Idiotypic networks for evolutionary controllers in virtual creatures

Nicola Capodieci[1], Emma Hart[2] and Giacomo Cabri[1]

[1]University of Modena and Reggio Emilia
[2]Edinburgh Napier University

Abstract

We propose a novel method for evolving adaptive locomotive strategies for virtual limbless creatures that addresses both functional and non-functional requirements, respectively the ability to avoid obstacles and to minimise spent energy. We describe an approach inspired by artificial immune systems, based on a dual-layer idiotypic network that results in a completely decentralised controller. Starting from a system initialised with five non-adaptive locomotion strategies, we show that an adaptive controller can evolve that both minimises energy requirements and maximises distance covered when compared to the initial strategies.

Introduction

In Blumberg and Galyean (1995) a *virtual creature* is defined as an animate object, capable of goal-directed and time-varying behaviour, situated within a simulated environment with which it can interact. In this paper, we will focus on how to provide the means for a virtual creature to discover adaptive movement patterns; this will be accomplished by applying a previously introduced design methodology that merges artificial immune systems (AIS) and autonomic computing (Kephart and Chess (2003)). AIS takes inspiration from the biological immune system, in order to extract algorithms and methodologies for designing computational systems able to feature the same characteristics of the biological immune system, such as scalability, adaptivity, emerging cognition and decentralization. The evolutionary features able to provide adaptivity in the observed systems depends on which AIS related paradigm is used for solving a specified computational problem (De Castro and Timmis (2002)). In this instance, idiotypic networks as theorized by Cohen (2000a,b) have been used, due to their ability to show cognition. We are going to model our creature as a collection of independent (autonomic) units, with the aim of discovering new trajectories of movement and detect/avoid obstacles while optimizing the energy consumption of the virtual creature. This can be translated into a distributed autonomic computing related problem due to the composite morphologic nature of the creature itself and its requirement to Self-Adapt over time. Adaptation here is seen as the ability of the creature to combine known movement patterns with no adaptive ability in order to evolve new locomotion strategies able to keep the creature in motion. Although there is a wealth of literature related to the use of evolutionary methods for achieving control, our work differs in two key aspects. Firstly, it considers both functional and non-functional requirements of the creature, achieving movement while minimising energy. Secondly, our algorithm (that we name *SelfEx*) is completely distributed, in that every constituent unit of the virtual creature is able to evolve independently, sharing the minimum amount of information throughout the whole creature. Another difference compared to previous literature is the kind of network(s) involved: using an idiotypic network instead of a neural network (e.g. evolutionary morphologies as in Miconi and Channon (2005)) has the potential to further enrich the literature regarding evolutionary virtual creatures.

The paper is organized as follows: after presenting a review of related work, a description of the components of the virtual creature and its surrounding environment will be provided. The SelfEx approach is presented by detailing the modelling choices used. After the model is presented, all the details regarding the simulations and experiments performed and their related parameters will be given. The paper is concluded with discussions about the obtained results and related future research directions.

Related work

The vibrant field of Artificial life uses computer simulation to investigate evolution of behavioural and cognitive mechanisms in virtual creatures Sims (1994), potentially leading to advances in both biology (e.g: Palyanov et al. (2012)) and robotics (e.g: Černỳ and Kubalík (2013)). We restrict our review to work related to understanding the evolution of movement strategies that might ultimately be applied to the robotic field. A significant volume of work exists in the evolutionary computing literature, summarised by Prez-Moneo Surez and Rossi (2013) in relation to movement of limbless creatures. Typically, evolution evolves centralised

controllers in which performance is evaluated in terms of the evolved trajectory and ability to avoid obstacles but does not account for energy consumption of the movement, a relevant factor if the motion is to be transferred to real robots. Moreover, our approach (SelfEx, named after the Self-* property of a system to autonomously change its coordination pattern during run time execution of tasks (Cabri and Capodieci (2013)) uses a set of pre-coded movement strategies as baseline behaviours to evolve. This is in contrast with known evolutionary controllers in virtual creatures, since these latter ones rely on single atomic actions.
From the robotics perspective, a number of authors have advocated the use of immune-inspired control strategies, beginning with Ishiguro et al. (1995) (autonomous navigation for a single robot). This work was more recently extended by Whitbrook et al. (2010), in which the authors detailed how similar algorithms can be properly transferred in real robots. In Capodieci et al. (2013a), the authors proposed that ideas from autonomic computing could be combined with immune-inspiration to provided distributed control. An idiotypic network algorithm was proposed and applied to selecting movement strategies in swarm foraging task in Capodieci et al. (2013b). The model was formalised into a framework (Capodieci et al. (2014)) but only considered functional requirements.

