A Metabolic Subsumption Architecture for Cooperative Control of the e-Puck

Verena Fischer and Simon Hickinbotham

Abstract. Subsumption architectures are a well-known model for behaviour-based robotic control. The overall behaviour is achieved by defining a hierarchy of increasingly sophisticated behaviours. We are interested in using evolutionary algorithms to develop appropriate control architectures. We observe that the layered arrangement of behaviours in subsumption architectures are a significant obstacle to automating the development of control systems. We propose an alternative subsumption architecture inspired by the bacterial metabolism, that is more amenable to evolutionary development, where communities of simple reactive agents combine in a stochastic process to confer appropriate behaviour on the robot. We evaluate this approach by developing a traditional and a metabolic solution to a simple control problem using the e-puck educational robot.

1 Introduction

The behaviour-based approach to robotics and artificial intelligence [4] has given a new spirit to a field that seemed lost in abstractions of the real world. While "traditional" robotics built complicated reasoning systems that created models of the real world and successfully produced reasonable behaviour in simple and static environments, it seemingly failed to extend these systems to deal with dynamic real world situations [11]. Behaviour-based robotics works on the assumption that internal representations of the real world are unnecessary to produce reasonable behaviour in dynamic environments and proves this to be true with many examples described in several of Brooks' papers [2, 3].

Verena Fischer
Department of Informatics, University of Sussex, Falmer, Brighton BN1 9QJ
e-mail: vf37@sussex.ac.uk

Simon Hickinbotham
YCCSA, University of York, Heslington, York YO1 5DD, UK
e-mail: sjh@cs.york.ac.uk

J.R. González et al. (Eds.): NICSO 2010, SCI 284, pp. 1–12, 2010.
springerlink.com

Subsumption architectures are as highly engineered as their traditional counterparts. The approach identifies a hierarchy of autonomous behavioural layers with simpler behaviours placed at the lower layers. Every layer produces some behaviour in the robot and the higher layers can subsume the behaviour of lower layers, while lower layers are not aware of the higher ones. During the design of the controller, each layer is implemented as an autonomous system before ascending the hierarchy. The upper layers can override instructions from the lower layers should the situation demand it. Control modules are then assigned to appropriate layers, each of which can connect sensors to actuators in different ways.

The problem of designing any form of layered control remains challenging. For sophisticated environments, the number of layers can proliferate and it becomes unclear where a control module should be placed and what the interconnectedness should be. Attempts to automate the process have tended to simplify the problem by evolving the system a layer at a time [9, 13]. However, it is difficult to ensure that the entire system is optimised, since the overall control rarely depends on a single layer.

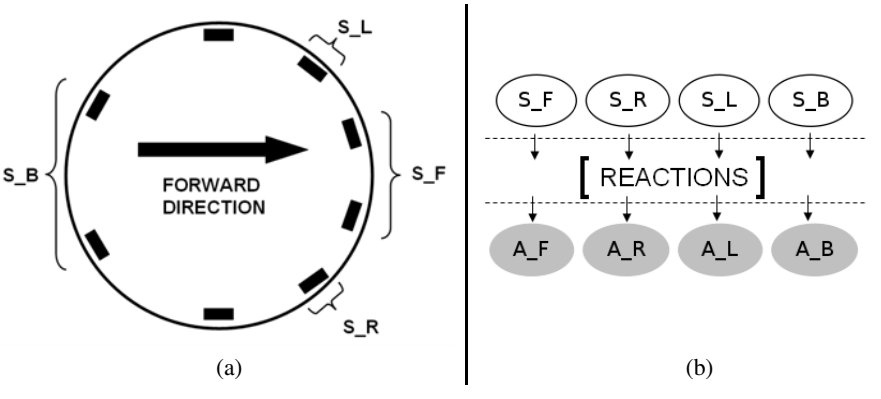
We note that as the behaviours get richer, more and more internal modules are connected only to other internal modules rather than being connected to sensors or actuators. There is potential to make such modules and the connections between them subject to adaptation via evolutionary algorithms since as long as the connections to the outside world are preserved, the internal processing can change. Evolving this sort of network is a difficult challenge however, particularly if the role of modules and their connections is predefined (e.g. connections relating to “feel-force", “heading" and “turn"). We propose a finer-grained solution, in which control is shared amongst a community of very simple processing agents that behave like molecular species in biological reaction networks [7], and whose connections are set by simple reaction rules that can be changed arbitrarily. This metabolic representation allows a high level of interconnectedness between control layers, which is more akin to biological reaction networks than control engineering. The metabolism can be thought of as a community of control agents, which through their interaction rates, network topology and concentrations give rise to emergent behaviour.

