Controlling task distribution in monee

Evert Haasdijk, Nicolas Bredeche

Controlling Task Distribution in MONEE

Evert Haasdijk[1] and Nicolas Bredeche[2,3]

[1]Vrije Universiteit Amsterdam, Netherlands
[2]UPMC Univ Paris 06, UMR 7222, ISIR, F-75005, Paris, France
[3]CNRS, UMR 7222, ISIR, F-75005, Paris, France
e.haasdijk@vu.nl, nicolas.bredeche@isir.upmc.fr

Abstract

The MONEE framework endows collective adaptive robotic systems with the ability to combine environment- and task-driven selection pressures: it enables distributed online algorithms for learning behaviours that ensure both survival and accomplishment of user-defined tasks. This paper explores the trade-off that must be reached between these two (possibly contradictory) requirements, in the case where a foraging task is defined by the user. In particular, we study the impact of enforcing specialisation (i.e. the collective must acquire two mutually exclusive foraging skills) as well as the mechanism for tuning the level of specialisation in an on-line fashion. Results show that the actual behaviour of the collective system can be guided on request during the course of evolution in order to achieve a particular distribution of specialisations, albeit within a certain range of values.

Introduction

The work in this paper is inspired by a vision of a collection of robots that evolve to survive and operate in an environment where human control can be effected only intermittently. In such circumstances, the robots have to act autonomously, without direct human intervention. They must therefore survive long periods without any guidance and when they do receive guidance, it is at a considerable delay. The environment is not completely known at deployment time and it changes over time, as do the tasks that the robots have to complete. Therefore, the robots must adapt to survive the environment and to perform their tasks.

The environment in which robots operate indirectly circumscribes goals for the population of organisms to survive and evolve, but does so without specifying objective functions: the robots must for instance move about to spread their genomes, or they must maintain their energy levels, but these goals are not defined directly: it is just that robots that display this behaviour get more opportunities to procreate. By virtue of its similarly unbounded nature, biological evolution has resulted in the high levels of adaptability and robustness that we see in natural living organisms. To exploit this creative potential in a system of evolving robots (or robot controllers), we would want to give evolution as much freedom as possible, pushing for open-ended, unbounded adaptivity, unconstrained by user-defined objective functions.

On the other hand, if the system is to be of any practical relevance, the robots must of course also perform user-defined tasks, pushing for specific, crisply defined task-related objectives.

Evolution has been employed to achieve both of these facets. Artificial Life research abounds with examples of objective-free evolutionary systems since the 1980s (Langton, 1989, 1995). In such experiments, evolution serves as a force for adaptation. Evolutionary robotics research typically employs evolution as a force for optimisation when it focusses on the task-driven aspect (Nolfi and Floreano, 2000).

Balancing these two aspects of evolution –environment-driven adaptation and task-driven optimisation– represents a vital step towards implementing our vision of autonomous, functional, responsive and self-sufficient robot collectives.

In earlier work, we presented the MONEE (**M**ulti-**O**bjective a**N**d open-**E**nded **E**volution) to solve the problem of combining objective-free and task-driven evolution in a single algorithmic framework (Haasdijk et al., 2013).

The principal idea behind MONEE is to employ concurrently two selection mechanisms in different roles: environmental selection for open-ended evolution and parent (or mate) selection for task-driven adaptation. As the 'Multi-Objective' part of the name implies, MONEE accommodates settings with multiple tasks. Jones and Mataric noted that collectively tackling multiple tasks also entails a division of work (2003). If there are multiple tasks, the population of robots as a whole must tackle all of them, even though individual robots may specialise in only a subset. To cope with such cases the MONEE framework uses a market mechanism. This mechanism regulates task-based rewards during mate selection according to the market logic that scarcity increases worth. In our multiple task context this implies that tasks that only a few robots (can) perform yield relatively high rewards and therefore higher selection probabilities.

We showed that the MONEE paradigm does indeed allow

the robots to adapt their behaviour to the environment as well as to multiple tasks. Also, MONEE's market mechanism is crucial to keep the population from focussing exclusively on easier tasks, even when the environment induces specialisation in particular tasks at the individual robot level (Haasdijk et al., 2013).

The market mechanism offers an intriguing possibility for intervention in the adaptive process: users can define premiums for particular tasks to (de-)emphasise their importance and promote or prevent their take-up by the robots. This amounts to defining an exchange rate between credits earned for the various tasks. Such premiums provide a straightforward and intuitive method for human-*on*-the-loop intervention in the behaviour of the robot collective.

