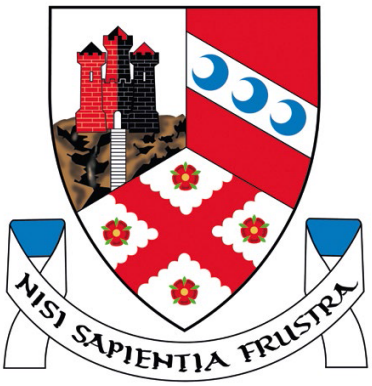
A Closer Look at Adaptation Mechanisms in Simulated Environment-Driven Evolutionary Swarm Robotics



Andreas Siegfried Wilhelm Steyven

A thesis submitted in partial fulfilment of the requirements of

Edinburgh Napier University, for the award of

Doctor of Philosophy

October 2017

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

External examiner Prof. Susan Stepney

Internal examiner Dr. Simon T. Powers

Director of studies Prof. Emma Hart

Second supervisor Prof. Ben Paechter

Andreas Siegfried Wilhelm Steyven

October 2017

Abstract

This thesis investigates several aspects of environment-driven adaptation in simulated evolutionary swarm robotics. It is centred around a specific algorithm for distributed embodied evolution called mEDEA.

Firstly, mEDEA is extended with an explicit relative fitness measure while still maintaining the distributed nature of the algorithm. Two ways of using the relative fitness are investigated: influencing the spreading of genomes and performing an explicit genome selection. Both methods lead to an improvement in the swarm's ability to maintain energy over longer periods.

Secondly, a communication energy model is derived and introduced into the simulator to investigate the influence of accounting for the costs of communication in the distributed evolutionary algorithm where communication is a key component.

Thirdly, a method is introduced that relates environmental conditions to a measure of the swarm's behaviour in a 3-dimensional map to study the environment's influence on the emergence of behaviours at the individual and swarm level. Interesting regions for further experimentation are identified in which algorithm specific characteristics show effect and can be explored.

Finally, a novel individual learning method is developed and used to investigate how the most effective balance between evolutionary and lifetime-adaptation mechanisms is influenced by aspects of the environment a swarm operates in. The results show a clear link between the effectiveness of different adaptation mechanisms and environmental conditions, specifically the rate of change and the availability of learning opportunities.

Acknowledgements

I would like to express my sincere gratitude to Prof. Emma Hart for the continuous support, patience and motivation. Your guidance has made this possible. Also to Prof. Ben Paechter for his many inputs during thought-provoking scientific discussions. To my partner Simone, firstly for proof-reading this thesis, but most of all for all your continuous encouragement, support, patience and understanding. You have made my life so much richer. To the few close friends I can always count on, thank you for always being there for me. You have provided a balance between work and life that has kept me sane. You have made this journey a very enjoyable one. Many thanks to all the different people who have played a role in my life during the writing of this thesis. You have contributed in one way or another to me getting here.

And last but not least, my parents and my entire family. Mein aller herzlichster Dank gilt meinen Eltern und meiner gesamten Familie. Nicht nur für die moralische und finanzielle Unterstützung, sondern dafür das Ihr immer an mich geglaubt habt. Ihr habt es möglich gemacht.

— A.S.W.S.

Für Oma und Simone!

Table of contents

List of figures xii

List of tables xvii

1 Introduction 1

1.1 Research Questions . . . 3

1.2 Contributions . . . 4

1.3 Methodology . . . 6

1.4 Publications . . . 6

1.5 Thesis Overview . . . 7

2 Background 9

2.1 Robotics . . . 9

2.2 Evolutionary Robotics . . . 10

2.2.1 Swarm Robotics . . . 11

2.2.2 Controller . . . 12

2.3 From Centralised and Offline to Distributed and Online . . . 14

2.4 Algorithms for Distributed Online Evolution . . . 16

2.4.1 Probabilistic Gene Transfer Algorithm . . . 16

2.4.2 odNEAT . . . 16

2.4.3 mEDEA . . . 17

2.5 Drivers of Evolution . . . 19

2.5.1 Fitness Function 19
2.5.2 Environment-driven Adaptation 20
2.5.3 Environment-driven + Fitness Function 21
2.6 Role of the Environment 22
2.6.1 Morphology 23
2.6.2 Influence of the Environment 25
2.7 Combining Learning and Evolution 26
2.7.1 Learning and Adaptation Mechanisms 27
2.7.2 Individual and Social Learning 28
2.8 Summary 29

3 Using Relative Fitness to Improve Survivability in mEDEA 31
3.1 Contribution 31
3.2 Introduction 32
3.3 The mEDEA Algorithm in Detail 33
3.3.1 Algorithm Outline 33
3.3.2 A Closer Look at the Key Steps 35
3.4 mEDEA with Relative Fitness 37
3.4.1 Explicit Selection Mechanisms 38
3.4.2 Biasing Broadcasting of Genomes 38
3.5 Hypotheses 40
3.6 Experiments 42
3.6.1 Methodology 43
3.6.2 Experiment Set 1: Explicit Selection Mechanism 44
3.6.3 Experiment Set 2: Varying the Broadcasting Mechanism . . . 45
3.7 Evaluation and Analysis 46
3.7.1 Experiment Set 1: Explicit Selection Mechanism 46
3.7.2 Experiment Set 2: Varying the Broadcasting Mechanism . . . 48

3.7.3 Combining Explicit Selection with Biased Broadcasting . . . 52
3.8 Summary and Conclusion . . . 54
3.8.1 Summary . . . 54
3.8.2 Conclusion . . . 55

4 Influence of Communication Cost on the Effectiveness of the Broadcasting Variation Mechanism in mEDEA$_{rf}$ 57

4.1 Contribution . . . 58
4.2 Introduction . . . 58
4.3 The Use of Energy in mEDEA$_{rf}$ . . . 59
4.4 Improving the Energy Model in the Simulator . . . 60
4.4.1 Energy Consumption Characteristics of Hardware Communication Modules . . . 61
4.4.2 Free-Space-Model . . . 62
4.4.3 Calculation of Communication Costs for the Simulation . . . 64
4.4.4 Movement Dependent Living Costs . . . 67
4.4.5 The New Energy Model . . . 68
4.5 Hypothesis . . . 68
4.6 Experiments . . . 68
4.6.1 Methodology . . . 69
4.7 Evaluation and Analysis . . . 70
4.7.1 Methodology . . . 70
4.7.2 Influence of the Energy Model . . . 71
4.8 Summary and Conclusion . . . 79

5 Influence of the Environment on the Emergence of Behaviour 81

5.1 Contribution . . . 82
5.2 Introduction . . . 82
5.3 Hypotheses . . . 83

5.4 Experiments . . . 83
5.4.1 Methodology . . . 84
5.4.2 Adaptation of mEDEA$_{rf}$ for Experiments . . . 85
5.5 Evaluation and Analysis . . . 86
5.5.1 Different Performance Regions . . . 87
5.5.2 Environmental Influence on Behaviour . . . 90
5.5.3 Behaviours in the Neutral Region . . . 91
5.5.4 Results of a Different Environment . . . 92
5.6 Summary and Conclusion . . . 95

6 Influence of the Environment on the Benefit of Lifetime Adaptation 96
6.1 Contribution . . . 96
6.2 Introduction . . . 97
6.3 Designing the Lifetime Adaptation Mechanism . . . 98
6.3.1 Scenario Overview . . . 98
6.3.2 The Individual Learning Mechanism . . . 99
6.4 Hypotheses . . . 102
6.5 Experiments . . . 102
6.5.1 Variations of the Adaptation Mechanism . . . 103
6.5.2 Selecting Environmental Configurations . . . 103
6.5.3 Sets of Experiments . . . 104
6.5.4 Energy Model and Updated Communication Costs . . . 106
6.6 Evaluation and Analysis . . . 107
6.6.1 Methodology . . . 107
6.6.2 Adaptation Mechanism . . . 108
6.6.3 General Observations . . . 111
6.6.4 Influence of the Adaptation Mechanism . . . 113
6.6.5 Influence of Environmental Parameters . . . 117