The creature

The virtual creature used for our simulations is depicted in Figure 1 (left hand side). It is composed of 10 identical constituent units, each a perfect cube shape. Each unit is connected to its neighbouring unit(s) through a chain of universal joints, thus giving them complete freedom to rotate along the X axis, while rotations along the other axes are constrained by collisions with nearby units. The initial spacing between two units is set to one fifth of the length of the cube size. Trivially, the creature is simulated in a 3D environment in which the three axis have the orientation shown in Figure 1 (left hand side) and collision, friction and gravity forces are present. Each unit is completely independent — the only shared variable is an analogous to a biological clock that ensures synchronised adaptation of units. This is depicted in Figure 1 (right hand side) and is represented by a periodic square wave in which we can identify two distinctive phases, labelled as positive (P) and negative (N) phases. The use of a common shared clock is common in the field of virtual robotic creatures in attempting to imitate the oscillatory and periodic locomotion activity cycles of many existing animals (e.g. see Ijspeert (2008) for a survey on the existing methods for implementing Central Pattern Generators (CPGs)). The period of the clock can be adjusted according to T_{eval}.

Movements and related energy consumption

The functional objective of our creature is to keep moving and to discover new means of locomotion. In previous work, movement has been achieved by use of fixed functions, e.g. the *serpenoid* function as proposed by Hirose and Morishima (1990), applied at junctions on the body. However, this requires the use of an external and centralized controller. In the presented decentralised model, a generic notation for representing a movement pattern is to consider that during each clock phase, each unit independently applies a force from the centre of its mass with magnitude $M_g \leq 1$ with a direction described as a vector in which the unitary magnitude of the force can be distributed among the three axes: $(m_{gx}\mathbf{e_x} + m_{gy}\mathbf{e_y} + m_{gz}\mathbf{e_z})$, given $|m_{gx}| + |m_{gy}| + |m_{gz}| = M_g$ and where $\mathbf{e_x}, \mathbf{e_y}, \mathbf{e_z}$ are the canonical basis vectors in $\mathbf{R}^3$. The direction of each force is always relative to the orientation of the unit. Movement of the creature therefore results from physical interactions between moving units; not all units are required to apply a force (and consume energy) for movement to occur: push or pulling behaviour of a unit can result from a force applied to a neighbouring unit. The energy spent by a single unit during a single time interval is calculated by summing the magnitudes of the applied force in both the positive and negative phase of the clock, therefore a single unit, during a time interval can show an energy consumption in the range $[0, 2]$. The total energy consumed (related the whole creature as indicated with EnC_{tot}) during a time interval is trivially calculated by summing the energy consumption experienced by each constituent unit during each phase of the clock.

Movement patterns

A series of 5 initial pre-coded movement patterns are used as starting points for evolution. A *movement pattern* is a description of how the whole creature moves and it is obtained by indicating magnitude and direction of the forces to be applied to each of its constituent unit during each phase of the clock. In the initial pre-coded movements, all forces magnitude are 1 and they are applied to one of the possible directions among $[\pm X, \pm Y, \pm Z]$. These movement patterns are loosely based on how limbless creatures move in space and are summarized in Table 1. In that table, for each unit (identified by a number from 0 to 9), the direction in which the force is applied is shown for each of the five pre-coded movement patterns (from Type1 to Type5). Individual units iterate the application of such forces as described in Table 1 during every time interval. Each initial pattern has an associated energy consumption that is fixed through the duration of an experiment. Moreover, the initial patterns do not enable obstacle avoidance or adaptation to external perturbations that cause unexpected rotations of the unit(s) and thus provide a baseline for evaluating whether these behaviours can emerge.