We perform an experimental analysis of the influence of premium settings in an implementation of the MONEE paradigm where a simulated population of robots has two tasks: it must collect red and green pucks. The experiment is set up so that controllers for each task must be learned separately. In particular, our research questions are:

- To what extent can a premium direct the focus of the robot swarm to a particular task?
- Does a negative premium prevent the robots from displaying particular behaviour?
- How does swarm behaviour react to changing premium settings?

Related Work

Bredeche et al. (2012) describe mEDEA, an open-ended evolutionary algorithm where autonomous robots move around an arena while continually broadcasting their genome over a short range. Meanwhile, they also receive genomes from other robots that come in communication range. When a robot's lifetime expires, it randomly selects one of the received genomes, modifies that using mutation and starts a new life of broadcasting this new genome. This set-up promotes, with only environmental selection, robot movement through the environment: genomes that cause the robot to move around a lot are spread at a much higher rate than genomes that cause their host to stand still.

Similar settings have been extended with forms of parental investment, for instance in Mascaro et al. (2005); Ventrella (2005); Schwarzer et al. (2010). In artificial life parental investment is often used to give the offspring a starting value of (virtual) energy (Menczer and Belew, 1996; Menczer et al., 1994; Burtsev et al., 2001; Scheutz and Schermerhorn, 2005) and a parent's energy level is often linked to task performance (e.g., agents tasked with eating grass to gather energy in Burtsev et al. (2001)). Distributed on-line evolutionary systems such as Watson et al.'s embodied evolution similarly employ task-related (virtual) energy to determine parent and survivor selection (Watson et al., 2002; Wischmann et al., 2007), typically considering single tasks. These experiments showed that task-related virtual energy (equivalent to credits for appropriate behaviour) is an effective way to guide evolutionary adaptation to tackle tasks.

Market-based schemes provide a well-known solution to the task allocation problem in multi-agent and multi-robot settings, for instance in (Walsh and Wellman, 1998; Tang and Parker, 2007).

Fitness sharing is a well-known technique that was introduced to promote genetic diversity and so prevent premature convergence in evolutionary algorithms. With fitness sharing, an individual's fitness is reduced if there are many similar (in terms of their genetic makeup) individuals in the population. Traditionally, fitness sharing is not necessarily associated with multiple objectives, but with maintaining diversity in general – typically, but not exclusively, in single-objective settings.

MONEE: Multi-Objective & Open-Ended Evolution

As mentioned above, earlier work showed that MONEE effectively combines environment- and task-driven adaptation (Haasdijk et al., 2013). The population of robots shows similar adaptation to the environment with MONEE as it does with its purely environment-driven counterpart mEDEA. In addition, the robots learn to perform puck-collecting tasks. They equitably distribute the collective foraging effort over different puck types, even when one type is more prevalent than the other or when the environment inhibits individual robots gathering multiple types of puck.

The robot –actually, their controllers'– lifecycle in MONEE consists of two phases: life and rebirth. The robot controllers have a limited, fixed, lifetime during which they perform their actions; moving about, foraging, et cetera. When their lifetime ends, they enter a rebirth phase and become 'eggs': stationary receptacles for genomes that are transmitted by passing live robots. This rebirth phase also lasts a fixed amount of time, and once this has passed, the egg selects parents from the received genomes to create a new controller. The robot then reverts to the 'life' role with this new controller. Thus, robot controllers can procreate by transmitting their genome to eggs, and the more eggs a robot inseminates, the more chances it has for procreation. Because the transmission of genomes is continuous and at close range (e.g. through infrared), the more a robot moves about the arena, the better its chances of producing offspring. This aspect of MONEE is open-ended in the sense that it is objective-free: there is no calculated performance measure that defines the chances of being selected as parent, there is no task. Only the environment and robot behaviour dictates what robots may or may not become parents.

To add task-driven parent selection to this basic evolutionary process, the robots can, during their lifetime, amass

credits by performing tasks. For instance, a robot could get one credit for every piece of ore it collects, one for successfully solving some puzzle, and so on. If multiple tasks are defined, the robots maintain separate counts for the credits awarded for each task, for instance one counter for the pieces of ore collected and another one for the number of puzzles solved. When a robot inseminates an egg, it passes the current credit counts along with the genome and the egg uses that information to select parents when it revives. This scheme is reminiscent of parental investment, but it differs subtly yet crucially from most parental investment schemes: a parent does not actually invest when impregnating an egg because the credits aren't *transferred* but *copied* at no cost to the parent.