6.6.6 Influence of Environmental Change 117
6.6.7 Analysis 119
6.7 Summary and Conclusion 119

7 Conclusion **121**
7.1 Summary 121
7.2 Answers to Research Questions 122
7.3 Discussion 126
7.4 Future Work 127

References **129**

Appendix A Publications **139**
A.1 Improving Survivability in Environment-driven Distributed Evolutionary Algorithms through Explicit Relative Fitness and Fitness Proportionate Communication 139
A.2 The Cost of Communication: Environmental Pressure and Survivability in mEDEA 148
A.3 Understanding Environmental Influence in an Open-Ended Evolutionary Algorithm 151
A.4 An Investigation of Environmental Influence on the Benefits of Adaptation Mechanisms in Evolutionary Swarm Robotics 162

Appendix B Adaptation Mechanism **171**
B.1 Visualisation of Individual Learning Parameter 171

List of figures

2.1 Emergent Behaviour 23

3.1 Robot in environment with energy token 35

3.2 Explicit selection added to the mEDEA algorithm. Figures show the energy for experiments E1, E1+t, E1+rw, E1+b. 47

3.3 Explicit selection added to the mEDEA algorithm. Figures shows the number of active robots for experiments E1, E1+t, E1+rw, E1+b. . . . 48

3.4 mEDEA, control experiments and biased broadcasting: figures show the energy for each of the experiments E1-E5. 49

3.5 mEDEA, control experiments and biased broadcasting: figures show the number of active robots for each of the experiments E1-E5. 50

3.6 mEDEA, control experiments and biased broadcasting: figures show the number of genomes received for each of the experiments E1-E5. . 51

3.7 Combining the biased broadcasting of genomes with explicit selection by individuals. 52

3.8 Comparison between vanilla mEDEA, mEDEA with explicit selection by individuals and $mEDEA_{rf}$ with explicit selection by individuals. . 53

4.1 Comparison of the median number of received broadcasts per unique genome between experiments, without (E*x*,E*x*+rw) and with the communication costs (E*x*-em, E*x*-em+rw). Violin and box plots showing the distribution of values over the last three generations of the experiment. Lines are smoothed to better show trends, with the raw data displayed in the same colour and thinner lines. 74

4.2 Comparison of the median value for lifetime per unique received genome at the end of a generation between experiments, without (E*x*,E*x*+rw) and with the communication costs (Ex_{em}, Ex_{em}+rw). Violin and box plots showing the distribution of values over the last three generations of the experiment. Lines are smoothed to better show trends, with the raw data displayed in the same colour and thinner lines. . . . 75

4.3 Comparison of the median energy level at the end of a generation between experiments, without (E*x*,E*x*+rw) and with the communication costs (Ex_{em}, Ex_{em}+rw). Violin and box plots showing the distribution of values over the last three generations of the experiment. Lines are smoothed to better show trends, with the raw data displayed in the same colour and thinner lines. 76

4.4 Comparison of the median number of active robots at the end of a generation between experiments, without (E*x*,E*x*+rw) and with the communication costs (Ex_{em}, Ex_{em}+rw). Violin and box plots showing the distribution of values over the last three generations of the experiment. Lines are smoothed to better show trends, with the raw data displayed in the same colour and thinner lines. 77

4.5 Comparison of the median value for collected tokens at the end of a generation between experiments, without (Ex,Ex+rw) and with the communication costs (Ex_{em}, Ex_{em}+rw). Violin and box plots showing the distribution of values over the last three generations of the experiment. Lines are smoothed to better show trends, with the raw data displayed in the same colour and thinner lines. 78

5.1 View on the resulting surface from different angles. The figure was created by plotting the median δ_E of the last 2500 iterations of the experiment. The grey plain marks a value for δ_E of zero, at which point robots in an experiment have an energy balance of zero. In other words, the same amount of energy as they started the experiment with. An interactive 3D model can be found at [119] 87

5.2 Overview of the landscape (zero contours from 5.1), as plot of the real data on the left and as a cartoon version on the right. Four different regions are shown: A) Dead Zone, B) Lean Valley, C) Neutral Line, D) Excess Energy. 88

5.3 Cuts through different parts of the landscape. Points towards different behaviours in terms of exploration. a-b) $value = 1150$, vary $count$; c-d) $count = value$; e-f) $count = 1150$, vary $value$. 91

5.4 Cuts through the surface landscape of experiments with 100 robots in environments that differ in availability and concentration of energy, and contain equal amounts of positive and negative tokens. Environments are defined by amount (*count*) and *value* per token, with a constant value of -400 for negative tokens. The coloured lines show the Neutral Line, the line where the surface plot crosses a plain drawn at delta-energy (δ_E)=0, averaged over 2500 iterations at different stages of the experimental run. Red, green and blue ending at iteration 125,000, 250,000 and 375,000 respectively. The three crosses, located just off the line, mark the identified environments of the previous experiments. The crosses, located on top of the line, mark the environmental configurations where the extension of the *equi-value*, *equi-count* and *count=value* lines cross the Neutral Line. 94

6.1 Top down view of the surface plot created using the method introduced in the previous chapter. The red line shows the Neutral Line, the line where the surface plot crosses a plane drawn at delta-energy (δ_E)=0. . 105

6.2 Adaptation mechanism and performance values for Evo + IL in three different environments, defined by the quantity (**c**) and value (**v**) of energy tokens available in a *static* setting (a-c) at a seasonal change (**s**) of 5k (d - f), 15k (g - i) and 30k (j - l). These intention of these figures is to demonstrate the general trends and patterns and not the accurate representation of data. Hence, certain elements, such as legends and axis descriptions, have been omitted. The reader is referred to figure 6.3 for a more detailed version of sub figure (j) where the omitted details of the graphs are shown. 109

6.3 Detailed sample figure for adaptation mechanism and performance values in figure 6.2. 110

6.4 Normalised difference between positive and negative tokens collected. Solid line is the value combined over all seasons, dashed = season 0, dotted = season 1. 113

List of tables

3.1 mEDEAs evolutionary algorithm parameters for all experiments in this thesis. 43

3.2 Simulation and experimental parameters for all experiments in this chapter. 44

3.3 p-values obtained from applying Wilcoxon's Rank-Sum test across pairs of experiments, including biased broadcast only and biased broadcasting coupled with an explicit selection method. 53

4.1 Values taken from the data-sheet for wireless communication module Texas Instruments CC2420 [125]. 64

4.2 Size of a data message (transmitted genome) in byte. 65

4.3 Values for communication costs used in simulator. 67

4.4 Simulation and experimental parameters for all experiments in this chapter. 70

4.5 Showing median of end values for *energy*, *active*, *broadcasts : genome*, *age : genome* and *token collected* over the last three generations of the experiment. The symbols $\downarrow$, $\leftrightarrow$, $\uparrow$ indicate how the shown values compare to the result of the corresponding experiment without the energy model, with the number of arrows indicating the magnitude level of the effect size, based on a Vargha and Delaney A test (1 = small, 2 = medium, 3 = large). 72

5.1 Simulation and experimental parameters for all experiments in this chapter. 84
5.2 Communication cost parameters adjusted to reflect change in genome length. 86
5.3 The environmental configurations sought out for all future experiments. 92
5.4 Results obtained at three configurations within the neutral region. . . 92
5.5 Environmental configurations: description refers to the prevalence of energy tokens within the environment. 93

6.1 Simulation and experimental parameters for all experiments in this chapter. 102
6.2 Learning parameters with initial values and ranges in which they can change during runtime of the experiment. Method of adaptation (through evolution or lifetime-learning) depends on the experiment. 103
6.3 Learning scenarios investigated showing heritability of information. . 104
6.4 Environmental configurations: description refers to the prevalence of energy tokens within the environment. 104
6.5 Communication cost parameters. 107
6.6 Showing median of end values by seasonal change and the experiment for *totalTokenRatio* over the final 5000 iterations (N:30). 116
6.7 Showing p-values of pairwise comparisons of the learning mechanism for *totalTokenRatio* (row vs. column) over the final 5000 iterations. . . 116
6.8 Showing p-values of pairwise comparisons of environments for *totalTokenRatio* (row vs. column) over the final 5000 iterations. 118
6.9 Showing p-values of pairwise comparisons of seasonal change for *totalTokenRatio* (row vs. column) over the final 5000 iterations. . . . 118

Chapter 1

Introduction

Thanks to advances in science and technology machines are now closer to what science fiction authors have promised us for decades. Companies, like DeepMind[1] and Boston Dynamics[2], demonstrate great showmanship when promoting their achievements and have brought advances in AI and robotics to the public's attention. Robots in the form of vacuum cleaners and lawn mowers have taken over household chores and made their way into everyday life. In manufacturing, industrial robots have replaced humans for difficult, repetitive or dangerous tasks, such as those that expose the worker to hazardous substances, extreme temperatures, or require heavy lifting, while maintaining continuous accuracy, to name but a few. Robotic systems can also operate in hazardous environments or remote locations, such as inside nuclear contaminated areas, in outer space, at the bottom of the ocean, or in nano scale in the human body.