This paper compares an implementation of a subsumption architecture controller with a controller based on a model metabolism. We refer to the two systems as “subsumption control" and “metabolic control" respectively. We favour the latter approach because we believe it lends itself more readily to solutions which can be found through artificial evolution [8]. The work we present here shows how an engineered control system can be implemented in an evolvable community control system, and compares the performance of the two.

2 The Robot Model

The platform for our robot experiments is the e-puck, which is a readily available open-source research platform [10]. In addition to the physical hardware being

available, a simulator model is available for the open-source player/stage platform [1, 6]. We developed our control software using the simulator, but we used the physical characteristics of the real robot to constrain the design. The e-puck is equipped with a variety of sensors and actuators. There are eight infrared sensors, as shown in figure 1(a). We combine these into four channels to produce a sufficiently fine-grained reaction to the environment: front (S_F); left-of-front (S_L), right-of-front (S_R), and back (S_B). The e-puck is driven by two wheels, which are controlled by actuator behaviours: Forward speed (A_F); Backward speed (A_B); Left turn speed (A_L); Right turn speed (A_R). In the two control architectures we investigate here, we define two functions called `SensorHandler` and a `RobotUpdater` for the sensors and actuators respectively, to carry out any signal transduction between the e-puck and the control system. The control challenge is thus to link the sensor data to the actuator instructions, as illustrated in figure 1(b).



(a) (b)

Fig. 1 (a) coupling of sensors on the e-puck simulation. (b) the control task: incoming sensor data must be coupled with actuator controllers to determine the speed and heading of the robot

E-pucks have a range of control settings. The proximity sensor range is 0.1 metres, which imposes constraints on the responsiveness of the control system if a collision is to be avoided. The maximum speed of the real e-puck is 0.12m/s, therefore the robot has 0.83s to respond appropriately to an obstacle detected by its sensors. We are of course free to change this speed in the simulation.

As an experimental framework we used the open source platform Player/Stage. Player is a network server that handles the actual low level robot control and provides a clean and simple interface to the robot's sensors and actuators. Stage simulates mobile robots, sensors and objects in a 2D environment and therefore provides the hardware basis for the robot control handled by Player.

3 Subsumption Architecture

A subsumption architecture is a layered control system. Our subsumption architecture control system contains the following layers:

Layer 1: AVOID responds to obstacles flagged by the SensorHandler and changes the speed of the robot accordingly. Each sensor produces a different hard coded behaviour, although the behaviour has a small noise component built into the design. The avoid behaviour sets a new heading as soon as a sensor flags an obstacle. Details are given in table 1

Layer 2: WANDER pseudo-randomly produces a new heading and speed for the robot after a set number of time steps. The goal is to induce a behaviour which allows the robot to explore its world. When wandering, headings are set every 15 time steps by selecting a turn value in the range $+/-15$ degrees, while the speed is set to a pseudo-random value between 0.05 - 0.095 m/s.

