ICES 2008, LNCS 5216, pp. 59–70, 2008.

graph. This dynamic directed graph can be a synaptogenetic neural network, like the work presented in [3], or an evolutionary graph model in which the imitation network changes over time like the work presented in this paper.

The main scientific objective of the Perplexus project is to ease the simulation large-scale complex systems. One of such complex systems are societies as loci of culture dissemination or *cultural evolution*, i.e. the general dynamic process resulting from the changes in cultural traits (beliefs, behaviors, ideas) over time in a human population. The population macro-phenomena to be explained can be, for instance, the emergence of globally shared norms, a social contract, cooperative behavior, or opinion convergence or polarization. The micro-processes generating these macro-processes are inter-personal social influence and imitation.

Evolutionary game theory [4] is the evolutionary extension of classical game theory. Originally conceived as an application of the game theory formalism to model genetic evolution [5], it has also been used to model how cultural traits spread in a population, i.e. how culture evolves [6]. In cultural evolutionary game models, individuals are characterized by a cultural trait ruling behavior, which we call *strategy*. The *payoff* or social success of an individual is conditioned by the strategy not only it but also other individuals in the population follow. In the game theory formalism, this frequency-dependent success is modeled according by means of a *payoff matrix*. Particular orderings of the payoffs determine different *games*, and games are thus simple metaphors of social interaction. Evolution in these models stem from the aggregation of micro-level processes of selective imitation. In the most common of these cultural evolution models, individuals have a tendency to imitate more successful individuals, or those having obtained a large payoff.

Evolutionary game models generally assume large, well-mixed populations in which everybody interacts with everybody and anyone can be imitated. Evolutionary graph theory [7] models more realistic population structures by embedding the population in graphs representing social networks. A given individual interacts with and imitates only its immediate neighbors, and not any member of the whole population. The sole fact of restricting interaction and imitation to neighbors introduces important changes in the dynamics of the cultural evolutionary process. For instance, in the famous Prisoner's Dilemma game, cooperators are doomed to extinction in large and well-mixed populations, but are able to survive in clusters when populations have some kind of topological structure [8].

When no particular population topology is assumed, evolutionary models often reduce to difference or differential equations that can be analytically solved. However, when spatial or network structure is taken into account, the resulting model becomes analytically intractable and can only be studied by means of agent-based or complex-system simulations. One of the main problems faced by complex-system modeling is the curse of dimensionality associated to sensitivity analysis, i.e. how sensitive the predictions of a model are to variations in its parameters. Sensitivity analysis of agent-based models often imply running a large number of simulations, varying parameters in all combinations. Frequently, the total number of required simulations for covering the parameter space can be

simply unfeasible for traditional computing systems. It is our claim that one way of alleviating this problem is to exploit the explicit parallelism of agent-based models, and to implement them partially or completely in dedicated hardware.

This work explores the ubichip implementation of an evolutionary graph model with a dynamic imitation network. The paper is organized as follows. Section 2 introduces the ubichip. Section 3 defines the evolutionary graph model with dynamic topology we are interested in, while section 4 describes its hardware implementation on the ubichip. Section 5 presents the experimental setup and results. Finally, we draw some conclusions in section 6.

2 Ubichip

The ubichip is a custom reconfigurable electronic device capable of implementing bio-inspired mechanisms such as growth, learning, and evolution [2]. These bio-inspired mechanisms are possible thanks to reconfigurability mechanisms like dynamic routing, distributed self-reconfiguration, and a simplified connectivity.

The ubichip is mainly composed of three reconfigurable layers. The first layer consists of an array of ubicells, the reconfigurable logic elements used for computation purposes. A *ubicell* is composed of four 4-input look-up tables (LUT) and four flip-flops (DFFs). These ubicells can be configured in different modes such as counters, FSMs, shift-registers, 4 independent registered and combinatorial LUTs, adders, subtractors, etc. An ubicell can also implement a simple 4-bit processing element of a single instruction, multiple data (SIMD) processing platform, and several ubicells can be merged to create a higher resolution processor. In this last mode, an on-chip centralized sequencer is responsible for decoding the instructions for the multi-processor management. A particular configuration mode of a ubicell, very important for the work presented in this paper, is the 64-bit LFSR mode, which permits using the 64 configuration bits of the four 4-input LUTs as a good quality pseudo-random number generator. This feature allows us to implement processes such as probabilistic functions and pseudo-random event triggers at a very low cost of reconfigurable resources.