Centuries before the use of industrial robots in automation, humanity was fascinated by automata, the self-operating machines that appeared to be working under their own command. These earlier versions of autonomous machinery tried to mimic nature by using clockwork or cams to act out pre-defined programs to produce natural-looking movements and behaviours [134]. The benefits of taking inspiration from nature have been recognised in many fields of science and engineering. For example, the way

[1] https://deepmind.com, accessed 23/10/2017
[2] https://www.boston-dynamics.com, accessed 23/10/2017

feathers at the tip of a bird's wings break up turbulence has inspired the design of more efficient aeroplane wings, and water repellent surfaces have been fabricated by recreating the nano structure of a lotus plant's leaves.

The field of computer science has taken inspiration from natural systems and processes many times. Most famously the neural network, a computational model based on an abstracted model of the nervous system, which forms the basis of the current successes of deep-learning [106]. Optimisation algorithms and heuristics, like ant-colony optimisation [76] and genetic and evolutionary algorithms, have been developed to tackle computationally hard problems [37]. Creating and tuning the control architecture of a robot is an example of a computationally hard problem. In the field of evolutionary robotics, evolutionary algorithms are used to evolve the controller, most commonly in the form of a neural network [85].

Rather than relying on a single robot, which also constitutes a single point of failure, multiple and potentially simpler robots can be used. Using multiple robots allows the workload to be distributed across the swarm, and multiple tasks can be worked on simultaneously [3]. Swarm robotics is a special case in robotics and another case where nature has inspired research. In a swarm, the global behaviour emerges from the local interactions of the simple individuals that make up the swarm. The distributed nature makes it more resilient to failure with no single point of failure. When considering deployment of robots in remote locations, the ability to autonomously adapt is a crucial feature, especially in scenarios with *a priori* unknown environmental conditions and very limited or no ability to communicate with a remote controller [5]. Hence, adaptation needs to happen right there and then, during runtime and without supervision.

Evolutionary algorithms fit the bill quite nicely. Often referred to as *black-box optimisers* they are well-suited to optimise when the location of the optimal solution in the search space is unknown [31]. Bredeche and Montanier [19] shifted the focus to the environment, which constitutes a driving force for the evolutionary process in this

case. Many researchers have followed by investigating several aspects of systems based around environment-driven evolution, such as the selection pressure [40], balancing survivability and task performance [95], or the dynamics of the evolutionary process itself by analysing phylogenetic trees [14]. However, many more questions in this area remain to be tackled. Considering that the environment is one of the key drivers in the evolutionary process, a line of research naturally follows to investigate its implications.

The research presented here shows how the selection pressure stemming from the environment can be focused to emphasise certain attributes of the swarm, as well as how the environment's influence on the emerging behaviours in different algorithmic settings.

1.1 Research Questions

The following research questions address the gaps identified through the literature review. In this context performance means a robot's ability to maintain energy and stay alive for an extended amount of time. In particular, the work extends an existing environment-driven algorithm called mEDEA [19].

RQ 1 *To what extent can adding an explicit relative fitness to mEDEA (creating $mEDEA_{rf}$) increase the maintained energy across the swarm in comparison to the existing algorithm without explicit fitness?*

RQ 2 *How does accounting for the cost of communication influence the performance of an evolutionary algorithm that relies on communication?*

RQ 3 *To what extent does the specific parametrisation of the environment influence the emergence of behaviour in $mEDEA_{rf}$?*

RQ 4 *To what extent does the environment influence the most appropriate combination of evolutionary and lifetime-adaptation mechanisms in $mEDEA_{rf}$?*

1.2 Contributions

This thesis makes a number of contributions to the field evolutionary robotics by investigating several aspects of environment-driven adaptation in simulated evolutionary swarm robotics and mEDEA in particular.

`A novel relative fitness function`:

The mEDEA algorithm from [19] is augmented with an explicit fitness measure (chapter 3, section 3.4). This novel way of calculating a relative fitness function maintains the algorithm's distributed nature while providing increased selection pressure.

`Two ways of applying the relative fitness`:

Two ways of using the relative fitness are investigated: firstly, influencing the spread of genomes (section 3.6.3) and, thereby, indirectly increasing the selection pressure; secondly, moving from a random to an explicit selection of genomes based on the relative fitness (section 3.6.2). It is shown that both methods lead to an improvement over the original algorithm in the swarm's ability to maintain energy over longer periods.

`Explicit communication energy model`:

An energy model for communication is derived from the field of wireless sensor networks research that accounts for communication costs in the robot simulation (chapter 4, section 4.4). This cost is very important in small battery powered robots but is usually missing in simulation environments. The experiments carried out show that accounting for the costs of communication exerts additional environmental pressure.

`Novel methodology to explore environmental influence on performance`:

A novel methodology to investigate the environment's influence on the emergence of behaviours at the individual and swarm level is introduced (chapter 5, section

5.2). A 3-dimensional map is constructed that relates an environmental configuration c, defined by parameters (x, y) to the retained median energy value of the robot population operating in c. This method can be generalised to any setting and therefore constitutes a useful contribution to anyone working in evolutionary robotics.

`Analysis of two different experimental settings`:

Using the methodology above for the development of performance maps in two different settings, one with only positive items in the environment, and another with positive and negative items. The map is used to identify interesting regions for future experimentation in two different experimental settings (section 5.4 and 5.5.4). This is in contrast to ad-hoc methods currently used in most publications for parameter setting. These regions provide challenging environments for the robots while still allowing algorithm specific characteristics to show effect and be explored.

`A novel individual learning mechanism`:

An individual learning mechanism is developed that doesn't require any *a priori* knowledge of the environment and can help the evolutionary process to adapt (chapter 6, section 6.3).

`Analysis of environmental influence on the usefulness of different adaptation mechanisms`

The analysis of how environmental parameters influence the best performing combination of evolutionary and lifetime-adaptation mechanisms in mEDEA$_{rf}$ (section 6.5). The results show that there is a clear link between environmental conditions, specifically the rate of change and the availability of learning opportunities, and the effectiveness of different adaptation mechanisms. Further, the incentive to learn in form of reward or punishment, in other words, the environ-

mental pressure which drives the evolutionary process must be high enough to make any form of adaptation worthwhile.

1.3 Methodology

This thesis takes a quantitative research approach. Experiments are carried out in simulation using Roborobo [22], the simulator that was used in the original publication of mEDEA [19] and many subsequent publications (e.g. [95, 16, 40]). To this date, the majority of experiments in the field use simulation [83], which allows for the easy collection of data for analysis. Statistical tests are used to determine the results of experiments throughout. For the purpose of testing for statistical significance, multiple independent runs of the same experiment have been performed. The exact number for each experiment is given in the corresponding methodology section. All analyses have been performed using the open-source statistical package R.

1.4 Publications

The content of this thesis has been published in the following peer-reviewed publications:

Hart, E., Steyven, A., & Paechter, B. (2015). Improving Survivability in Environment-driven Distributed Evolutionary Algorithms through Explicit Relative Fitness and Fitness Proportionate Communication. In S. Silva (Ed.), *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15* (pp. 169–176). New York, New York, USA: ACM Press.