Table 1 Reactions specified by the AVOID layer in the subsumption architecture

Sensor direction	speed	turn
S_F	A_B: -0.15 m/s	A_R: 90-180 degrees
S_L	A_B: -0.1 m/s	A_R: -60 degrees
S_R	A_B: -0.1 m/s	A_L: 60 degrees
S_B	A_F: 0.15 m/s	

4 Artificial Metabolomes

Our endeavours to create a mobile robot control system using an artificial metabolism are built upon the particle metabolome developed within the Plazzmid project [12]. The metabolic model is composed of four components. Firstly, there exists a *container*, which specifies the volume and dimensionality of the space in which the agents exist. We specify a simple 2D container of area $v_c = 40$ units. Secondly, we have a set of metabolite *agents*, of area $v_a = 9$ units which are present in varying quantities in the container, analogous to the various quantities of different molecular species in a biological system. Thirdly we have a stochastic *mixer*, which governs the movement and changes in adjacency of the elements within the container. For a bimolecular reaction such as the bind B, our mixer utilises a simple propensity function $\mathrm{P}(B)$, which estimates the probability of two agents being sufficiently close enough for the reaction to occur. For any one agent in a bimolecular reaction, the chance of the second agent in the reaction being close enough to react is:

$$\mathrm{P}(B|v_c, v_a, n) = 1 - (1 - (v_a/v_c))^n \tag{1}$$

where n is the number of instances of the second agent in the metabolism. Space in the system is represented abstractly via the ratio of container area to agent area. Apart from this consideration, the model is aspatial. Fourthly, agents react according to a set of *rules*, which specify the reactions in the system. There are four types

of rules in the system as shown in table 2. Each rule has a rate, which governs how often the reaction occurs when it is selected via a stochastic process. *Influx* is the spontaneous generation of new agents in the system. In our case, objects detected by sensors cause production of corresponding sensor agents to be generated in the metabolism. *Binding* occurs when two reactants combine to create a single product. Binding is the only bimolecular reaction permitted. Bimolecular reaction rates are governed by the concentration of the two reactants in the system (via $P()$) and a further reaction rate specified by the reaction rule. Behavioural switching is caused by a sensor agent binding with a WANDER enzyme to produce an AVOID enzyme. *Dissociation* is the splitting of a single agent into two agents. A dissociation rule which has the same agent type on either side of the reaction (for example $\mathsf{A} \rightarrow \mathsf{A}+\mathsf{X}$) can be thought of as representing the production of new agents using materials that are available at saturation in the metabolism, and whose concentrations are not modelled for computational expediency. *Decay* is the spontaneous removal of an agent from the system, and is important for sensor and actuator molecules, which must decay quickly in order for the system to be responsive. Note that uni-molecular changes from one molecular species to another are not permitted. The probability of a bimolecular reaction is the product of the propensity and the reaction rate. Unimolecular reactions are governed by their reaction rate alone, since adjacency does not need to be considered.

Table 2 The four types of reaction rule in the metabolic controller

Reaction	Rule format	network symbol
Influx:	$\rightarrow \mathsf{A}$	□
Binding:	$\mathsf{A}+\mathsf{B} \rightarrow \mathsf{C}$	●
Dissociation:	$\mathsf{A} \rightarrow \mathsf{B}+\mathsf{C}$	○
Decay:	$\mathsf{A} \rightarrow$	■

These ingredients allow us to specify a metabolic control model for our e-pucks. For a more detailed overview of this metabolic model see [7].

In our metabolic controller, there are three classes of agents which possess different qualities within this framework. *Sensor* agents are generated when a sensor detects an obstacle. These are shown in white on the network diagrams below. *Actuator* agents are used to govern the speed and turning rate of the robot. These are shown in grey. Both sensors and actuators decay quickly, in order to allow the robot to be responsive. *Enzyme* agents form the connectivity between sensors and actuators. They are shown in black. Enzymes do not decay, but can be changed into other enzymes by reacting with other agents in the system. Reactions must be designed such that the total number of enzymes in the system is conserved.

5 Metabolic Subsumption

The metabolic network that we have designed for the robot control is based on the control layers described in section 3. We describe here the reaction system that we use to build behaviours that emulate these layers.