The second layer is made of self-replicating units that allow parts of the circuit to self-replicate somewhere else on the chip, independently of any external control. This truly new feature can be very useful for heterogeneous cellular systems with dynamic topologies. A node in a graph can, for instance, decide to self-replicate or to self-destroy according to an underlying algorithm, so as to populate the complete chip. This mechanism is not exploited by the work presented in this paper, but more details of it can be found in [9].

Finally, the third layer contains dynamic routing units connected to their eight neighbors that permit the ubicells to dynamically connect to any part of the circuit. Cellular systems with dynamic topologies, requiring the ability of creating and destroying paths at runtime, can greatly benefit from this feature. The dynamic routing mechanism implemented in the routing units, along with the computational capabilities offered by the ubicells, allow us to tackle the modeling of evolutionary graph systems exhibiting dynamic topologies.

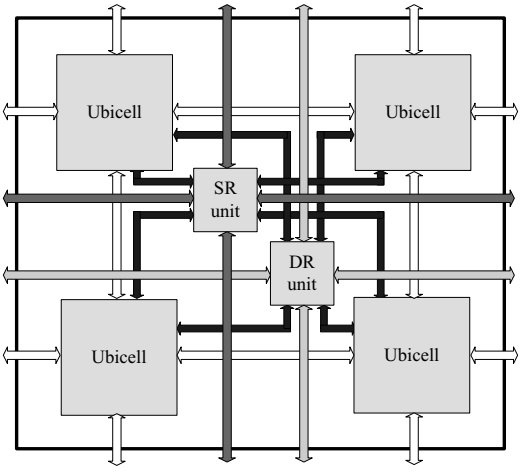


Fig. 1. Macrocell architecture

Based on identifiers and a concept of sources and targets trying to reach a correspondent with the same ID, this mechanism looks quite similar to the system described in [10], while having enhancements on different aspects. The ID, being stored in a routing unit, can be dynamically modified by an ubicell connected to it. The basic idea of the routing algorithm is to construct paths between sources and targets by dynamically configuring multiplexers, and by letting data follow the same path for each pair of source and target. Sources and targets can decide to connect to their corresponding unit at any time by launching a routing process. These routing processes are triggered by the ubicells, so a node in a graph, for instance, can decide to create a new connection with another node. During a routing process, after the identification of the sources and targets based on their IDs, a phase of path creation executes a breadth-first search distributed algorithm, looking for the shortest path. If such a path exists, the multiplexers are configured accordingly. If not, the ubicell is noticed about the failure of the routing process. Once a source and a target have been connected, the path is fixed and data can be directly sent at any time, until the path is destroyed.

The management of the dynamic routing is performed via control signals from the ubicells to the routing unit. These signals create and destroy connections, and modify the routing unit ID. In the opposite direction we have a couple of flags that allow the routing unit to inform the system implemented on the ubicells whether or not a connection is currently being created (*is_connecting*) and whether or not a specific routing unit is connected (*is_connected*).

The three layers of the ubichip are interconnected so as to allow the ubicell layer to control the two top layers. Units from the three different layers are grouped together and constitute what we call a *macrocell*. A macrocell contains four ubicells connected to a routing unit and a self-reconfiguration unit as

depicted in Fig. 1. The ubicell layer can thus implement a circuit able to control the dynamic routing and the self-replication layers.

3 Evolutionary Graph Model with Dynamic Topology

We are particularly interested in evolutionary graph models in which, as recently proposed by Ohtsuki et al. [11], the connectivity between individuals is given by two graphs: the *interaction graph* H and the *imitation graph*[1] G. Edges of H determine who interacts with whom and edges of G determine who imitates whom. Each agent in the population is labeled by an index i, also referring to the node it occupies in both graphs. Each agent is also characterized by its *strategy* s_i, which can take one of two values: A or B. The set of neighbors of i in H (resp. in G) is denoted by $N_H(i)$ (resp. $N_G(i)$). We define the neighborhood of i as the set of nodes having a directed edge with i as the final vertex.