A copy of the publication can be found in the appendix (A.1).

Steyven, A., Hart, E., & Paechter, B. (2015). The Cost of Communication: Environmental Pressure and Survivability in mEDEA. In S. Silva (Ed.), *Proceedings of*

the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference - GECCO Companion '15 (pp. 1239–1240). New York, New York, USA: ACM Press.

A copy of the publication can be found in the appendix (A.2).

Steyven, A., Hart, E., & Paechter, B. (2016). Understanding Environmental Influence in an Open-Ended Evolutionary Algorithm. In J. Handl, E. Hart, P. R. Lewis, M. López-Ibáñez, G. Ochoa, & B. Paechter (Eds.), *Parallel Problem Solving from Nature – PPSN XIV* (pp. 921–931). Springer International Publishing AG.

A copy of the publication can be found in the appendix (A.3).

Steyven, A., Hart, E., & Paechter, B. (2017). An Investigation of Environmental Influence on the Benefits of Adaptation Mechanisms in Evolutionary Swarm Robotics. In G. Ochoa (Ed.), *Proceedings of the 2017 on Genetic and Evolutionary Computation Conference - GECCO '17* (pp. 155–162). New York, New York, USA: ACM Press.

A copy of the publication can be found in the appendix (A.4).

1.5 Thesis Overview

The remainder of this thesis is structured as follows:

- Chapter 2 discusses the current state of the field and helps further define the scope of the research by grounding it in the literature.
- Chapter 3 covers the extension of mEDEA with a relative fitness function to create mEDEA$_{rf}$, and explores two ways of applying the new relative fitness measure.

- Chapter 4 describes the development of a communication energy model and evaluates the influence of accounting for the costs of communication in mEDEA and the newly extended version with relative fitness (mEDEA$_{rf}$).

- Chapter 5 describes a novel method of exploring the environmental influence on the performance of mEDEA$_{rf}$ and applies it to identify interesting regions for future investigation in two experimental settings.

- Chapter 6 introduces a novel individual learning method and goes on to investigate how the most appropriate balance between evolutionary and lifetime-adaptation mechanisms is influenced by aspects of the environment a swarm operates in.

- Chapter 7 summarises the contributions of this thesis to the research field, provides answers to the posed research questions and outlines how the work presented here can be extended in the future.

Chapter 2

Background

2.1 Robotics

The word robot originates from the Czech word *robota* meaning 'forced labour' [96]. The term was coined in 1920 by K. Čapek in his play "Rossum's Universal Robots". Robotics covers a variety of things, ranging from manufacturing machines to planetary exploration, and one day nano-scale robots may even circle the human bloodstream. Nowadays, most robots follow hard wired behaviour: preprogrammed rules that enable the machine to cope with any scenario imagined by the designer of the robot system. Winfield [134, p. 11] illustrates this with a robot vacuum cleaner that has only five preset rules to fulfil its duties. However, if the encountered environment is dynamic or can't be predicted at design time, e.g. under water or in space exploration, these predefined rules can be insufficient. Furthermore, human intervention is typically limited in said environments, meaning the robot system must be capable of acting and adapting autonomously to sustain the anticipated behaviour.

Autonomous robots are controlled by a controller, which makes decisions based on sensory inputs that are usually the sole source of data available to the robot [42]. Factors, such as a changed environment, a faulty sensor in the robot or the transfer of the control program from simulation into hardware, can make a difference in the data

available to the controller. The challenge in adaptable autonomous robots now relies on developing systems capable of coping with all these uncertainties.

2.2 Evolutionary Robotics

The term evolutionary robotics (ER) has been phrased by Cliff et al. [27] and has been defined as "*Evolutionary Robotics aims to apply evolutionary computation techniques to evolve the overall design or controllers, or both, for real and simulated autonomous robots*" [128].

The field is at the intersection of artificial life (ALife) and robotics engineering. ALife tries to explore and understand processes observed in nature. ER has been used to model evolutionary dynamics, such as the evolution of specialisation [6] and cooperation [7], altruism [129] and self-organising behaviours [126]. The engineering side employs nature-inspired concepts to create novel solutions that human engineers have yet to come up with [42]. Bongard [13] envisions the long term goal of ER to be a mechanism for creating algorithms that can create autonomous adaptive robot systems.

Evolutionary algorithms (EA) work on population levels where good solutions evolve over generations [37]. EAs typically use an application specific objective function to determine the fitness of a candidate solution. In ER the evolved candidate solution can be the robot's morphology, controller, or both [72]. The quality of such solutions is represented by their expressed behaviour over time when the phenotype interacts with the world [88].

To this date evolutionary robotics experiments have been carried out in terrestrial, swimming and flying robots [33, 44, 56], as well as a large volume of work in simulation [73, 83].

2.2.1 Swarm Robotics

This thesis focuses on a specialised approach called swarm robotics (SR), which takes inspiration from swarms of insects as found in nature. The swarm's behaviour and ability to perform certain tasks emerges out of the interaction of the simple and quasi identical[1] agents that act asynchronously due to the lack of a central control [5]. Swarm robotics uses large numbers of autonomous and situated robots with local sensing and communication capabilities [15]. Şahin [103] defined swarm robotics as *"the study of how large numbers of relatively simple physically embodied agents can be designed such that a desired collective behaviour emerges from the local interactions among agents and between the agents and the environment"* .

With the inspiration for this approach taken from nature, it comes as no surprise that SR is often used as a tool to research concepts from nature, such as exploring biological or ecological models in the area of artificial life [78]. In more engineering like applications, a swarm of robots offers certain advantages over the use of a single more complex robot. Due to the large number of robots the workload can be distributed across the swarm and multiple tasks simultaneously worked on in parallel [3]. It further offers distributed sensing capabilities and an increased robustness to failure as demonstrated in the Swarmanoid project [32]. However, these advantages aren't without limits as the costs of coordination grow with swarm size. Reliability and robustness to failure have been shown to quickly fall with increasing numbers of robots [10]. These limitations add to the complexity of the task to create controllers for the swarm that lead to the emergence of the desired behaviour.

The swarm's behaviour is often optimised using evolutionary algorithms. To name a few, researchers have successfully evolved a swarm's ability to adapt to unknown environments [127, 17], its resilience to failure [77] and the planning and following of formation patterns [104, 32].

[1] meaning either identical hardware or similar in actuation and sensing capability as well as appearance as required by the application

2.2.2 Controller

A robot's controller issues control signals to the actuators based on information that is available through sensory inputs and the internal state of the robot. Its complexity can vary from a simple reactive mechanism with fixed rules [102], over sophisticated programs that develop over time to adapt to the current operating conditions the robot faces [28], to multi-layered control systems such as subsumption architecture [23].

A controller can be optimised using an evolutionary algorithm, a versatile black box optimiser that can evolve a huge variety of potential controller types. Doncieux et al. [30] note that in order to not constrain evolution, the control paradigm should be able to take in raw sensor inputs and output low-level actuator commands. Hence, the most widely used control paradigm in evolutionary swarm robotics is a graph-based controller in the form of a neural network (NN), which draws inspiration from a simple neuron model of the brain [69].

Neural networks can be implemented in many different forms and levels of complexity. A NN is a graph of neurons, often structured in layers, from an input layer to an output layer, with none, one or more hidden layers inbetween (hidden, as they have no connections to the outside world) [65]. Each neuron sums all inputs attached to it and uses an activation function to generate an output. The synapses, the connections between neurons, have a weight associated with them that changes the relative importance of the information flowing through it. The simplest form of which is the straight feed-forward NN, which represents a purely reactive controller. A network is referred to as straight feed-forward if all connections within the network only point in one direction, namely from the input layer to the output layer. The network is stateless, meaning in each computational cycle of the network only the currently available information at the input layer is used to produce the output signal. Memory can be added to the network by using recurrent connections [114], where internal loops allow the output signal of a neuron to travel in the direction of an input of the same or a previous layer. The network

is now no longer stateless as the previously generated signal will recur as an input in the next computational cycle of the network.