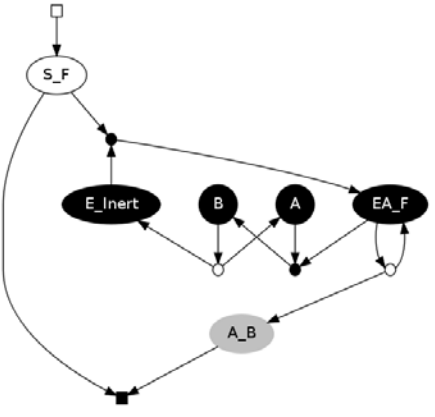


Fig. 2 Change in behaviour as a result of a sensor event. The metabolism switches from an inert behaviour to an avoidance behaviour on the influx of S_F agents by producing A_B agents

Figure 2 shows the network of an AVOID behaviour for a single sensor/actuator pair, which illustrates the basic reaction system we use for metabolic robot control. Symbols for reactions are described in table 2 and associated text. The system exists in an inert state until sensor information is received. In this state, the only agent types present in the system are the enzyme E_Inert and the deactivator enzyme A. When the sensor is activated, sensor agents of type S_F are generated in the metabolism. S_F binds with E_Inert to create EA_F. This enzyme uses a dissociation rule to create a copy of itself and the actuator agent A_B, which instructs the robot to move backwards. Once sensor agents have decayed out of the system, enzyme EA_F binds with A to produce an intermediate B. B then dissociates back to E_Inert and A.

The network in figure 2 shows an AVOID behaviour for a single sensor and a single actuator. We extend this model in figure 3 to show the metabolic network for one sensor type that produces AVOID behaviour subsumed by a WANDER behaviour appropriate to the input. Since wandering involves moving in a particular direction, actuators for turning are required. Information from the sensor is represented as quantities of S_F agents, which bind with the enzyme EW_F to create the EA_F. The avoid enzyme produces the signalling agents A_B that instruct the robot to reverse away from the obstacle. Note that EW_F and EA_F produce different actuator enzymes, whereas EW_L and EA_L produce A_L at different rates, appropriate to the dominant behaviour. (We have not represented these different rates on

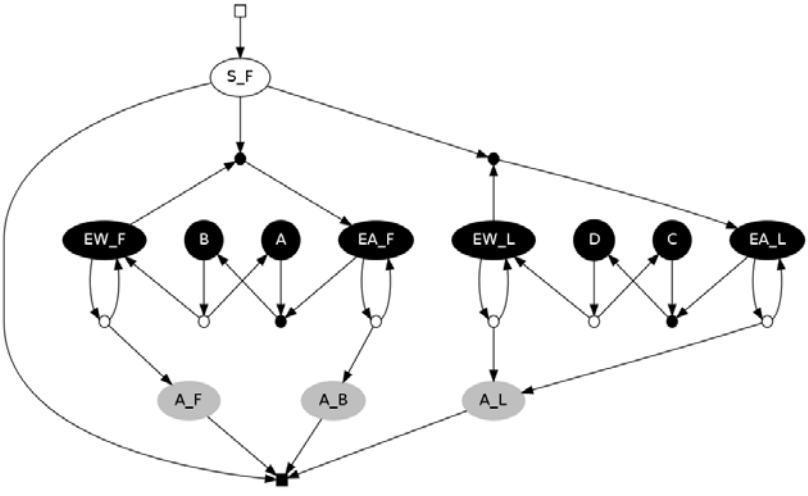


Fig. 3 A metabolic network for input coming from a front sensor

the diagram for clarity.) This metabolic approach to control emulates a subsumption architecture since the AVOID behaviour is autonomous and there is a switching mechanism from one behaviour to another as the situation demands it. As long as an avoid reaction is needed the wander behaviour is inhibited, because the wander enzymes are turned into avoid enzymes. When it is not needed anymore, i.e. no sensor agents are injected, the avoid enzymes are turned back into wander enzymes. For every type of sensor agent, the binding strengths are different to produce a different change of speed. For example, if there is an obstacle directly in front of the robot, there needs to be a wider turn than would be needed if the obstacle were slightly to the left or right.