When a robot's egg phase finishes, it compares the parents' credits for each genome it has received. To enable this comparison across tasks, the egg calculates an exchange rate between tasks. This ensures that genomes that invest in tasks for which few credits are found overall (presumably hard tasks) are not eclipsed by genomes that favour easier tasks.

The credits relate task performance to reproductive success: besides the open-ended goal of 'merely' transmitting genomes to eggs, robots must also become proficient at the defined tasks for these genomes to be selected. The more proficient a robot is at a task, the higher its chances of procreating. The comparison of credits across multiple tasks introduces an exchange rate between the earnings per task: the more common credits are for a particular task, the less their worth and vice versa. Thus, parent selection becomes a marketplace for skills and features that the user requires. This system naturally caters for multi-objective approaches.

MONEE's market mechanism is similar to fitness sharing in the sense that it also reappraises fitness, favouring tasks that are less commonly tackled by robots in the population. A crucial difference with traditional fitness sharing is that MONEE considers an individual's *behaviour*, not its genetic make-up (reminiscent of *syntactic* fitness sharing in genetic programming (Nguyen et al., 2012)). Hence, it does not promote genetic, but behavioural diversity: it modifies fitness not to prevent premature convergence, but to ensure that the robot population tackles multiple tasks.

It also allows the user to prioritise tasks in a straightforward manner: the user can influence the credit comparison by defining a premium for some or all of the tasks. For instance, if she deems collecting ore more important than solving puzzles, she can define a premium for collecting ore; the credits earned through this task are then multiplied by the premium. Compared to not defining a premium (or defining a premium of 1), setting a premium > 1 increases the payoff for the relevant task, setting it between 0 and 1 reduces it, while setting it to a negative value causes the robot adaptation to shy away from the task.

The pseudo-code in algorithm 1 details the credit comparison market mechanism with premiums defined.

```
for every defined task do                                   // total credits
    for every received genome do
        credits_task ←
        credits_task + (premium_task · genome.credits_task)
    end
    credits_overall ← credits_overall + credits_task
end
for every defined task do                          // exchange rate per task
    rate_task ← credits_overall / credits_task
end
for every received genome do                              // credits per genome
    for every defined task do
        genome.rating ← genome.rating + (premium_task ·
        genome.credits_task · rate_task)
    end
end
                                          // select, mutate and revive
parent ← rank_based_selection(received genomes)
child ← mutate(parent)
reactivate(child)
```

Algorithm 1: MONEE's market mechanism

Experimental Set-up

We implemented the MONEE algorithm in a simple 2D simulator called RoboRobo (**?**) In our experiments, 100 simulated robots are placed in an environment that contains obstacles and pucks. The sides of the square arena are roughly 330 robot body lengths long (1024 pixels in the simulator), and it contains a number of obstacles (see Fig. 1). We run 64 repeats of each experiment.

The environment contains two types of puck: green and red, defining a concurrent foraging scenario. Concurrent foraging is a variation of regular foraging where the arena is populated by multiple types of objects to be collected (Jones and Mataric, 2003), rather than just a single resource. In our case, these objects are green and red pucks and the collection of each different colour is a different task. The pucks are distributed throughout the arena, and they are immediately replaced in a random location when picked up. The (re-)placement of pucks is governed by a 2D gaussian distribution centred on the middle of the arena and with σ of half the arena width. The robots move around the arena, spreading their genome as they encounter eggs and dying when their allotted time has passed. They collect pucks simply by driving over them and the more pucks they gather, the more likely their genome is to be selected once an egg they impregnated revives.

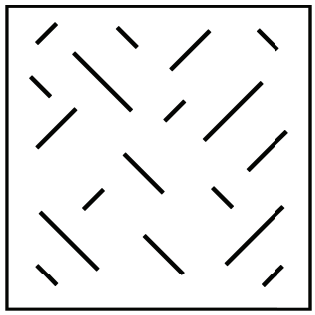

Figure 1: Experiment arena.

To detect pucks, the robots have 16 sensors that detect either red or green pucks (i.e., 8 sensors per puck-type). Each

set of 8 sensors is laid out so that 6 face forward, 2 face to the rear (similar to, for instance, obstacle sensors on a khepera robot). Because individual sensors only detect a single type of puck, collecting one type of puck is a task distinct from (but very similar to) collecting the other type of puck. Thus, behaviour to collect either type of puck has to evolve separately.

Each robot is controlled by a single-layer feed forward neural network which controls its left and right wheels. The inputs for the neural network are the robot's puck and obstacle sensors.