Networks can be fully or partially connected. Unless the structure of the network is evolved, most NN in ER applications use fully connected networks. A widely used algorithm to evolve the networks structure is Neuro Evolution of Augmenting Topologies (NEAT) [117] and its variants, e.g. HyperNEAT [116]. These algorithms remove the need to fix the architecture but rather let them adapt to the complexity level required [46, 47, 33].

If sufficient training data is available, a neural network can be trained using a supervised method, such as back-propagation, where the difference between the expected output and the actual output is used to gradually reduce the error. However, when using NNs as low level controllers, as is often the case in evolutionary swarm robotics, it becomes difficult to obtain labelled data. It would be easy to record all inputs to the network during a robot's operation, however, determining the appropriate low level action for each set of inputs and labelling the data accordingly would be a very difficult and labour intensive task. In most evolutionary robotics applications a different approach is used. The weights of the neural network are evolved. This unsupervised adaptation mechanism allows the NN to be tuned without *a priori* knowledge of the input data.

Evolving the weights of a NN comes with some restrictions. Mutation is often used as the sole variation operator. This is necessary due to competing conventions [41] among different solutions in the gene pool. Individual solutions have different internal conventions to express similar behaviours. Applying crossover might replicate or eliminate functionality altogether. Algorithms such as NEAT [117] apply a special technique to ensure that the crossover will lead to functional solutions. However, this method requires global knowledge of the gene pool, which might not be given if the EA is decentralised and spread across the swarm.

Although not used in this thesis, many other encoding methods, representations and algorithms have been explored in the evolutionary robotics community. For example,

direct-encoded methods that are not based on neural networks, e.g. Q-Learning [100], GP trees [75]. Furthermore, attention has been given to indirect encodings, where the final structure of the network is the result of a developmental or generative process. The most prominent examples here are cellular encoding [68], L-Systems [63], gene-regulatory networks (GRN) [28] and Hyper-cube based NEAT (HyperNEAT) [116]. These methods have all been described in scenarios in which the introduced approach is superior in comparison to other methods. However, as each method caters to a specific problem and the demonstrated performance is very scenario dependent, it is impossible to determine an overall best.

2.3 From Centralised and Offline to Distributed and Online

An evolutionary algorithm in evolutionary robotics can be deployed in many forms. Following the taxonomy by Eiben *et al.* [35], the aspects *when*, *where* and *how* the EA is run can vary.

When: At runtime (online) [111] or during design time (offline) [46] of the robot.

Where: Inside (on-board/intrinsic) [112] or outside (off-board/extrinsic) of the robot's body [43].

And, when employing the evolutionary algorithm in a swarm of robots:

How: Distributed across the swarm (decentralised) [132] or centralised [130].

The criteria for *when* is the point in time when the individual fitness evaluations and evolutionary operators are executed.

The concept of *how* refers on one hand to the location of the individual genomes of the gene pool, and on the other hand to the algorithm itself [132]. In a classic EA

both components, the gene pool with all genomes and the algorithm, are in the same place. Especially in the field of evolutionary robotics either of these components can be distributed. Each robot can carry one genome of the pool, and the algorithm can be designed in a way that requires the interaction of two or more members of the swarm to be executed. If both criteria are met the algorithm is referred to as distributed. Of course, mixed variations can also be imagined, e.g. where the algorithm is distributed and each individual robot carries a sub-population of the gene pool, much like an island in a traditional EA [17].

The concept of *where* refers to the physical location in which the evolutionary operators (selection, variation and replacement) are executed [35]. However, it does not refer to the location in which the fitness evaluations take place. In the extrinsic or off-board approach, the computation is outsourced from the actual robot body to an external entity. This might be due to constraints in computational power available on small mobile robots or the availability of a computationally more powerful external device, e.g. specialised processing hardware. If all the evolutionary operators are executed inside the robot's body it is referred to as intrinsic or on-board.

In ER it is not explicit enough to classify algorithms by where the computation happens alone. The crucial aspect of where the fitness evaluations are performed is missing. Many researchers, such as Bongard [12] and Stepney [118] and most recently Bredeche *et al.* [18], argue the concept of embodiment. This classification does not contradict the former, but rather shifts the focus onto where the fitness evaluation takes place:

`Embodied`: The fitness evaluation is performed inside the body, hence the EA is immersed into the environment. It can directly sense the environmental impact on the robot and has direct access to the feedback from the robot's actions. This approach implies the use of on-board and online evolution [14].

`Disembodied`: The fitness evaluation is performed outside the body and the EA has no direct access to the world through the eyes of the robot. This approach might also make use of variables that are available to an external observer.

Section 2.6.2 will discuss the interaction between embodiment and the evolutionary process in more detail.

Which form the evolutionary algorithm takes depends on the extrinsic constraints of the application. For example, in this thesis where the focus is on continuous autonomous adaptation in a swarm of robots, the algorithm needs to be distributed and online, and use embodied fitness evaluations.

2.4 Algorithms for Distributed Online Evolution

2.4.1 Probabilistic Gene Transfer Algorithm

Watson *et al.* [133] proposed a completely decentralised algorithm for embodied evolution (EE). In their Probabilistic Gene Transfer Algorithm (PGTA), robots exchange randomly selected genes through short range communication. This algorithm doesn't have a dedicated variation and replacement steps. Each robot holds a single genome of which only individual genes are replaced at runtime. The fact that all genomes are active makes this evaluation inherently parallel. Transmission frequency and gene acceptance are based on the explicit fitness value of the respective robot, which reflects its performance on a task. PGTA has successfully been applied to a photo-taxis task, and they found that it produced better performing solutions than their best hand-designed solutions.

2.4.2 odNEAT

Following the success of NEAT [117], Silva *et al.* [112] extended the algorithm for use in a decentralised multi-robot application. They evolved the structure and synaptic

weights of the neural network, thus adjusting the level of complexity needing to be specified a priori. The algorithm requires synchronisation of the internal clocks to allow for the use of timestamps. Similar to NEAT, odNEAT uses timestamps within the genotypes to mark the addition of elements to the network to allow crossover. To date, it has been used in simulation and real robot [109] experiments with a number of tasks, e.g. obstacle avoidance, aggregation, maze navigation and homing [111]. It has been demonstrated that controllers across the swarm displayed a diverse range of behaviours and strategies for exploration and aggregation, as well as the ability to adapt to periodic changes in task requirements.

2.4.3 mEDEA

The completely distributed evolutionary algorithm for environment-driven evolution, *mEDEA*, was first proposed in Bredeche and Montanier [19] and has been used [21, 14, 82, 80] and adapted [54, 95, 40, 49] many times. The algorithm was demonstrated to be both efficient with regard to providing distributed evolutionary adaptation in unknown environments, and robust to unpredicted changes in the environment.

mEDEA relies on an implicit fitness function that results from two potentially conflicting motivations for a robot: an *extrinsic* motivation to cope with environmental constraints in order to maximise survival ability and an *intrinsic* motivation to spread its genome across the population in order to survive. A complex trade-off exists in which behaviours that maximise mating opportunities might negatively impact survival efficiency, e.g. failing to maintain stable energy levels; as a result, mEDEA (or an environment-driven EA) must find some equilibrium between the two states.

The original version of mEDEA exploited a simple strategy in which a robot continuously broadcasts its genome — this can be received and stored by any robot within communication range. At the end of a generation, each robot makes a *random* selection from its set of stored genomes, applies a mutation operator, and then replaces

its current genome, exactly as in a (1,1) Evolution Strategy [8]. Although there is no selection pressure on an individual basis, from a global perspective, the most widely spread genomes will, on average, be selected more often. While this achieved success in evolving stable populations, it is of interest to attempt to improve both the size of the swarms maintained and their net energy levels to allow for more complex user-defined tasks to be added in future. It is reasonable to assume that spare energy, over and above that required to survive, can be exploited to achieve complex tasks, while a large swarm offers more potential in terms of the tasks that might be accomplished.

Within mEDEA, the evolutionary mechanism differs from natural evolution that also drives adaptation. All robots broadcast their genome continuously and within a fixed range, each robot has equal opportunity to pass on its genome, regardless of its quality. This might create a bias for agents to stay close in order to spread their genome. The reproduction mechanism is asexual, as can be found in bacteria, which uses no form or crossover — variation is provided only by a mutation operator — hence the emphasis is on the spreading of *genomes* rather than *genes*.