The complete control network for the metabolic subsumption is shown in figure 4. Such a network becomes necessarily complex when information from four sensors is combined using a simple reaction rule set. Although the control architecture for each sensor follows the same basic pattern, there are subtle differences. The most striking of these is the sub-network for the rear sensor S_B. This is for two reasons. Firstly, the WANDER behaviour has no connection to the actuator enzyme A_B since when wandering the e-puck always moves in the forward direction as governed by A_F. Secondly, if S_B is present, the e-puck should move forward just as in the WANDER behaviour, but remaining A_B agents from previous reactions might have to be counteracted, so that more A_F have to be produced to ensure a forward movement.

We use the graphviz program [5] to visualise the network that our reaction rules represent. Although this is a useful tool, the high level of connectedness in the network prevents the automatic creation of network visualisations that makes the two control layers distinct. Although the concept of control layers is essential to the design of the subsumption architecture, the embodiment of the layers in the metabolic

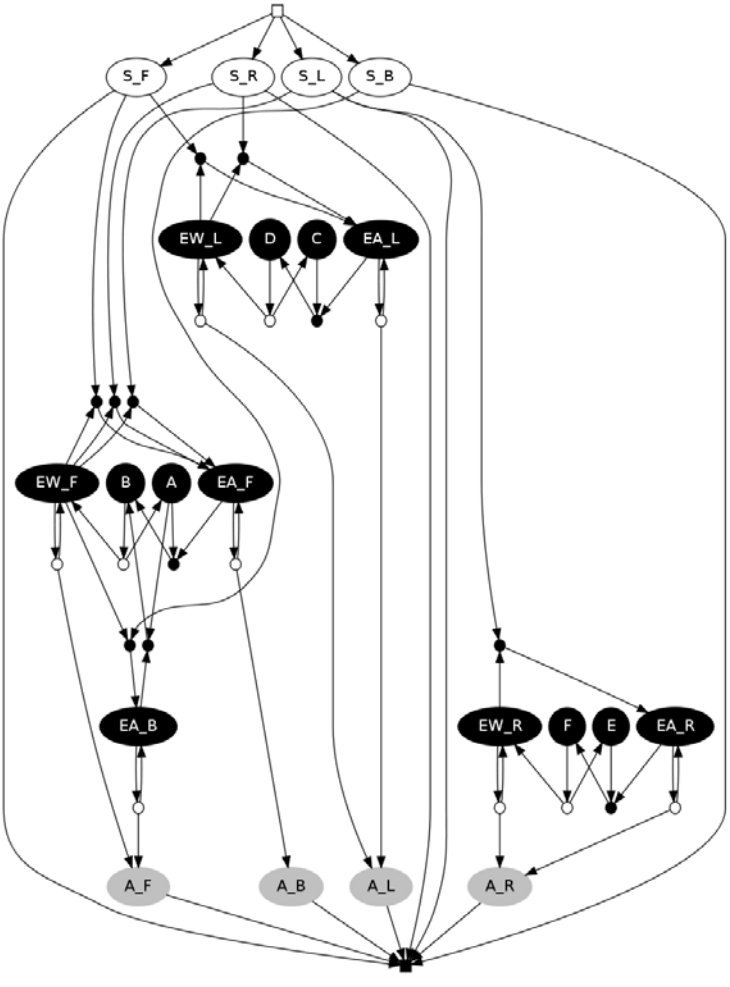


Fig. 4 The metabolic network for 4 input directions: front, front-left, front-right and back

network exhibits strong connectivity between the layers. This rich connectedness means that small changes at the level of nodes and reactions between them have the potential to cause larger changes at the emergent level.