The robot's genome directly encodes the neural network's weights (3 types of sensor $\times$ 8 sensors $\times$ 2 outputs plus 2 bias connections plus 4 feedback (current speed and current rotation to either output) = 54 weights) as an array of reals.

As mentioned, the robots alternate between periods of explorative puck gathering and motionless genome reception. To prevent synchronised cycles among the robots, we add a small random number to each robot's fixed lifetime. This forces desynchronised switching between life and rebirth even though our runs start with all robots perfectly in sync at the first time-step of their lifetime.

At the end of the egg phase offspring is created by selecting a parent from the received genomes as shown in algorithm 1 and mutating the weights in that genome using gaussian perturbation with a single, fixed mutation step size $\sigma = 1$. This single-parent, mutation-only scheme is common in evolution strategies that are known to perform well on problems with continuous-valued genomes (Beyer and Schwefel, 2002).

Note that MONEE does not prescribe any particular controller implementation nor any choice of variation operator. The implementation we chose here of an artificial neural network with the weights encoded as real-valued genes provide a convenient, flexible and well-established representation.

Table 1 summarises the experimental set-up. The paragraphs below describe our experiment's variants in detail. Code for the experiments is available at `http://pages.isir.upmc.fr/evorob_db/moin.wsgi`.

Experiment details	
Robot group size	100
Simulation length	1,000,000 time-steps
Number of repeats	64
Number of pucks	500, 150 or 50 green, 500 or 150 red
Arena	See fig. 1
Premium settings	-1,0,1,2,5,10,20,50,100
Controller details	
Controller	Perceptron neural net
Input nodes	8 obstacle sensors, 16 puck detectors, 2 bias and 2 recurrent nodes
Output nodes	2 (left and right motor values)
Evolution details	
Representation	Real valued vectors
Chromosome length	54
Mutation	Gaussian $N(0,1)$
Parent selection	Rank-based
Robot lifetime	2000 time-steps
Egg-phase	200 time-steps
Comm. range	ca. 9 body lengths

Table 1: Experimental set-up

Premium It is straightforward to (de-)emphasise particular tasks in MONEE by simply putting a premium on credits earned for that task. To investigate how premiums influence adaptation, we apply premiums ranging from -1 to 100 to the task of collecting green pucks, including a number of runs where the premium is redefined during the run. The premium for red pucks remains constant at 1.0.

During parent selection, the premium is used as a multiplication factor for the number of green pucks collected. Thus, with a premium set to -1, robots collecting green pucks are penalised. A premium of 0 means that there is no benefit to collecting green pucks: only red pucks are considered for parent selection. A premium of 1 means that red and green pucks contribute equally to the chance of a genome being selected, and higher values increase the importance of collecting green pucks.

Mutually exclusive skills Equitable task distribution is more challenging when the tasks that the robots must perform are to some extent exclusive, for instance because they require irreconcilable skills. To test how premium settings affect the MONEE paradigm in such situations, we also run experiments where the environment constrains multi-skilled robots so that the robots must specialise in collecting one type of puck. Without this constraint, robots can collect green and red pucks equally well without any penalty when selecting both or merely one colour. In the mono-skill experiments the speed of robots depends on their specialisation level: the robot's speed is multiplied by the ratio of most prevalent pucks it has collected. Thus, if a robot collects exclusively pucks of one colour, its speed is maximal. If it collects 75% green (or red) pucks, its speed is reduced by 25% and if it collects red and green pucks in equal amount, the speed is halved. This penalty is recalculated whenever a robot picks up a puck. It is important to note that this is enforced by the environment, not during the parent selection phase when an egg revives. The environment causes specialising robots to move faster, so that they perform better than non-specialised robots: their higher speed allows them

to collect more pucks during their lifetime, but more importantly, it allows them to impregnate more eggs. This results in an increase in the proliferation of mono-skilled genomes without altering the selection process inside the eggs.

Distribution of pucks Another determinant for the difficulty of task distribution in our experiments is the ratio of puck colours. This can be seen as a proxy for having a difficult (rare pucks) and an easy (common pucks) task. To determine the impact of setting premiums with an uneven distributions of pucks, we run two variants of our experiments: one with 150 pucks of each colour, one with 50 green and 150 red pucks. We perform additional runs with denser spreads of pucks where there are 500 pucks of each colour.