When mEDEA was first proposed in [19] the system was tested under two scenarios: the first evaluated mEDEA in an environment providing limited pressure in which energy is ignored and an agent survives as long as it collects at least one genome. In the second, environmental pressure is introduced by forcing robots to compete for limited resources in order to gain energy. The algorithm was demonstrated to be both efficient with regards to providing distributed evolutionary adaptation in unknown environments, and robust to unpredicted changes in the environment.

Furthermore, given its lightweight nature, mEDEA was demonstrated to be suitable for hardware and software setups that have limited computation. Bredeche *et al.* [21] implemented mEDEA on a set of E-Puck robots with Linux extension boards to investigate the emergence of consensus amongst the swarm. This was achieved by placing an object in the environment and giving robots a virtual sensor for the object's location. As there was no benefit or penalty for interacting with the object, the

controllers were free to evolve to ignore the object or hone in on it. They found that besides some technical difficulties, the algorithm was able to create a stable population and in most cases consensus amongst the swarm.

Montanier *et al.* [82] used mEDEA as a platform to conduct an investigation in the field of ALife by studying the tragedy of the commons. The experiments demonstrated that the algorithm naturally evolves altruistic agents with an increasing tendency to greedy behaviour when it doesn't impact the population's chance of survival. The altruistic agents expressed behaviours that did not harvest the total possible energy ration amount from the available sources. They concluded that environment-driven algorithms produce a trade-off between survival and task performance, rather than the optimum of either of those criteria.

2.5 Drivers of Evolution

When employing evolutionary optimisation, one crucial aspect is the driving force which guides the process. Intrinsically, it is the mechanism that determines the quality criteria for solution and therefore the selection pressure. Recently, researchers have started focusing on identifying and quantifying selection pressure in algorithms used in evolutionary robotics [31, 51].

This section will explore the two most common drivers in evolutionary robotics: the explicit fitness function and environment-driven evolution.

2.5.1 Fitness Function

Artificial evolution is guided by an objective, a fitness function that needs to be optimised to solve a predefined task. In evolutionary robotics this is often referred to as goal- or task-driven evolution [4]. Other objectives can be chosen to determine the fitness, such as novelty or diversity of solutions [31], however, in this thesis the focus is on fitness functions for explicitly defined tasks. Common tasks in evolutionary swarm robotics are

collaborative manipulation, such as object transportation and assembly, group foraging, path planning, coordinated motion and collective decision making.

Nelson *et al.* [88] gave a broad overview of fitness functions in evolutionary robotics. They found that it is better to incorporate not too much knowledge into the fitness function as this would create a bias towards a known solution and would limit the ability of evolution to surprise. The survey focused mainly on *a priori* knowledge that had been incorporated into the fitness function. Doncieux and Mouret [31] extended this overview and introduced categorisation into task-specific and task-agnostic functions.

Defining an explicit fitness function can give a clear definition of the goal robots should achieve. However, when employing a distributed online adaptation algorithm the evaluation becomes noisy and it can be challenging to get precise fitness evaluations [81]. As previously discussed, the evaluation of a genome, i.e. the controller, takes time in evolutionary robotics. The emergent behaviour needs to be observed for long enough to get a clear picture of the quality of the controller. When robots operate autonomously they can't be repositioned which leads to different starting points for each genome as the individual's evaluation begins in the location of the robot's body left by the previous controller. Hence, a genome might not be able to display a potentially useful behaviour. For example, if stuck in a bad starting position it might not be able to move at all. This makes the fitness evaluation noisy and leads to potentially longer times to find a good solution.

2.5.2 Environment-driven Adaptation

Future environments in which swarm robots operate will often be unknown and potentially dynamic, e.g. swarms of robots being sent to remote or hazardous places. This has led to a number of recent efforts to study evolution within a swarm as a mechanism for driving adaptation, as opposed to a mechanism for optimising an explicit fitness function, which is common in much work within evolutionary robotics. Montanier [80]

notes that this step is in fact a pre-requisite to studying any kind of user-driven task behaviours within a robotic swarm in an open-environment, as the former cannot be achieved if the integrity of the swarm is compromised.

This type of evolution is often referred to as *environment-driven evolution* [9]. Typical approaches, such as [112], remove the need for any central control, resulting in algorithms that perform distributed and online evolution. An additional feature of environment-driven algorithms is that no explicit fitness function is defined: instead, mate selection and reproduction depend on selection pressure provided only by the environment, yet need to lead to stable populations. The mEDEA algorithm as described above (2.4.3) is another example.

2.5.3 Environment-driven + Fitness Function

Both of the previously introduced drivers of evolution can be combined. Noskov *et al* [95] extended mEDEA so that in addition to surviving and operating reliably in an environment, a robot could also perform user-defined tasks without compromising its ability to survive. Survivor selection is driven by the environment, whereas parent selection is driven by task performance. Their new framework MONEE (Multi-Objective aNd open-Ended Evolution algorithm) shows that task-driven behaviour can be promoted without compromising environmental adaptation. Robots accumulate credit for accomplishing particular tasks — this credit value is transmitted along with a genome, and is utilised in a fitness function to select parents, replacing the random selection seen in mEDEA.

Fernández *et al.* [40] augmented mEDEA with an explicit fitness function to study the impact of four different selection methods on the selection pressure. Two different tasks were performed, namely obstacle avoidance and collective foraging. For obstacle avoidance Nolfi and Floreano's [94] fitness was used, which maximises the distance travelled and minimises the collision with objects. They found that higher selection pressure results in improved performances, especially with more challenging tasks.

Boumanza [14] examined phylogenetic trees for embodied evolutionary robotics and selected mEDEA as a representative algorithm for the study. The algorithm was augmented in a similar fashion to Fernández *et al.* as mentioned above, using roulette-wheel selection to select from the received genomes. Through variation of parameters they investigated the effect of different selection pressures on the properties of the resulting phylogenetic tree. They demonstrated how the creation of phylogenetic trees and analysis of the resulting graph gives useful insights into the evolutionary dynamics of the algorithm.

Although not part of this thesis, there is a current trend towards algorithms like Novelty search [70], or so called quality and diversity (QD) algorithms, such as MapElites [84]. Novelty search still has an explicit selection criteria, but rather than focusing on optimising task performance it selects based on the novelty of behaviours. MapElites helps build up a repertoire of known best solutions in different parts of the search space that might become relevant at a later stage, e.g. when the objective shifts due to external influences.

2.6 Role of the Environment

In the previous section it has been shown that the environment can be used as driver for the evolutionary process. This section will focus on why it is important to take the environment into account and the ways in which it influences the evolutionary process.

Whether an explicit fitness function is used to shape the solution or a robot's simple need to stay alive, the environment in which they are immersed in is always a factor. The so called *sensory-motor interaction* describes the relationship between the state of the environment that is sensed by a robot and the changes its actuators inflict on that state [91]. In other words, the robot's controller uses the on-board sensors to gather information about its surroundings. The controller issues commands based on this information to its actuators, which leads to an interaction with the environment. The

consequences of these actions will influence the information that is now sensed. Even in a completely static environment, moving just a fraction will already have changed the perception the controller has of its surrounding world.

The robot's behaviour emerges as a result of the interplay between the current robot configuration, the phenotype in the vocabulary of the evolutionary algorithm and the environment. This is true whether the controller, the morphology or both are produced by the evolutionary process. Figure 2.1 illustrates this interaction and the dynamic processes within the controller and environment.