When the metabolism is initialised, all that is present in the network are 10 of each of the WANDER enzymes, and 1 of the return enzymes A, C, and E. Actuator agents for the wander behaviour are created as the metabolism runs. When a sensor detects an obstacle, 10 of the corresponding sensor enzymes are placed in the metabolism. The reaction rates that were used in our experiments are shown in table 3.

Table 3 Reaction rates of a metabolic control network for the e-puck

Reaction	Rate	Reaction	Rate
Actuator signals for WANDER		*Actuator signals for AVOID*	
$EW\_F \rightarrow EW\_F + A\_F$	0.8	$EA\_F \rightarrow EA\_F + A\_B$	0.9
$EW\_L \rightarrow EW\_L + A\_L$	0.16	$EA\_B \rightarrow EA\_B + A\_F$	0.9
$EW\_R \rightarrow EW\_R + A\_R$	0.16	$EA\_L \rightarrow EA\_L + A\_L$	0.9
		$EA\_R \rightarrow EA\_R + A\_R$	0.9
Switch to AVOID behaviour		*Reversion to WANDER*	
$EW\_F + S\_F \rightarrow EA\_F$	0.9	$EA\_F + A \rightarrow B$	0.1
$EW\_L + S\_F \rightarrow EA\_L$	0.9	$EA\_B + A \rightarrow B$	0.1
		$B \rightarrow A + EW\_F$	0.1
$EW\_F + S\_R \rightarrow EA\_F$	0.7		
$EW\_R + S\_R \rightarrow EA\_R$	0.7	$EA\_L + C \rightarrow D$	0.1
		$D \rightarrow C + EW\_L$	0.1
$EW\_F + S\_L \rightarrow EA\_F$	0.7		
$EW\_L + S\_L \rightarrow EA\_L$	0.7	$EA\_R + E \rightarrow F$	0.1
		$F \rightarrow E + EW\_R$	0.1
$EW\_F + S\_B \rightarrow EA\_B$	0.1		
Decay of actuators		*Decay of sensors*	
$A\_F \rightarrow$	0.15	$S\_F \rightarrow$	1
$A\_B \rightarrow$	0.05	$S\_B \rightarrow$	1
$A\_L \rightarrow$	0.1 2	$S\_L \rightarrow$	1
$A\_R \rightarrow$	0.1 2	$S\_R \rightarrow$	1

6 Experimental Evaluation

An appropriate WANDER behaviour should allow the robot to explore the arena without giving it any particular strategy of exploration. The two control strategies were manually tuned such that their average speeds were approximately equivalent. We evaluated this via a visual inspection of the routes of the e-puck using the two different controllers. Sample traces for both controllers during a 5 minute run (simulated time) are shown in figure 5. It is clear that both controllers induce behaviour that can be interpreted as “wandering". However, it is difficult to obtain a quantitative evaluation of the pattern of exploration that the two control strategies confer on the e-puck.

Successful behaviours should prevent the e-puck from colliding with obstacles and walls. To compare the performance of the control systems we looked at 50 wall encounters for each set-up and counted the number of collisions. Since the metabolic controller was more difficult to tune, we compared a single metabolic controller with three subsumption architectures with different average and maximum speeds. Collision events for both controllers are shown in table 4. It is clear that the subsumption controller is more successful at responding to obstacles since the number of

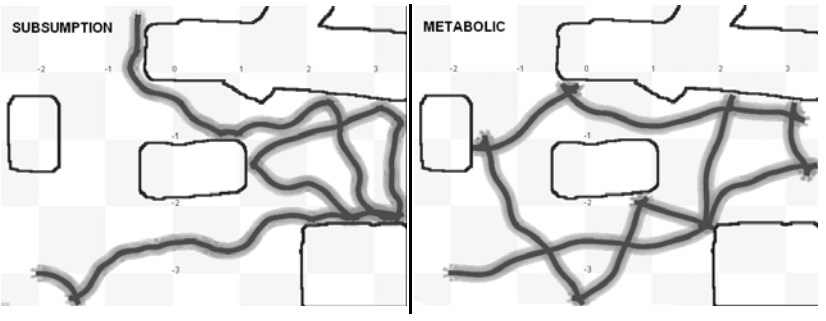


Fig. 5 A trace of the subsumption robot control running for a simulated time of 5 minutes

collisions is the same when it travels at nearly twice the speed of an e-puck that uses the metabolic controller.