In Figure 2 the movement pattern labelled as Type1 is rep-

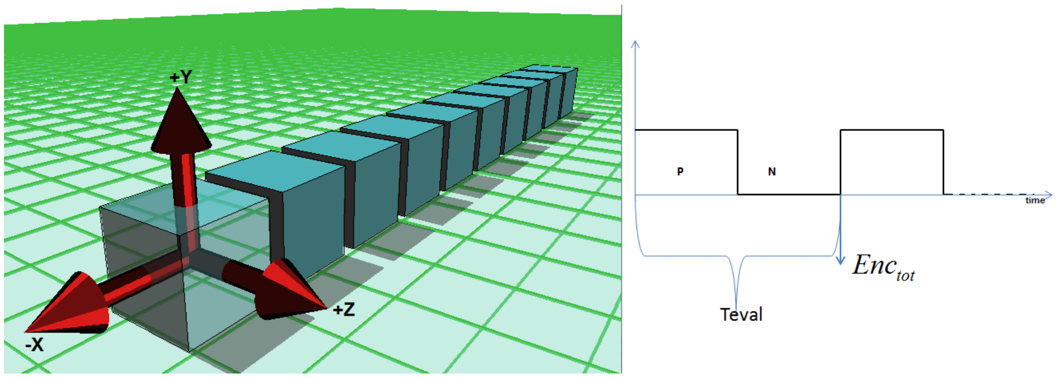

Figure 1: A representation of the virtual creature and its shared biological clock.

	Unit																				EnC_{tot}
	0		1		2		3		4		5		6		7		8		9		
	P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N	
Type1	+X	/	+X	/	+X	/	+Y	/	+Y	/	+Y	/	+Y	/	+Y	/	-X	+X	-X	+X	12
Type2	+X	+Y	+X	/	+X	/	+X	-Z	+Y	-Z	+Y	-Z	/	-Z	-X	/	-X	/	-X	+Y	15
Type3	+X	/	+X	/	+Y	/	+Y	/	+Y	/	+Y	/	/	-Z	/	-Z	/	-Z	/	-Z	10
Type4	+Y	+Z	+Y	+Z	+Y	+Z	+Y	+Z	/	/	/	/	+Z	/	+Z	/	+Z	/	+Z	/	12
Type5	-Z	/	-Z	/	-Z	/	-Z	/	-Z	/	-X	+Z	-X	+Z	-X	+Z	-X	+Z	-X	+Z	15

Table 1: Table of initial movements patterns, Type1-Type5, showing force applied to each unit at each clock phase (P,N). EnC_{tot} : total consumed energy during T_{eval}. Example: in the movement pattern Type5, unit 0 applies a force of 1 magnitude over the negative Z axes in the positive (P) phase; no forces applied during the negative (N) phase.

resented and resembles the locomotion strategies adopted by caterpillars. Other movement patterns are based on rolling and crawling, similar to snakes or worms[1]. It is important to stress that different movement patterns result in different outcomes both in terms of non-functional requirements (see the last column of Table 1, the total energy consumption EnC_{tot}) and in terms of functional requirements (calculated as the space distance the whole creature managed to travel during a time interval). Regarding the distance travelled by the creature, the performance varies according to the orientation of the constituent units, therefore any perturbation that causes the units to rotate from their original position can drastically change its performance and even prevent the creature to keep moving.

SelfEx: an idiotypic control model

Each constituent unit of the creature is seen as an autonomous component and has an associated *lymphnode* l_{c_i}. Each lymphnode is consisting of an *internal* network of interconnected *antibodies*. Additionally, an antibody in l_{c_i} may be connected by stimulatory or suppressive links to antibodies inside neighbouring lymphnodes $l_{c_{i+1}}$ and $l_{c_{i-1}}$, forming an *external network*. Antibody concentration varies over time according to the process outlined in Figure 3 and 4 and described in detail in the following sections. At the end of each time interval, the unit will select the highest concentrated antibody, so to extract from it a computational equivalent to an immunological response. Before describing the details of this process, it is important to show how the virtual antibodies are modelled.

[1]All movement patterns are simulated for reference in a video resource located at https://vimeo.com/89119516

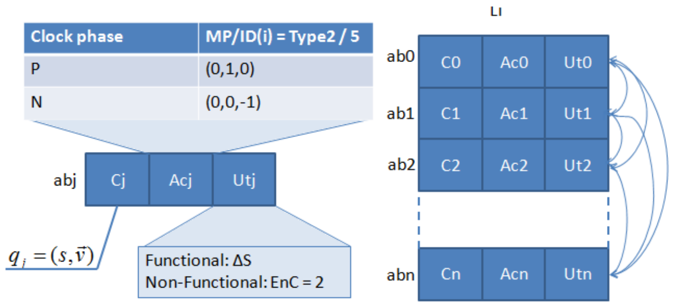

Figure 3: On the left side the model of a single antibody is shown. Example values are given to the action field (as verifiable from Table 1). The right part shows the connections among antibodies inside a single lymphnode.