Depending on its current strategy and the strategies of the agents in its neighborhood $N_H(i)$, agent i gets an *accumulated payoff* given by:

$$W_i(t) = \sum_{j \in N_H(i)} V\left(s_i(t), s_j(t)\right),$$

where V is the *payoff matrix* of a simple two-person symmetric game:

	A	B
A	a	b
B	c	d

After interaction, agents in our model revise their strategies via an imitation rule that takes into account the distribution of payoffs and the proportion of behaviors in their imitation neighborhoods $N_G(i)$. We use the following general rule: the focal agent i first draws an agent j at random from its neighborhood and then decides to copy j's strategy depending on the difference between payoffs $W_j(t) - W_i(t)$. We consider two alternatives. The *deterministic* update makes agent i copy j if $W_j(t) - W_i(t)$ and keep its old strategy otherwise. The *probabilistic* update makes agent i imitate j with a probability given by $f\left(W_j(t) - W_i(t)\right) > 0$, where f is any suitable monotonic increasing function. It must be noticed here that, even when using the deterministic update rule, imitation is essentially a stochastic process since the agent to be imitated is chosen randomly.

Depending on the initial distribution of strategies, the structure of the interaction and imitation graphs, the entries of the payoff matrix and the imitation rule used, the system will eventually evolve towards one of its two absorbing states: all-A (all individuals follow strategy A) and all-B (all individuals follow strategy B). The aim of evolutionary graph theory is examining what conditions lead to the fixation of one of two competing strategies.

[1] Originally called *replacement* graph in [11]. We call it imitation graph since we are interested in cultural rather than genetic evolutionary processes.

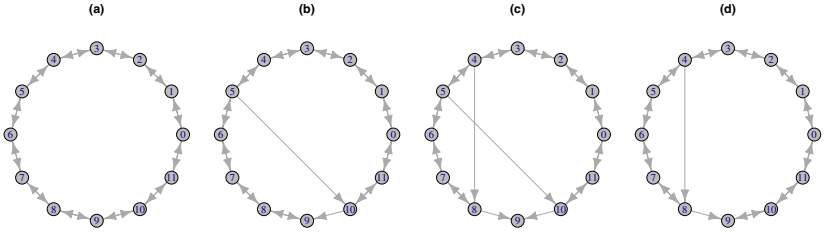


Fig. 2. Example of the dynamics of an imitation graph for a population of 10 agents. The imitation graph starts as a simple cycle (a). A first dynamic connection is created: node 5 is connected to node 10, node 10 is disconnected from neighboring node 9 (b). A second dynamic connection is created: node 4 is connected to node 8, node 8 is disconnected from neighboring node 9 (c). One dynamic connection is killed (d).

In this paper, we explore one particular instance of this general model in which the interaction graph is a static cycle and the imitation graph is a directed, dynamic small-world network constructed over this cycle. With this setup, we try to model the fact that individuals often compare their success with and imitate others with whom they do not interact, and that the resulting imitation social network is in general more dynamic than the interaction graph (think of the effect of long-range communication in contemporary societies).

The dynamics of the considered imitation graph are exemplified in Fig. 2. The imitation graph starts as a cycle perfectly coinciding with the interaction graph, so that each node is connected to its left and right neighbors. Each node has a fixed probability p_c of connecting to a random node. When this happens, the target node is disconnected from one of their immediate neighbors, to assure that the size of the imitation neighborhoods of all nodes remains equal to 2. This arbitrary constraint was added in order to to simplify the hardware implementation of the model, to be explained in Section 4. Finally, all dynamic connections have a fixed probability p_k of being deleted. It can be seen that graphs with $p_c/p_k >> 1$ will grow in disorder whereas graphs with $p_c/p_k << 1$ will practically be cycles with some small-world connections being sporadically created.