Both control systems were designed to reverse away from an obstacle recorded on the sensors. While the subsumption architecture is able to change the speed and heading immediately upon receipt of a signal from a sensor, the metabolic model suffers from latency in its response. The wall encounters for the metabolic controller shown on the right of figure 5 are different from those on the right for the subsumption controller because of the latency in response between sensors and actuators. Latency is caused by the actuator agents that are present in the metabolism as the sensor data comes in. When an obstacle is encountered, actuator agents must be generated to counteract the actuator agents from the wander enzymes extant in the metabolism. This is illustrated in figure 6, which shows the changes in enzyme and actuator levels after an obstacle is encountered on sensor S_F. It is clear that this configuration of the metabolic controller cannot respond immediately to an obstacle since there are about 50 A_F agents in the system which instruct the robot to move forward. This situation could be changed by tuning the disassociation rate of A_F and making the enzyme EW_F produce A_F more quickly and so maintain a similar number of A_F whilst the WANDER behaviour is dominant.

Table 4 Area (in pixels) covered by the control systems and collisions for 50 wall encounters

	Subsumption			**metabolic**
Max speed (preset)	0.3	0.15	0.15	0.15
Average speed (recorded)	0.15	0.15	0.075	0.075
Median area covered for 5 runs of duration 5 minutes	6,264	6,491	3,460	3,574
collisions out of 50 wall encounters	**11**	**4**	**0**	**4**

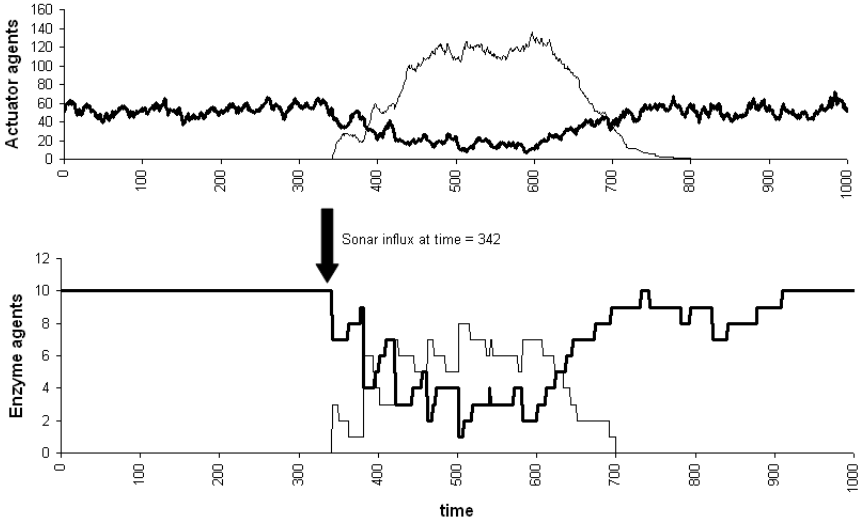


Fig. 6 Change in levels of actuator agents (top) and enzyme agents (bottom) as a result of a sensor event, indicated by the black arrow. The wander enzyme EW_F (thick line) is converted to the avoid enzyme EW_F (thin line) by binding with sensor agent S_F. This results in a change in the levels of two actuator agents in the metabolism: A_F (thick line) diminishes in quantity as A_B (thin line) accumulates, resulting in a change of direction from a forward to a backward motion

7 Conclusions

We have succeeded in our main objective to create a metabolic control system that emulated a simple subsumption architecture. This was motivated by the concept that a chemical reaction network would be more amenable to evolutionary adaptation.