Antibody Representation

An antibody is represented by a tuple $\langle condition\ C, action\ Ac, expected\ utility\ Ut\rangle$ and has an associated concentration (see Figure 3). This generic format was introduced in Capodieci et al. (2013b), and its implementation is application dependent. Here, the condition field is a quaternion in the form $(s, \overrightarrow{v})$ representing a possible orientation of the cube/unit in the three

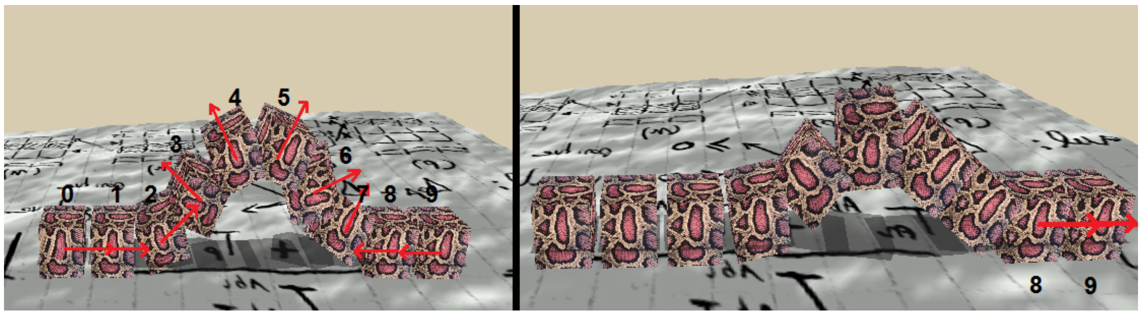

Figure 2: A graphical representation of the movement pattern “Type1”. Left part: positive phase P of the clock, Right Part: negative phase N of the clock. Identifiers of the single units are also indicated in this picture.

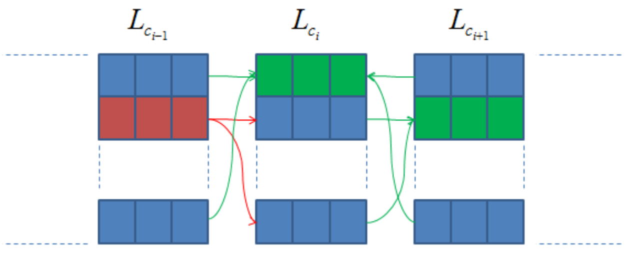

Figure 4: An example situation of how stimulation and suppression signals propagate in the external idiotypic network among three lymphnodes. Green antibodies resulted in positive feedback, while the red one obtained a negative feedback. Green arrows correspond to the increase of affinity by all the other unselected antibodies of neighbouring lymphnodes towards the green antibodies, while red arrows correspond to the increase of affinity by the red antibody towards all the other unselected antibodies.

possible directions of rotation. An *action* is a data-structure that represents a component identifier, the identifier of a movement pattern, and the magnitude and direction of the forces to be applied to the component at each clock phase. Finally, the *utility* field specifies two values corresponding to the expected value of the functional and non-functional measures. ΔS represents the functional utility and is simply the Euclidean distance between the starting position of the unit and its position at the end of a single time interval. The non-functional quantity is the energy consumption EnC (related to the single unit).

Initialisation and pre-experiments

Typically, initial antibodies populations in both AIS and evolutionary algorithms are created randomly. Instead, we seed the lymphnodes of each unit with a population of antibodies from information learned during a pre-experimental phase: each of the movement patterns described earlier are tested for a variable length of time (in terms of multiples of T_{eval}) to determine utility values. During this phase, external perturbations are applied randomly to the creature in order to obtain a large interval of rotations as condition field of the antibodies. All antibodies are then initialised with concentration c_{init}. In the subsequent sensing phase, in each lymphnode l_{c_i}, the distance between the orientation specified by each antibody in the lymphnode (q) and the current orientation of the unit (q') is determined according to equation 1(a). Within each lymphnode lc_i, antibodies are ordered according to distance and assigned a fitness f_i according to equation 1(b), where ΔS_i and EnC_i are expected functional and non-functional utilities according to the utility field. Using a fitness proportionate selection method based on these fitness values, an antibody is chosen and its action applied to the relevant component (testing phase). It is important to notice that, since this very first step of the algorithm, each unit/lymphnode of the creature can choose actions related to different movement patterns, hence the search for alternative trajectories begins in the first time interval. The testing phase is followed by an evaluation phase in which the adaptation mechanisms take place.

$$
\begin{aligned}
(a)\ d(q,q') &= 2(1-|q\cdot q'|) \\
(b)\ f_i &= \frac{\Delta S_i}{1+EnC_i}
\end{aligned}
\qquad (1)
$$

Adaptation

This is the adaptation phase in which the original movement patterns evolve to optimise the functional and non-functional requirements of the creature. New antibodies are created that combine and/or adapt components of the 5 initial movement patterns into new strategies (see algorithm 1).