4 Evolutionary Graph Model in the Ubichip

The dynamic routing capabilities of the ubichip offer an interesting connectivity mechanism for implementing the dynamic topology evolutionary graph model described in the previous section. In this section we propose an architecture for such a model and describe its implementation on the ubichip.

4.1 Evolutionary Graph Model Architecture

We consider two hardware implementations: the first one for an agent with a deterministic imitation rule and the second one for an agent with a probabilistic update rule. The hardware implementation of the first agent is illustrated

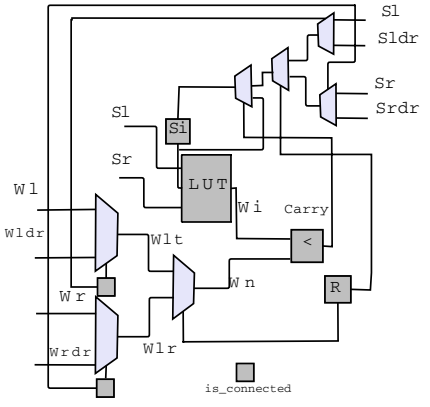


Fig. 3. Hardware architecture of the deterministic agent

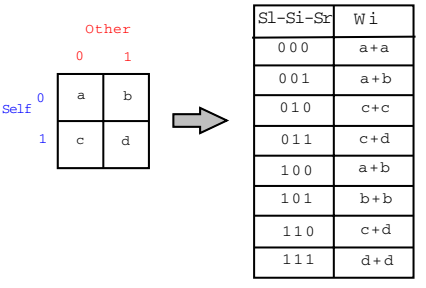


Fig. 4. LUT implementation of a payoff matrix

in Fig. 3. In this figure we can identify two different types of inputs, one for the interaction graph and the other for the imitation graph. For the interaction graph, the inputs s_l ("strategy left") and s_r ("strategy right") provide the strategies of neighbor agents. These two inputs, along with the current agent's strategy s_i stored in a register, allow the computation of the agent's payoff with a payoff matrix implemented in the LUT. Strategy A is coded as a logical 0, and strategy B as a logical 1. Figure 4 depicts how to implement such matrix in a LUT. This implementation considers a 3-input and n-output LUT, where n is defined by the desired resolution of the matrix entries. In the specific case of our implementation in the ubichip we set $n = 3$, i.e. each entry of the payoff matrix can consist of one out of eight different possible values.

Unlike the interaction graph, the topology of the imitation graph changes over time. We exploit the dynamic routing capabilities of the ubichip to make imitation edges modifiable and accessible from the dynamic routing units. In order to do this, the architecture of Fig. 3 has the inputs W_l and W_r for the payoffs of neighboring agents (i.e. the left and the right neighbors in the interaction graph), and the inputs W_l^{dr} and W_r^{dr} for payoffs of the remote agents dynamically connected through the dynamic routing mechanism. A multiplexer selects whether the agent to imitate is the same neighbor in the interaction graph or a remote agent. This multiplexer is controlled by a flag driven by the dynamic routing unit (*is_connected*), that indicates whether or not the dynamic routing unit is connected to a remote unit. A second multiplexer randomly selects the agent to imitate: either the agent connected to the left or the agent connected to the right. Finally, a comparator is used to decide whether or not the imitation is done by selecting the value to load in the strategy register (s_i).

The hardware implementation of an agent with a probabilistic imitation rule is shown in Fig. 5. It keeps the same inputs and base structure as the deterministic agent. The only difference is the part of the circuit that decides whether