A key difference between the two network types is that the metabolic architecture has *node multiplicity* - each node is represented by a quantity of autonomous agents. Each agent is capable of reacting with agents representing other nodes in the network. The metabolic approach lends itself to evolutionary adaptation [12], since agents for each node in the network can play a number of roles, allowing for duplication and divergence of function on evolutionary timescales. In this work, both the subsumption architecture and the metabolic architecture had to be “engineered" in the sense that the actual avoiding reaction and speeds needed to be optimised by hand. This approach allowed us to establish that an appropriate metabolic control could actually be produced within this framework.

Although the behaviour of both systems is qualitatively similar, the metabolic system suffers from latency in its reaction to obstacles. While the subsumption architecture basically reacts immediately to sensor inputs, the metabolic control needs some time to perform the necessary reactions and produce a sufficient metabolic response. This means that the metabolic control reacts inherently slower than the sub-

sumption architecture. However, it should be noted that our metabolic controllers are more difficult to engineer by hand, since they have been designed to be trained *a posteriori* by an evolutionary system. Our goal was not to implement a system that performs “better" than a traditional subsumption architecture. Instead we focussed on creating a system that lends itself more readily to evolutionary adaptation than a traditional subsumption architecture. Our future work with the e-puck will therefore concentrate on implementing evolutionary adaptation in the metabolic controller.

Acknowledgements

The authors thank Susan Stepney, Peter Young, Tim Clarke, Edward Clark and Adam Nellis for comments and suggestions during the preparation of this manuscript. Verena Fischer is funded by the TRANSIT project, EPSRC grant EP/F032749/1. Simon Hickinbotham is funded by the Plazzmid project, EPSRC grant EP/F031033/1.

References

[1] Anon: Player-driver for e-puck robots (2009), `http://code.google.com/p/epuck-player-driver/`
[2] Brooks, R.A.: Elephants don’t play chess. Robotics and Autonomous Systems 6(1&2), 3–15 (1990)
[3] Brooks, R.A.: Intelligence Without Reason. In: IJCAI 1991, pp. 569–595 (1991)
[4] Brooks, R.A.: Cambrian intelligence. MIT Press, Cambridge (1999)
[5] Ellson, J., Gansner, E.R., Koutsofios, E., North, S.C., Woodhull, G.: Graphviz - open source graph drawing tools. Graph Drawing, 483–484 (2001)
[6] Gerkey, B.P., Vaughan, R.T., Howard, A.: The player/stage project: Tools for multi-robot and distributed sensor systems. In: ICAR 2003, pp. 317–323 (2003)
[7] Hickinbotham, S., Clark, E., Stepney, S., Clarke, T., Young, P.: Gene regulation in a particle metabolome. In: CEC 2009, pp. 3024–3031. IEEE Press, Los Alamitos (2009)
[8] Hickinbotham, S., Clark, E., Stepney, S., Clarke, T., Nellis, A., Pay, M., Young, P.: Molecular microprograms. In: ECAL 2009 (2009)
[9] Liu, H., Iba, H.: Multi-agent learning of heterogeneous robots by evolutionary subsumption. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O’Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1715–1718. Springer, Heidelberg (2003)
[10] Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J., Floreano, D., Martinoli, A.: The e-puck, a robot designed for education in engineering. In: Robotica 2009, pp. 59–65 (2009)
[11] Pfeifer, R., Scheier, C.: Understanding Intelligence. MIT Press, Cambridge (1999)
[12] Stepney, S., Clarke, T., Young, P.: Plazzmid: An evolutionary agent-based architecture inspired by bacteria and bees. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS (LNAI), vol. 4648, pp. 1151–1160. Springer, Heidelberg (2007)
[13] Togelius, J.: Evolution of a subsumption architecture neurocontroller. Journal of Intelligent and Fuzzy Systems 15(1), 15–20 (2004)