Affinity At the end of the evaluation phase, within each lymphnode, the currently active antibody ab_* (as the last selected antibody) discriminates between situations that lead to positive or negative feedback. Positive feedback is received whenever the obtained ΔS_o is greater or equal to the expected ΔS_{exp} stored in the currently selected antibody; negative feedback is received otherwise. Within each internal network, each antibody $ab \in l$ increases its affinity to ab_* if positive feedback is received. In case

$$
\begin{aligned}
&(a)\ \frac{\Delta c_i'}{\Delta t} = K_p \sum_{j=0}^{N} r_{j,i} c_i c_j - K_n \sum_{k=0}^{N} r_{i,k} c_i c_k + K_D \frac{d(q,q')}{1+EnC_i} - K_d c_i \\
&(b)\ r_{i,j} = \frac{T_{ni} + T_{pj}}{T_{i,j}} (1 + |\Delta S_o - \Delta S_{exp}|) \\
&(c)\ \frac{\Delta c_i}{\Delta t} = K_{int} C_{i_{int}} + K_{ext} C_{i_{ext}}
\end{aligned}
\tag{2}
$$

of a negative feedback, the previously selected antibody ab_* will increase its affinity towards the other unselected antibodies $ab \in l$. Affinities are used for calculating the variation of concentration for the antibodies, as shown in equation 2(a). Affinity $r_{i,j}$ between generic antibodies i and j are calculated according to equation 2(b). The same mechanisms are also applied to the external network, as shown in the example situation depicted in Figure 4. The purpose of the external network is to *share experiences* throughout the creature: if, for instance, lymphnode l_{c_i} obtained a negative feedback, the reason for it could be ascribed to the neighbouring units that did not push/pulled in the right direction; vice-versa, if the same lymphnode obtained a positive feedback, the other neighbouring units should be aware of that, even if they experienced a negative result. The SelfEx algorithm should be able to find a balance between these situations.

Equation 2(b) is borrowed directly from Ishiguro et al. (1995) and forces selection of the same antibody in a subsequent similar situation (in the case of positive feedback) and vice-versa. T_{pj} corresponds to the total number of evaluation periods over which ab_j received positive feedback and T_{ni} are the total number of times ab_i received a penalty; $T_{i,j}$ is the number of times both antibodies i and j have been activated. In contrast to Ishiguro et al. (1995) we multiply the quantity representing the ratio of positive to negative feedback by a term proportional to the difference between obtained and expected distance travelled to represent functional utility.

Dynamics The *concentration* c_i of an antibody i determines its ability to compete against other antibodies to be able to execute its action and is dependent on its affinity with other antibodies. We apply an equation very similar to the differential equation suggested by Ishiguro et al. (1995) to modify concentration, given in equation 2(a). As visible in equation 2(a), we can identify four components and each component is controlled by a constant value. The first term represents *stimulation* of an antibody, the second term *suppression*, the third term the *stimulation* of the antibody from the environment (proportional by the distance between conditions and inversely proportional by the energy consumption measure), and the final term is a *decay* term representing the tendency of antibodies to die if not stimulated. In order to calculate the concentration of an antibody due to its *internal* network, $C_{i_{int}}$, the equation is applied over the N_i antibodies in the lymphnode of antibody i. The concentration of the i due its *external* network $C_{i_{ext}}$ is calculated by applying the equation over the N_e antibodies in its external network. Finally, its total concentration its calculated by weighting the contributions from the internal and external network as in (c) where K_{int} and K_{ext} are constants. The final value of concentration is then stabilized through a squashing function as defined in Ishiguro et al. (1995).