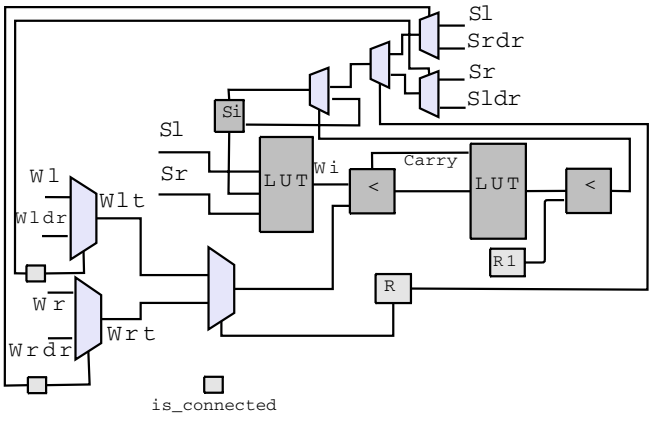


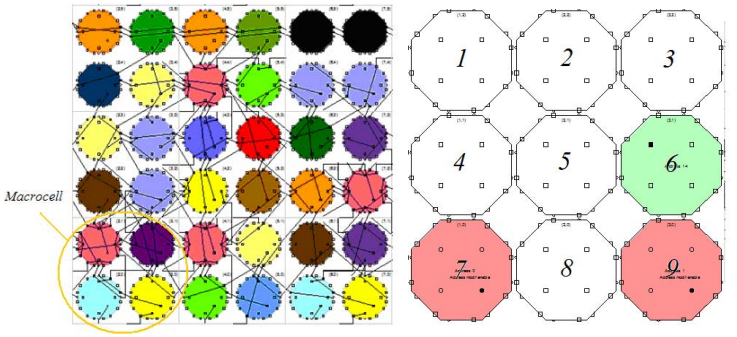
Fig. 5. Probabilistic agent architecture

imitation is performed or not. The decision to imitate is taken by means of a second LUT that computes the imitation probability given the difference between the payoff of the self (W_i) and the payoff of the neighbor (W_n), and a comparator that compares this probability with a 4-bit pseudo-random number. The implementation is flexible, allowing to parameterize the probabilistic function and thus to implement different probabilistic imitation rules.

4.2 Ubichip Implementation

We implemented both architectures on the ubichip using the UbiManager design tool [12]. The UbiManager is a graphical user interface tool that permits editing ubichip designs in a user-friendly manner. It provides independent views of the three configuration layers of the ubichip introduced in section 2.

Although architectures with deterministic as well as probabilistic agents have been implemented, the results shown in the next section consider only deterministic agents. The implementation of a single agent with a deterministic imitation rule requires 9 macrocells. Fig. 6 shows the UbiManager view of the two ubichip layers used in the agent implementation: the ubicell and the dynamic routing layers. Fig. 6(a) shows the ubicells and the static routing connectivity. The ubicell array is used for implementing the full agent's architecture and the static interaction graph. The imitation graph uses both static and dynamic routing resources. Links with neighboring agents use static and links with remote agents use the dynamic routing mechanism. Figure 6(b) shows the dynamic routing units configured as sources and targets. Routing unit 6 is the only one acting as source: it can send the agent's payoff and strategy to any other agent, allowing the remote agent to imitate it. Routing units 7 and 9 are configured as targets: they can receive the payoff and the strategy data sent from the sources of remote agents. Agents can thus create, destroy and modify imitation links in a completely autonomous way.



(a) Ubicell layer view (b) Dynamic routing layer view

Fig. 6. UbiManager view of the agent implementation

The process of dynamically creating a routing link is decomposed in two parts: one carried out on the source and the other on the target. Each dynamic routing unit configured as source is initialized with a unique ID. Its ID will not change during the evolution of the graph. On the other hand, dynamic routing units configured as targets are initialized with an arbitrary ID, which is further modified in a random way every clock cycle. Simultaneously, these units will constantly attempt to connect to a source with a certain probability p_c. The computation of this probability is implemented on two ubicells: one acting as a pseudo-random number generator and the second as a comparator with a constant. The connection attempt is successful if there is no other dynamic routing process in progress and if the target manages to find a source with the same ID. In this case, the link is created and the corresponding target does not attempt to connect to another source. The full process of link creation can take from 10 to about 30 clock cycles, depending on the physical distance between units.

The process of link destruction uses the same principle as the process of link creation. Targets are constantly attempting to destroy existing remote links with a certain probability p_k. This probability is typically much lower than the probability of attempting a connection because link destruction only needs a single clock cycle to be completed. If the probability p_k were lower, the effect of having a dynamic topology would be negligible because of the very low number of existing links. It must be also noted that once a link is destroyed the target can retry to connect with other sources.