Changing premium A last set of experiments explores the evolutionary dynamics in the context of changing premiums. The rationale is the following: what would be the effect on the ratio of harvested puck colours if the premium is reset on-the-fly by a human supervisor? Then, what happens if the premium is changed back to its initial value after awhile? The system dynamics would be more predictable if the harvested pucks ratio matches the original figures, that is evolutionary dynamics always converge to the same ratio values, independent from the initial conditions. It may, however, be expected that the evolutionary dynamics are affected by the behaviour from where it already converged (i.e. the ratio depends from the actual premium *and* from where evolution starts). To explore the influence of changing premiums on-the-fly we use the following set-up: in a setting with the same number of red and green pucks (150 of each) the premium is initially set to 10. After 500,000 time steps, the premium is changed to 1 for 250,000 time steps. It is then reset to 10 for the remainder of the experiment.

Results and Analysis

The Effect of Premiums Figure 2 shows the mean total number of pucks collected in the experiments with 150 green and red pucks. Setting a negative premium predictably decreases the total number of pucks collected: the robots learn to avoid green pucks, in effect halving the number of available pucks. Setting the premium to 0 in the mono-skilled (i.e. with specialisation) environment still results in much lower levels of collected pucks because, again, the robots learn to keep away from green pucks and so avoid the environment's speed penalty for generalist behaviour. This penalty does not apply in the multi-skilled environment (i.e. without specialisation), and the number of pucks collected for premium 0 is markedly higher than with premium -1. The robots now pick up green pucks accidentally and they can take more direct paths to red pucks because they do not have to avoid green pucks. Setting the premium to 1 increases the number of pucks collected: robots now actively

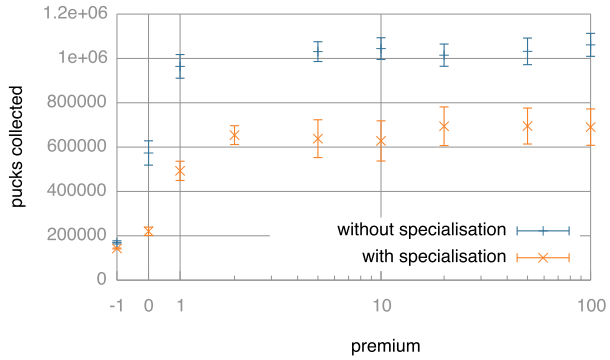

Figure 2: Mean total number of pucks collected over the whole run for different premium settings with 150 pucks of each colour. The vertical bars indicate the 99% confidence interval (over 64 repeats).

seek both types of puck. The mono-skilled environment still causes individual robots to avoid one type of puck or the other, therefore the number of pucks is lower than in a multi-skilled setting. Higher premium values slightly increase the number of pucks collected, but among these values it does not change appreciably.

To assess the impact of premium settings on the task distribution among the robot collective we consider the ratio of green pucks collected ('green puck ratio') over all collected pucks. Figure 3 shows how this ratio develops over time for different premium settings in the experiment with 150 pucks of each colour. Initially, the robot collective always gathers green and red pucks in a 50-50 ratio. With a premium of 1

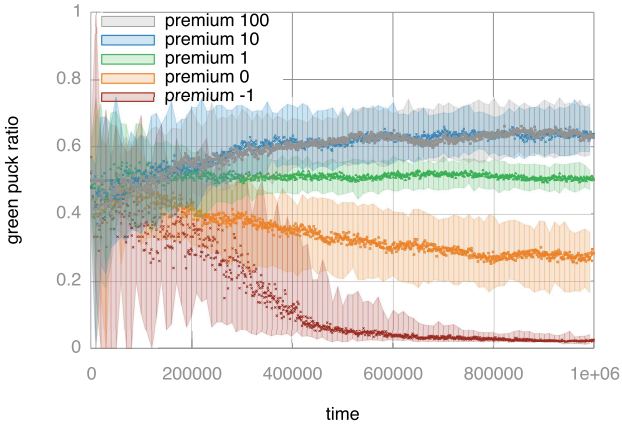

Figure 3: Development over time of the green puck ratio for a subset of the premium settings we considered. The points indicate the median green puck ratio for 1,000 time step intervals over 64 repeats, the shaded areas indicate lower and upper quartile.