Meta-dynamics Meta-dynamic processes create new antibodies which can become integrated into the network. In contrast to previous meta-dynamic models (e.g. Ishiguro et al. (1995)), in our SelfEx model, the creation of new antibodies is not a continuous process but is triggered by stagnation in the network. Specifically, mutation is triggered whenever more than one antibody converges to the same concentration, a condition that (according to initial experiments) occurs when the creature becomes stuck over a prolonged period of time. The α mutation operator, triggered when two or more antibodies reach the same concentration level, creates a new antibody in which the condition field is set to the current unit orientation and all other fields are averaged among all the antibodies that share the same concentration value. To better understand what *averaging the action field* of an antibody means, we can consider an example situation in which two antibodies (i and j) share the same highest concentration value: let us suppose that the action during the positive phase of i is (0,0,-1) while in j is (0,1,0) then the new antibody k will feature an action during the positive phase of (0,0.5,-0.5). As a result, new antibodies enable the unit to move though composite directions, hence providing the single unit with the basis to create new movement patterns.

A second mutation operator refines new antibodies in order to generate new obstacle avoidance strategies. Mutation β occurs according to a probability P_β as described in equation 3. M is the current number of antibodies and it increases whenever an α mutation occurs, while N is the initial number of antibodies. K_a is a constant that regulates how often this kind of mutation can occur. As soon as $M - N > 0$, there is a non-zero probability that a new antibody will be created featuring the same condition and expected utility field as the last selected antibody. The action field is calculated by detecting the dominant direction of the last antibody and then inverting its direction. For example, if the positive phase of the last selected antibody i featured an action represented as (0.8,0.2,0), then the newly generated antibody k will feature an action in the positive phase represented as (-0.8,0,0). Once a β mutation occurs, N is set to M and all the affinity and concentration values are reset to their initial values.

$$P_\beta = \frac{M - N}{K_a} \tag{3}$$

Experiments and Results

After the pre-experimental phase, the same initial values of inter-antibodies affinities and concentrations are assigned to all the antibodies. The mechanism used in the experiment is shown in algorithm 1.

Algorithm 1: mechanisms of adaptation

```
Data: Initial population of antibodies after pre-experiments
Executed by: each lymphocyte i;
Sensing: current orientation → Select antibody → action;
TEST: while ($t < t_{eval}$) do
    test the selected action;
end
performance = evaluate($\Delta S$, Enc);
updated affinities and concentrations;
if $\alpha$MutationCondition is true then
    applyMutation$\alpha$;
    if $\beta$MutationCondition is true then
        applyMutation$\beta$;
    end
    add newly generated antibodies to list of antibodies;
    select newly generated antibody;
end
else
    Select highest concentrated antibody;
end
goto: TEST;
```

The algorithm is tested in an arena represented by a room that is confined between 4 walls. The algorithm is implemented in Java^{tm} (jdk v. 1.7.0_45) using *JOGL* (http://jogamp.org/jogl/www/) for visualization purposes and *jinngine* as physics engine (Silcowitz-Hansen (2010)). After the pre-experimental phase (performed just once before the experiments), each unit in the creature is initialised with a population of 50 antibodies; each antibody is then set to the initial concentration $c_i = 0.5$. Parameters that were fixed during the experiments were $K_a = 3$, $K_d = 0.25$, $T_{eval} = 2s$ while all the other parameters undergo a process of search in order to find their best combinations resulting in settings of $K_p = 1$, $K_n = 0.85$, $K_D = 0.5$, $K_{int} = 0.45$

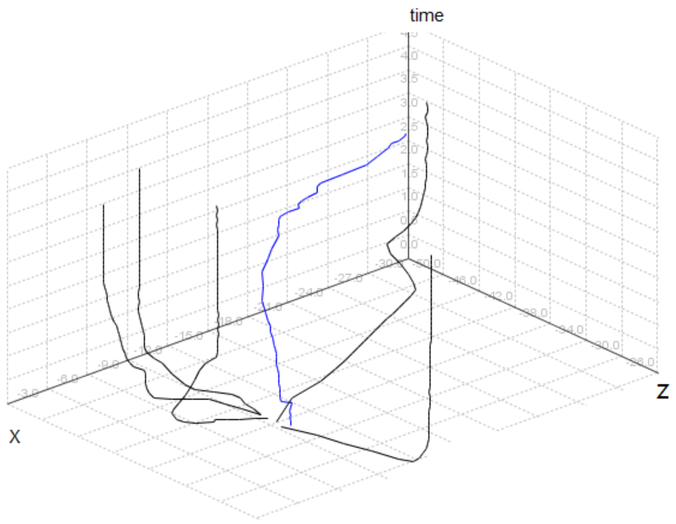

Figure 5: 50 time interval experiment in which all the trajectories caused by the pre-coded movement patterns (in black) are compared to SelfEx (in blue).

and $K_{ext} = 0.55$. Experiments are repeated 10 times with the total energy consumed and average distance covered (per single time interval) over varying time intervals; both these indicators are averaged throughout the whole creature. Experimental goals are to: (a) investigate whether pre-coded movement patterns can be combined in order to show different movement trajectories; (b) optimise distance travelled (c) optimise energy consumption.