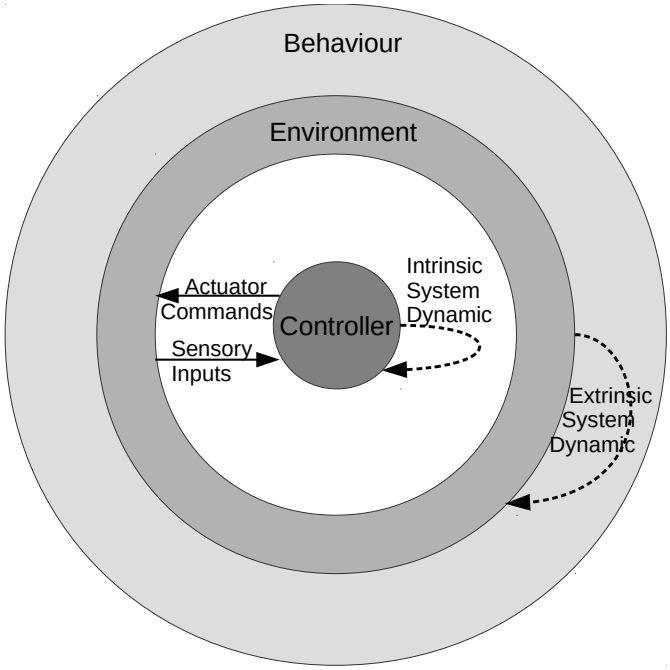


Fig. 2.1 A robot's behaviour emerges from its interaction with the environment. The schematic (based on [91]) illustrates the interaction relationship between a robot's controller, body and the environment. Solid arrows indicate the direct interaction with the environment. Dotted lines indicate dynamic processes within the controller and the environment, which also influence the resulting behaviour.

2.6.1 Morphology

The morphology of a robot plays a special role in evolutionary robotics. It hosts the robot's controller which is therefore "grounded" in the environment due to its embodied nature. As a result, the emerging behaviour is a consequence of the interaction with the

environment [91]. It comes therefore as no surprise that different concepts for shapes and forms have been explored. An evolutionary algorithm can be used to optimise all aspects of the robot's body [13], from the shape and size to the material composition, from traditional solid mechanical objects to smart-materials for the production of soft-robotics [101, 25] to self-assembling robots from many smaller units [48].

Karl Sims' [113] pioneering work on evolving *virtual creatures* in simulation set a milestone for the, back then, quite young field. In his experiments, both the morphology and the controlling mechanisms were produced by the evolutionary process. Pfeifer and Bongard later strongly argued the necessity to evolve body and controller at the same time [97] to achieve more complex and sophisticated machines.

Pollack and Lipson demonstrated the advantage of evolving all parts of the robot at the same time over designing each part separately. In their *Golem project* [98] they introduced an automated process to automatically manufacture the entire robot body shell to guide evolution running in simulation. Limited by the technology of their time, electronic components still needed to be attached by hand. New approaches, such as the 3D printing of different materials and electronics at the same time, could lead to a completely autonomous process [34].

It is now a well established paradigm that evolving morphology and controller together leads to better engineered robots [12]. However, when it comes to swarm robotics, the focus shifts from the individual robot to the formation and size of the swarm of simple agents [103, 15].

The research conducted for this thesis focuses on the evolution of robot controllers in a fixed morphology within a swarm of robots. Although all interactions of the robot occur through its body, the morphology is not subject to any change by the evolutionary process. It will certainly influence the resulting performance, however, due to its static nature, the influence will remain the same throughout all experiments and can merely be seen as a constant offset.

2.6.2 Influence of the Environment

The emerging behaviours arising from the interactions of the robot with its environment are not well understood, perhaps in part due to the time-consuming experimentation that needs to be done to conduct sweeps of the parameters that define the environment. It is common in optimisation to explore the relationship between algorithmic parameters and fitness [67]. However, evolutionary robotics adds an additional dimension in that it is not only the algorithm's parameters that change but also the environmental parameters. Hence, it is crucial to investigate the influence the environment has on the performance of the algorithm.

Auerbach and Bongard [1] investigated the relationship of environmental and morphological complexity in evolved robots. Much like in the works of Sims [113] or Pollack and Lipson [98], the morphology and controller were co-evolved. The study focused on a single robot at a time rather than a swarm. These robots were evaluated in 49 different environments in which evenly spaced low friction bars on high friction ground varied in height and distance, thus creating a variety of challenges. They analysed the difference in evolved complexity of the individuals that were able to successfully manoeuvre said environments in a fixed amount of time. They found a direct relationship between the complexity of the environment and the evolved complexity, leading to the hypothesis that a gradual increase in complexity of the problem domain will result in more complex individuals.

Bredeche *et al.* [19] demonstrate their algorithm's robustness by introducing a drastic environmental change in their experiment. This presented the algorithm with a different type of environment and required a change in behaviour in order to survive. Although this experiment evaluated their algorithm in two different environmental settings, it would be very hard to try to generalise a performance prediction from this.

Haasdijk *et al.* [52] did a thorough analysis of the impact the algorithmic parameters have on performance in a set environment. Their approach was limited to tuning the

parameters in order to achieve good task performance across four different problems. However, in this work, as in others, experimental parameters, e.g. the ones defining the environment, are chosen arbitrarily and the use of actual benchmarking is rare [30]. This is often down to the lack of well defined test-bed applications and metrics [15].

2.7 Combining Learning and Evolution

Evolution and lifetime learning are two adaptation mechanisms that are often combined in evolutionary robotics [124, 38, 62, 93]. Learning happens within an individual's lifetime and benefits the individual directly, whereas evolution gradually produces better adapted individuals over a much longer time-scale of generations. In ER, the entirety of a robot controller's lifetime corresponds to a single fitness evaluation of the EA. Learning can make use of this evaluation time by improving upon the evolved behaviour.

Unfortunately, in literature it is not often clear which type of adaptation is actually employed. The redefinition of concepts leads to blurred lines, for example, when evolutionary concepts are used in the time-scale usually reserved for learning, and a distributed evolutionary algorithm is described as social learning [59].

The interaction between learning and evolution is described by the *Baldwin Effect* [2]. It describes how the ability to learn can create an evolutionary advantage. A considerable body of research exists around the interaction of learning and evolution [124, 38, 62, 93]. Individuals that can adapt through learning during their lifetime have higher reproductive success compared to those who can't or take longer to learn. While the *Baldwin Effect* can be used to explain the interaction between learning and evolution, which is investigated in chapter 6, the focus is on the interaction of the adaptation mechanisms with the environment, rather than the effect itself.

2.7.1 Learning and Adaptation Mechanisms

Many different methods and approaches of learning and adaptation have been introduced in evolutionary robotics literature. Many different nature-inspired approaches have been used to introduce plasticity into a neural network, which then allows for a change of behaviour during runtime.

Noble and Franks [90] demonstrated reinforcement learning in a multi-agent system. For learning, a variety of social learning strategies were facilitated using Q-Learning as the underlying mechanism to enable agents to learn not just from direct but also delayed reward signals. Hebbian Learning [127] is an adaptation mechanism in which learning rules determine how the weight of a synapse is adjusted, based on the activity of the neuron located before, after or on either side of it. The evolved adaptation rules for synapses are then used to adjust a randomly initialised network during runtime. This approach was evaluated in different environments, in simulation and real robots. It was demonstrated to outperform evolved fixed-weight NN in a light-switching task. In Neuromodulation [115], additional modulation neurons are added to the network that can individually strengthen or suppress the activity of neurons. This allows the runtime adaptation of the network's behaviour in parts or as a whole. Neuromodulation augmented odNEAT outperformed vanilla odNEAT in a simulated concurrent foraging task in a robot swarm by generating stable controllers in significantly fewer generations [110]. Artificial hormones are applied in a similar fashion in a Neuro-Endocrine System [86] to amplify or suppress neuron activity. The sensitivities to different hormones can vary for each neuron, leading to a range of behavioural patterns created by the same NN depending on the current hormone concentration in the system. Potentially competing demands can be managed by switching between behaviours gradually [105]. The Artificial Immune System [24] employs a nature inspired approach that allows the build up of a behavioural repertoire in which immune cells are used to recognise

and map sensory inputs to actions. Learning occurs over time through stimulation and suppression of different actions, based on their success.

2.7.2 Individual and Social Learning

As indicated in the section about swarm robotics (2.2.1), in systems with multiple robots the learning effort can be shared. The effort of learning can be shared and benefit the learning individual as well as others in the swarm by sharing the acquired information. This can be achieved through observing other robots [39] or explicitly sharing information [50]. Having multiple entities learning at the same time can also help to verify and improve the accuracy of learned concepts [90]. The learning concept employed in the swarm can then be classed as either individual, learning on your own, or social-learning from others [50].