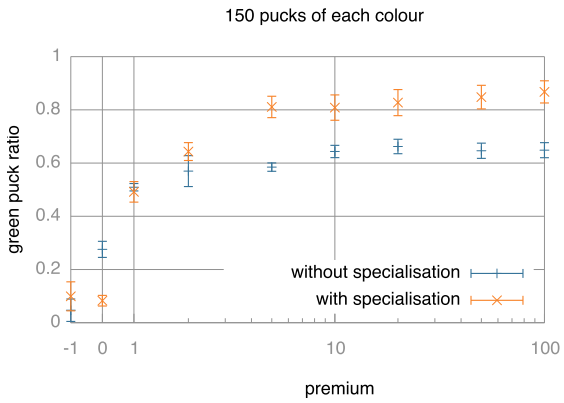

Figure 4: Mean green puck ratio in the final 1,000 time steps of runs with 150 pucks of each colour. The vertical bars indicate the 99% confidence interval over 64 repeats.

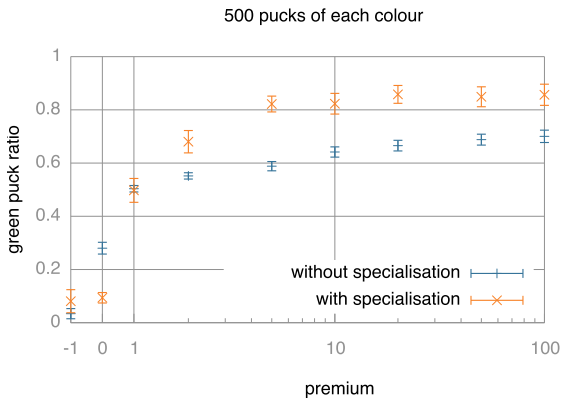

Figure 5: Mean green puck ratio in the final 1,000 time steps of runs with 500 pucks of each colour. The vertical bars indicate the 99% confidence interval over 64 repeats.

red and green pucks are equally valuable and the collective maintains this ratio. A premium of -1 has a profound impact: the robots learn to avoid green pucks and almost exclusively collect red pucks. This setting seems similar to the poisonous food experiment described by **?**, but the penalty for collecting 'poisonous' green pucks is effected during parent selection, not by the environment cutting short lifetime or reducing speed. A premium of 0 also leads to a substantial decrease in the green puck ratio: robots learn to focus on red pucks, but green pucks are not avoided and circa 30% of collected pucks is green. A premium of 10 increases the green puck ratio, which levels off around 0.65. The green puck ratio for premium 0 is in the same range as the red puck ratio for a premium of 10 (circa 0.3 and 0.35, respectively). This already indicates that larger premium values will do little to increase the green puck ratio (a premium of 0 for green pucks would have the same effect as a very high premium for red pucks).

This is borne out by the plot in Fig. 4, which shows the green puck ratio in the final 1,000 time steps of the simulation for varying premiums. We see that the green puck ratio among premium settings of 10 (or even 5) and higher barely changes. We also see that mono-skilled environments increase the impact of defining a premium: obviously, more robots will specialise in the higher rewarding task.

One reason for the lack of additional impact for higher premium values might lie in a saturation effect: if the robots simply cannot gather more green pucks than they do, the ratio can hardly improve. To test this hypothesis, we ran another set of experiments where there are 500 pucks of each colour. Figure 5 shows the results of those experiments. They show the same levelling off of premium impact, so it doesn't seem to result from a saturation effect.

Uneven Distribution of Pucks We use a setting where there are more red than green pucks (150 vs. 50) as a proxy for having easy and hard tasks. In these experiments, the 'natural' green puck ratio is 0.25, which is what we see in Fig. 6 when the premium is set to 1 in a multi-skill environment. When the environment discourages generalists, the ratio is slightly lower because robots tend to specialise in the simpler task (earlier work showed that MONEE's market mechanism plays a crucial role here (Haasdijk et al., 2013)). As was the case in the two scenarios where the puck distribution is balanced, increasing the premium past 5 or so has little further effect. The green puck ratio levels off between 0.3 and 0.4 for all premium values of 5 and greater.

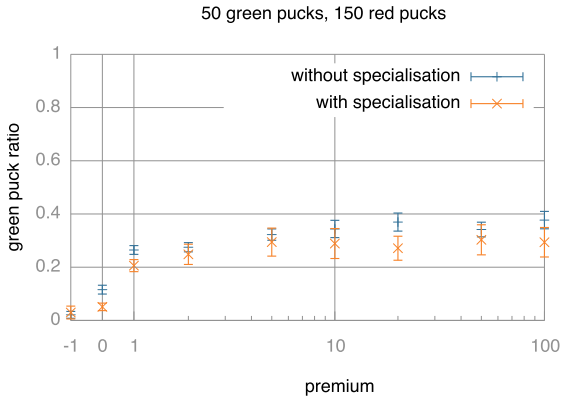

Figure 6: Mean green puck ratio in the final 1,000 time steps of runs with 50 green and 150 red pucks. The vertical bars indicate the 99% confidence interval over 64 repeats.