5 Experimental Setup and Results

As a particular example of the type of experiments we can build and simulate with our system, we set up the previously described evolutionary graph model with a population of 64 agents, playing the coordination game given by the following payoff matrix:

	A	B
A	1	0
B	0	3

Notice that the average payoff of a population converging in all-A is 1 while it is 3 for a population converging in all-B. This makes all-B *payoff dominant*, which means that the population is better off in the all-B state than in the all-A state.

Agents were implemented using the simple deterministic imitation rule. The imitation network was programmed to have a probability of attempting a dynamic connection belonging to the set $p_c = \{0, 1/256, 9/256, 1/4, 1/2\}$. The first case ($p_c = 0$) recovers the limiting case in which the imitation network is static and perfectly coincident with the interaction network. For all the dynamic cases (i.e. $p_c > 0$), the probability of killing a connection was set to $p_k = 1/256$. The initial proportion p_B^0 of agents having the payoff dominant strategy B was set to different values in the set $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. Each experiment (couple of values p_c, p_B^0) was replicated 20 times.

Fig. 7 shows the results we obtained in terms of the probability of converging to all-B and the mean time (simulation steps) to convergence in one of the two absorbing states (all-A and all-B) as functions of p_B^0. Different degrees of dynamism or disorder of the imitation graphs (measured by p_c) are shown with different line types. From the first plot we can see that, whatever the value of p_c, curves represent monotonically increasing functions, which means that a high initial proportion of strategy B in the population correlates with a high probability of having all the population converging to the all-B state. Curves are also ordered such that, for the same p_B^0, the probability of converging to

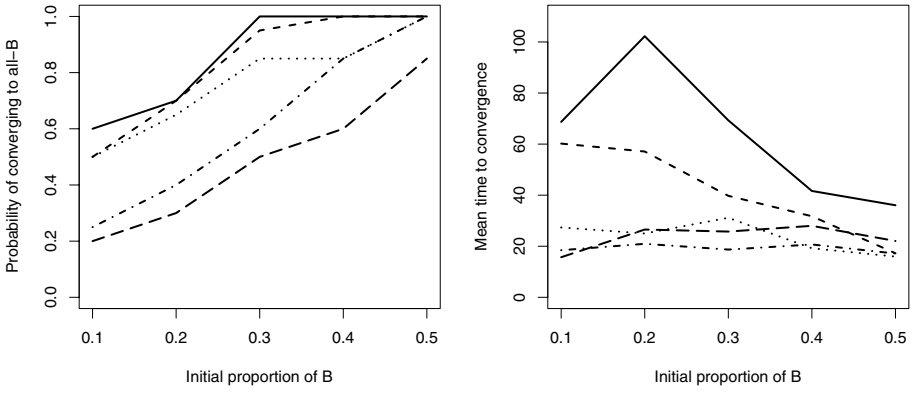


Fig. 7. Probability of converging to all-B (*left*) and mean time to convergence (*right*) as functions of the initial proportion of B for $p_c = 0$ (*solid*), $p_c = 1/256$ (*dashed*), $p_c = 9/256$ (*dotted*), $p_c = 1/4$ (*dotdash*) and $p_c = 1/2$ (*longdash*)

all-B is always higher for lower p_c. This indicates that more dynamic, disordered imitation networks are less favorable to converging in the payoff dominant equilibrium. Regarding the mean time to convergence, it can be seen that more dynamic, disordered imitation graphs favor a faster convergence in an absorbing state. Depending on the initial proportions of agents holding strategy A or B, the time to convergence can be, in average, from 2 to almost 5 times faster for an imitation graph with $p_c > 1/256$ than for a perfectly static cycle. Thus, more communication with non-neighbors favors faster consensus at the expense of a higher probability of convergence in the suboptimal norm.