Indicative results showing trajectories over a single run are depicted in Figure 5 and Figure 6. The y axis indicates movement over time, thus stationary behaviour is observed whenever there is no change in the XZ plane.

In Figure 5 shows that the evolved SelfEx trajectory differs from the single pre-coded movement patterns in that initially it makes slow progress, but once movement is established it continues throughout the experiment in a different trajectory.

In Figure 6 obstacle avoidance in addition to continuous movement is clear, as the plot shows how the creature manage to step back from walls.

In Figure 7 total energy consumption over time is shown, compared to the least costly single movement pattern (Type3) and the most costly (Type2, Type5). Over 10 runs, the average energy consumption is 10.11 (standard deviation $\sigma = 0.128$), while the average distance covered per single time interval is $avg\Delta S = 0.38$ ($\sigma = 0.129$). The comparison with the 5 pre-coded movement patterns is shown in Table 2[2]. Our SelfEx algorithm was able to discover new trajectories by combining *pieces* of pre-coded move-

[2]Do note that the outcomes of the pre-coded patterns are deterministic.

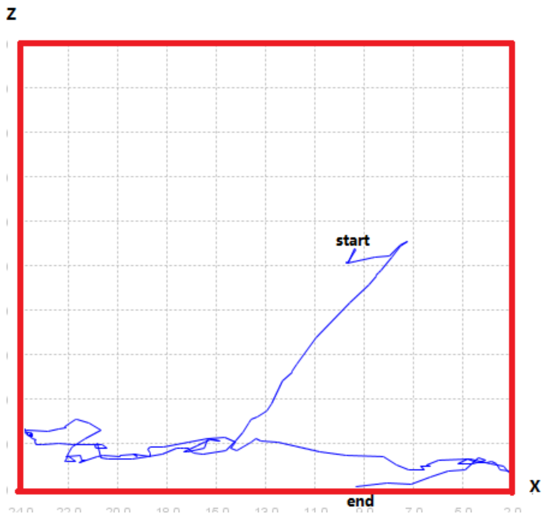

Figure 6: SelfEx trajectory over the XZ plane during 190 time intervals; walls are indicated in red.

MP	EnC_{tot}	$avg\Delta S$
Type1	12	0.26
Type2	15	0.21
Type3	**10**	0.21
Type4	12	0.33
Type5	15	0.25
SelfEx	*10.11*	**0.38**

Table 2: Average perfomance

ment patterns starting from the very first time interval of the experiment. This is an implication of the fact that each unit/lymphnode selects actions related to different movement patterns, thus creating new locomotion strategies even before starting generating new antibodies. The generation of new antibodies occurs whenever mixing the pre-coded strategies still does not cause movement to the creature; moreover the fitness function used during the sensing phase (equation 1(b)) and the third term of eq. 2 (a) forces the unit to constantly take into account its energy consumption, since the increment of the concentration value is inversely proportional to the energy cost of the movement: even if an antibody caused a positive feedback, the increase of the antibody concentration will not be substantial if the consumed energy is high; as an implication of this, less energy consuming actions are always selected.

Conclusions and future work

The paper has presented a modified version of an idiotypic network algorithm to evolve movement in a virtual creature. The algorithm was inspired by early work in robotic control by Ishiguro et al. (1995) and extends previous work in which a modified version of this algorithm was used to

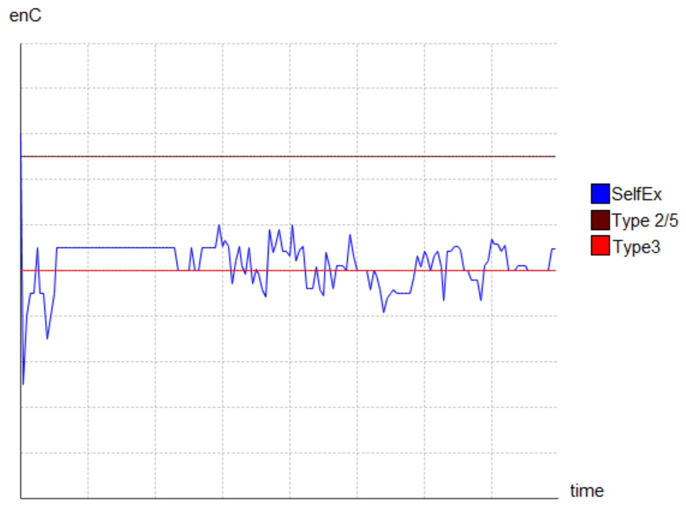

Figure 7: Energy consumption of SelfEx compared to Type1 (low energy), and Type3/Type5 (high energy).