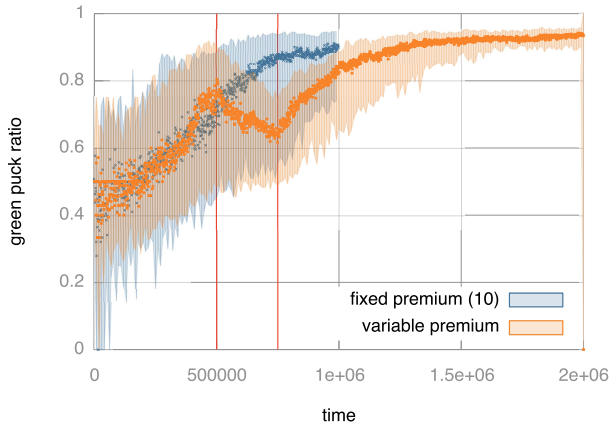

Figure 7: Development over time of the green puck ratio with changing premium with specialisation. The vertical red lines indicate when the premium is reset from 10 to 1 and back to 10. The points indicate the median green puck ratio for 1,000 time step intervals over 64 repeats, the shaded areas indicate lower and upper quartile. Green puck ratio for a constant premium of 10 shown for reference.

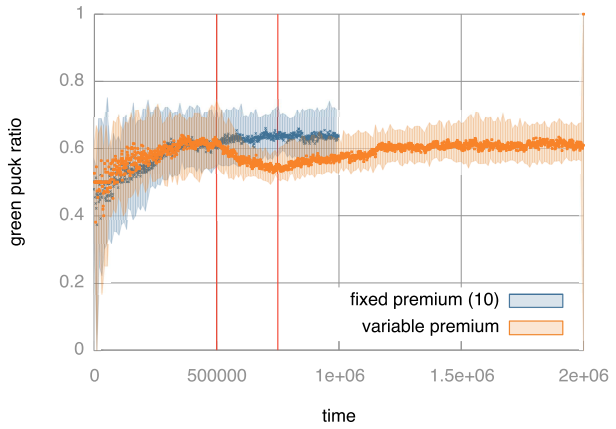

Figure 8: Development over time of the green puck ratio with changing premium without specialisation. The vertical red lines indicate when the premium is reset from 10 to 1 and back to 10. The points indicate the median green puck ratio for 1,000 time step intervals over 64 repeats, the shaded areas indicate lower and upper quartile. Green puck ratio for a constant premium of 10 shown for reference.

Varying Premium Settings Figures 7 and 8 show the puck ratio over time with changing premium with specialisation enforced and a multi-skilled setting, respectively. Note that we ran the experiment with changing premium for a further 1 million time steps to gauge long-term effects of changing premiums. With or without specialisation enforced, the green puck ratio initially develops unsurprisingly similar to the control experiment (using a constant premium of 10). Then, as soon as the premium is set to 1 after 500,000 time steps, the green puck ratio drops to reflect the new prioritisation of tasks. When the initial premium is restored at 750,000 time steps, with or without enforced specialisation, the green puck ratio starts to rise again quickly and levels off where it was before the premium was changed. Hence, there is no memory effect when we change the premium in the course of a run, advocating for stable attractors that depend only from the premium value at hand. The change in puck ratio as a reaction to varying the premium is considerably more pronounced in the single-skill setting than when robots can collect both types of puck.

Conclusions and Future Work

Experimental results on the effects of premium settings with the MONEE algorithm showed that setting premium values stand as an efficient mechanism to allow the user to control the prioritisation of tasks. On the one hand, setting negative premiums dramatically decreases the take-up of tasks. One the other hand, positive premiums enable to promote tasks, at least to some extent. Indeed, the relation between premiums and task distribution is not linear as the influence of increasing premium values is dampened after an environment-dependent threshold.

In the particular case of foraging with two kind of pucks, further experiments showed that controlling the evolution of a particular foraging behaviour is sensitive to the distribution of resources. Enforcing specialisation (i.e. penalising robots that forage both resources) can greatly increase controllability whenever both resources are available in equal amount, while dramatically decreasing controllability whenever an uneven distribution of resources is considered.