6 Conclusions

This paper has presented the model and ubichip implementation of an evolutionary graph with dynamic topology. The described implementation uses the dynamic routing mechanisms of the ubichip in order to create and destroy graph links in a completely distributed and autonomous way. We focused on an imitation graph consisting in a ring topology with randomly selected links being dynamically replaced by connections to randomly selected nodes. However, the presented implementation permits to easily define any other arbitrary type of topology (regular or not) by properly setting the dynamic routing interconnections. Moreover, the 2-input nodes described here can be easily upgraded to nodes with a larger number of inputs through slight architecture modifications. These modifications consist in the addition of extra input multiplexers for increasing the inputs of the imitation graph nodes and the inclusion of a larger LUT (built with several 4-input LUTs) for enhancing the interaction graph.

The ubichip has shown to be well adapted for the implementation of such type of models because of two main aspects: 1) the dynamic routing for addressing dynamic topology issues and 2) the configuration modes of the ubicell that have been shown to be well adapted for coarse and fine grained functions. In particular, the 64-bit LFSR mode provides a very good building block for stochastic cellular systems.

Regarding the implemented model, the obtained results suggest that for a simple coordination game, having a dynamic small-world imitation graph reduces both the time for reaching consensus and the probability that the whole population will agree on the optimal norm.

Acknowledgments

This project is funded by the Future and Emerging Technologies programme IST-STREP of the European Community, under grant IST-034632 (PERPLEXUS). The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

References

1. Sanchez, E., Perez-Uribe, A., Upegui, A., Thoma, Y., Moreno, J., Villa, A., Volken, H., Napieralski, A., Sassatelli, G., Lavarec, E.: PERPLEXUS: Pervasive computing framework for modeling complex virtually-unbounded systems. In: Arslan, T., et al. (eds.) AHS 2007 - Proceedings of the 2nd NASA/ESA Conference on Adaptive Hardware and Systems, pp. 587–591. IEEE Computer Society Press, Los Alamitos (2007)
2. Upegui, A., Thoma, Y., Sanchez, E., Perez-Uribe, A., Moreno, J.M., Madrenas, J.: The Perplexus bio-inspired reconfigurable circuit. In: Arslan, T., et al. (eds.) Proceedings of the 2nd NASA/ESA Conference on Adaptive Hardware and Systems, pp. 600–605. IEEE Computer Society Press, Los Alamitos (2007)
3. Upegui, A., Thoma, Y., Perez-Uribe, A., Sanchez, E.: Dynamic routing on the ubichip: Toward synaptogenetic neural networks. In: AHS 2008 - Proceedings of the 3rd NASA/ESA Conference on Adaptive Hardware and Systems, pp. 228–235. IEEE Computer Society Press, Los Alamitos (2008)
4. Gintis, H.: Game theory evolving. Princeton University Press, Princeton (2000)
5. Maynard Smith, J.: Evolution and the Theory of Games. Cambridge Univ. Press, Cambridge (1982)
6. Boyd, R., Richerson, P.J.: Culture and the evolutionary process. University of Chicago Press, Chicago (1985)
7. Szabo, G., Fath, G.: Evolutionary games on graphs. Physics Reports 446(4-6), 97–216 (2007)
8. Nowak, M.A., May, R.M.: Evolutionary games and spatial chaos. Nature 359(6398), 826–829 (1992)
9. Thoma, Y., Upegui, A., Perez-Uribe, A., Sanchez, E.: Self-replication mechanism by means of self-reconfiguration. In: Platzner, M., Grosspietsch, K.E., Hochberger, C., Koch, A. (eds.) ARCS 2007. LNCS, vol. 4415, pp. 105–112. Springer, Heidelberg (2007)
10. Thoma, Y., Sanchez, E.: An adaptive FPGA and its distributed routing. In: Proc. ReCoSoc 2005 Reconfigurable Communication-centric SoC, Montpellier - France, June 2005, pp. 43–51 (2005)
11. Ohtsuki, H., Pacheco, J.M., Nowak, M.A.: Evolutionary graph theory: Breaking the symmetry between interaction and replacement. Journal of Theoretical Biology 246(4), 681–694 (2007)
12. Thoma, Y., Upegui, A.: Ubimanager: a software tool for managing ubichips. In: AHS 2008 - Proceedings of the 3rd NASA/ESA Conference on Adaptive Hardware and Systems, pp. 213–219. IEEE Computer Society Press, Los Alamitos (2008)