select coordination patterns in swarm robotic applications (Capodieci et al. (2013b)). The presented control system is fully distributed, adaptive, and accounts for both functional and non-functional requirements. By only sharing a biological clock and by relying on a dual idiotypic network, the creature minimizes energy consumption and at the same time, discovers new movement trajectories and obstacles avoidance behaviours. Its novelty lies in the use of a 2-layer network, where antibodies interact in both internal and external networks, trading off unit-control against shared behaviours. Furthermore, the algorithm generates novel locomotion patterns in two ways: by combining fragments of existing patterns and by generating new patterns by mutating existing ones. As for future work, applying the same dual immune network for creating evolutionary morphologies for the creature is definitely an interesting development to be exploited in the near future.

Acknowledgement

The work is partially supported by the ASCENS project (EU FP7-FET, Contract No. 257414)

References

Blumberg, B. M. and Galyean, T. A. (1995). Multi-level direction of autonomous creatures for real-time virtual environments. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 47–54. ACM.

Cabri, G. and Capodieci, N. (2013). Runtime change of collaboration patterns in autonomic systems: Motivations and perspectives. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pages 1038–1043.

Capodieci, N., Hart, E., and Cabri, G. (2013a). Designing self-aware adaptive systems: from autonomic computing to cognitive immune networks. *In Proceedings of the 3rd Workshop on Challenges for Achieving Self-Awareness in Autonomic Systems, Philadelphia, USA*.

Capodieci, N., Hart, E., and Cabri, G. (2013b). An immune network approach for self-adaptive ensembles of autonomic components: a case study in swarm robotics. In *Advances in Artificial Life, ECAL*, volume 12, pages 864–871.

Capodieci, N., Hart, E., and Cabri, G. (2014). Artificial immune system in the context of autonomic computing: integrating design paradigms. *Proceedings of Genetic and Evolutionary Computation Conference, GECCO 2014*, (-):to appear.

Černỳ, J. and Kubalík, J. (2013). Co-evolutionary approach to design of robotic gait. In *Applications of Evolutionary Computation*, pages 550–559. Springer.

Cohen, I. R. (2000a). Discrimination and dialogue in the immune system. In *Seminars in Immunology*, volume 12, pages 215–219. Elsevier.

Cohen, I. R. (2000b). *Tending Adam's Garden: evolving the cognitive immune self*. Academic Press.

De Castro, L. N. and Timmis, J. (2002). *Artificial immune systems: a new computational intelligence approach*. Springer.

Hirose, S. and Morishima, A. (1990). Design and control of a mobile robot with an articulated body. *The International Journal of Robotics Research*, 9(2):99–114.

Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21(4):642–653.

Ishiguro, A., Watanabe, R., and Uchikawa, Y. (1995). An immunological approach to dynamic behavior control for autonomous mobile robots. In *Intelligent Robots and Systems 95. Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ*. IEEE.

Kephart, J. and Chess, D. (2003). The vision of autonomic computing. *Computer.*, 36(1):41–50.

Miconi, T. and Channon, A. (2005). A virtual creatures model for studies in artificial evolution. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 565–572. IEEE.

Palyanov, A., Khayrulin, S., Larson, S. D., and Dibert, A. (2012). Towards a virtual c. elegans: A framework for simulation and visualization of the neuromuscular system in a 3d physical environment. *In silico biology*, 11(3):137–147.

Prez-Moneo Surez, D. and Rossi, C. (2013). A comparison between different encoding strategies for snake-like robot controllers. In *Applications of Evolutionary Computation*, volume 7835 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

Silcowitz-Hansen, M. (2008-2010). Jinngine, a physics engine written in java.

Sims, K. (1994). Evolving 3d morphology and behavior by competition. *Artificial life*, 1(4):353–372.

Whitbrook, A. M., Aickelin, U., and Garibaldi, J. M. (2010). Real-world transfer of evolved artificial immune system behaviours between small and large scale robotic platforms. *Evolutionary Intelligence*, 3(3-4):123–136.