Lastly, controllability was also tested from the perspective of on-line tuning, i.e. changing premium values during the course of evolution to match user requests. Results revealed that premium values actually matched very stable attractors towards (expected) foraging behaviours.

Although the work presented here shows that collective foraging behaviour can be controlled to some extent through setting premium values, the non-linear (and thresholded) relation between premium values and task distribution remains to be further explored. We are indeed currently investigating the thresholding of the premium effect. Also, we are addressing the problem how to actually use premiums to automatically achieve a particular state of task distribution. To some extent, this is an inverse problem: while the desired task distribution may be known before hand, the method for tuning the premium values may well depend on the environment and the task at hand.

Acknowledgements

The authors would like to thank Jean-Marc Montanier, Berend Weel, A.E. Eiben and Nikita Noskov for many inspi-

rational discussions on the topics presented here. We thank SARA Computing and Networking Services (www.sara.nl) for their support in using the Lisa Compute Cluster.

References

Beyer, H.-G. and Schwefel, H.-P. (2002). Evolution strategies – A comprehensive introduction. *Natural Computing*, 1:3–52.

Bredeche, N., Montanier, J.-M., Liu, W., and Winfield, A. F. (2012). Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1):101–129.

Burtsev, M., Red'ko, V., and Gusarev, R. (2001). Model of evolutionary emergence of purposeful adaptive behavior. the role of motivation. In Kelemen, J. and Sosík, P., editors, *ECAL*, volume 2159 of *Lecture Notes in Computer Science*, pages 413–416. Springer.

Haasdijk, E., Weel, B., and Eiben, A. (2013). Right on the monee. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2013)*. To Appear.

Jones, C. and Mataric, M. (2003). Adaptive division of labor in large-scale minimalist multi-robot systems. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1969 – 1974.

Langton, C., editor (1995). *Artificial Life: an Overview*. MIT Press, Cambridge, MA.

Langton, C. G. (1989). *Artificial Life: Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Mascaro, S., Korb, K., and Nicholson, A. (2005). An alife investigation on the origins of dimorphic parental investments. In Abbass, H. A., Bossomaier, T., and Wiles, J., editors, *Advances in Natural Computation, Proceedings of the Australian Conference on Artificial Life (ACAL 2005)*, volume 3, pages 171–185.

Menczer, F. and Belew, R. (1996). Latent energy environments. In *Santa Fe Institute Studies In The Sciences Of Complexity-Proceedings Volume-*, volume 26, pages 191–210.

Menczer, F., Willuhn, W., and Belew, R. (1994). An endogenous fitness paradigm for adaptive information agents. In *CIKM Workshop on Intelligent Information Agents*. Citeseer.

Nguyen, Q., Nguyen, X., O'Neill, M., and Agapitos, A. (2012). An investigation of fitness sharing with semantic and syntactic distance metrics. In Moraglio, A., Silva, S., Krawiec, K., Machado, P., and Cotta, C., editors, *Genetic Programming*, volume 7244 of *Lecture Notes in Computer Science*, pages 109–120. Springer Berlin Heidelberg.

Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, Cambridge, MA.

Scheutz, M. and Schermerhorn, P. (2005). Predicting population dynamics and evolutionary trajectories based on performance evaluations in alife simulations. In Beyer, H.-G. and O'Reilly, U.-M., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2005)*, pages 35–42. ACM, ACM.

Schwarzer, C., Hösler, C., and Michiels, N. (2010). Artificial sexuality and reproduction of robot organisms. In Levi, P. and Kernbach, S., editors, *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*, pages 384–403. Springer–Verlag, Berlin–Heidelberg–New York.

Tang, F. and Parker, L. (2007). A complete methodology for generating multi-robot task solutions using asymtre-d and market-based task allocation. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3351 –3358.

Ventrella, J. (2005). Genepool: Exploring the interaction between natural selection and sexual selection. *Artificial Life Models in Software*, pages 81–96.

Walsh, W. and Wellman, M. (1998). A market protocol for decentralized task allocation. In *Multi Agent Systems, 1998. Proceedings. International Conference on*, pages 325 –332.

Watson, R. A., Ficici, S. G., and Pollack, J. B. (2002). Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39(1):1–18.

Wischmann, S., Stamm, K., and Wörgötter, F. (2007). Embodied evolution and learning: The neglected timing of maturation. In Almeida e Costa, F., editor, *Advances in Artificial Life: 9th European Conference on Artificial Life*, volume 4648 of *Lecture Notes in Artificial Intelligence*, pages 284–293. Springer-Verlag, Lisbon, Portugal.