Haasdjik *et al* [53] propose a framework for evolution, individual and social learning in collective systems, and consider the interaction of evolution and individual learning in which the latter is achieved by *reinforcement learning* [123]. Their experiments show that in a collective system it is possible for learning to counteract evolution. A *hiding-effect* can occur in which individual learning acts to mask the ill-adapted nature of non-optimal agents, and is therefore counter-productive. This can be explained with the Baldwin Effect. Although a number of environments were investigated that essentially modified the reward system, all environments were static, and the relationship of the learning framework to specific parametrisations of the environmental features was not examined.

Heinermann *et al* investigate the relationship between evolution, individual and social learning in a real swarm [58–60]. Here, the evolutionary part focuses on evolving a suitable sensory layout, while the individual learning runs an evolution strategy-like mechanism to learn the network weights during the robot's lifetime. Learnt weight vectors are broadcast to other robots during the social learning phase. The main focus

of this work was to investigate the impact of social learning. Individual learning is *required* to learn a controller and hence cannot be omitted.

A dynamically changing reward system was investigated by Bredeche and Montanier [19] who proposed mEDEA, a completely distributed evolutionary algorithm for environment-driven evolution. Here, efficient adaptation in a changing environment was demonstrated using a set-up that switched phases: in the *free-ride* phase there is no cost to movement, therefore a robot only needs to meet a single other robot to pass on its genome, while in the alternating phase the robot is required to harvest energy in order to move and therefore create opportunities for passing on its genome. Noskov *et al* extended mEDEA to add explicit task-selection in the MONEE framework [95]. Haasdijk *et al* [49] extended this work and examined in more detail the relative selection pressures induced by task performance and survival in different environments, finding that task performance is optimised even if it reduces the lifetime of robots (and therefore their ability to reproduce).

2.8 Summary

Evolutionary algorithms are widely used black-box optimisers for solving computationally hard problems. In evolutionary robotics they are used to adapt the controller of robots in a swarm operating in potentially unknown and changing environments. The evaluation of a potential solution, a genome of the evolutionary algorithm, requires a robot to operate in the environment to determine its fitness. This can be very time consuming. Operating a swarm of robots has the advantage that the evolutionary algorithm can be distributed across the swarm, hence creating the opportunity to evaluate solutions in parallel. The environment-driven distributed evolutionary adaptation algorithm mEDEA has been identified as a suitable approach as it focuses on survivability of the swarm, rather than the optimisation of a specific task performance. However, the algorithm appears to offer scope for improvement. The clever use of a fitness function

could increase the selection pressure for the evolutionary process while still maintaining the algorithm's distributed nature. In addition, the environment has been identified to influence the path the evolutionary process takes. Many researchers argue for the concept of embodiment and the influence the environment has when the situated agent undergoes evolutionary adaptation. However, no clear methodology has been found to determine the role the environment plays and how it influences an environment-driven evolutionary adaptation algorithm.

Further, there are additional types of adaptation mechanisms, e.g. different forms of individual or social learning. The link and influence between evolution and learning have widely been studied in the Baldwin effect. However, in the context of evolutionary robotics where different mechanisms can simply be used to get the best possible adaptation, the focus shifts away from explaining nature, to merely taking inspiration. This then poses questions, such as under which conditions are those methods beneficial, and how – if at all – this is linked to the role of the environment. The remainder of this thesis will address the issues raised by answering the questions posed in chapter 1.

Chapter 3

Using Relative Fitness to Improve Survivability in mEDEA

In many natural systems an individual's chance of reproduction is related to their fitness relative to other individuals in its vicinity. Additionally, individuals mate selectively, choosing partners based on some estimation of their quality and/or strength. Some species broadcast their quality through visual or behavioural displays: a peacock displays its tail feathers; a bird of paradise 'dances'. Fitter individuals can attract the attention of a greater number of potential mates. Inspiration can be taken from nature to implement a similar relative fitness mechanism to improve survival while maintaining the integrity of a fully distributed algorithm.

3.1 Contribution

Elements of the work described in this chapter have been published in the proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2015) [55]. A copy of the publication can be found in the appendix (A.1)

The mEDEA algorithm, an environment-driven distributed adaptation algorithm for swarm robotics, is augmented with an explicit fitness measure. This novel way of

calculating a relative fitness maintains the algorithm's distributed nature. Two ways of using the relative fitness are investigated: influencing the spreading of genomes, thereby indirectly increasing the selection pressure; and moving from a random to an explicit selection of genomes based on the relative fitness. It is shown that both methods lead to an improvement over the original algorithm in the swarm's ability to maintain energy over longer periods.

3.2 Introduction

The original mEDEA algorithm, as introduced by Bredeche and Montanier [19], uses an implicit fitness function combined with an indirect selection mechanism that promotes a higher number of unique robot encounters. Here the mEDEA algorithm is augmented with a relative fitness function that creates selection pressure for the individual robot towards behaviours to maintain energy. This has been identified as a pre-requisite to studying any kind of user-driven task behaviours within the robotic swarm, as the former cannot be achieved if the integrity of the swarm is compromised.

Inspired by nature, a relative fitness measure is introduced into mEDEA. A robot makes an estimate of its fitness to maintain energy relative to those within its broadcast range, thereby maintaining the distributed nature of the algorithm. The thus obtained fitness value can be used in two ways:

- An individual robot can make an informed rather than random selection from the genomes it has received, according to the relative fitness value.
- The relative fitness value can be used to influence the broadcasting behaviour of a robot to provide a bias towards the spread of good genomes.

The latter point changes the nature of the reproductive strategy used in mEDEA from a 'promiscuous' one, in which there is indiscriminate broadcasting of a genome, to one in which the *spread* of a genome (and therefore the probability of it being collected) is

dependent on its quality. Two novel methods for influencing broadcasting are introduced: the first causes the robot to adapt the probability with which it broadcasts, based on its fitness, and the second causes the robot to adapt the range over which it broadcasts its genome. As in the original version of mEDEA, robots still make a random selection from collected genomes. However, due to the biased broadcasting methods, on average, good quality genomes are more likely to be collected than poorer ones.

3.3 The mEDEA Algorithm in Detail

The general concept of mEDEA has been introduced in the previous chapter (2.4.3). Here, the algorithm is described in more technical detail with a special focus on the key steps in the selection mechanism of the evolutionary algorithm. The original mEDEA algorithm is defined in algorithm 1.

In order to map the standard terminology of an evolutionary algorithm to the terms used in this thesis about swarm robotics, the physical (or in this case simulated) body of the robot is merely an evaluation vessel for genomes. An individual is an instantiated genome, meaning it is currently being used to define the controller of a robot and therefore determining its behaviour. The *population* refers to the unique genomes held by the swarm of robots, as currently active genomes and those within the list of received genomes. This is similar to the gene's eye view of evolution by Dawkins as used with his *selfish gene* metaphor [29].

3.3.1 Algorithm Outline

For a fixed number of time steps (or iterations) of one generation, robots move according to their control algorithm, which is a neural network (a multi-layered perceptron, MLP). The currently active genome is used to define the neural network controller weights. Upon an encounter with another robot, the currently active genomes are transmitted and both robots keep an exact copy of the other's genome in their local genome storage.

Therefore, multiple copies of a genome can exist among all robots' genome storages. At the end of a generation, when it comes to selecting a new genome, the current active one is discarded and a replacement randomly selected from the local storage. At this point the local genome storage is cleared and all encountered genomes removed. In the case of a robot not picking up any other genomes, it becomes inactive. It removes its current genome and remains stationary until a passing robot comes into communication range and passes its genome. Being the only genome in the local storage, it is immediately selected as a replacement. The selected genome undergoes variation before being instantiated, meaning it is becoming the current active genome. This variation takes the form of a Gaussian random mutation operator, inspired from Evolution Strategies [8] that can easily be tuned through a σ parameter.

At the start of each generation, a robot is initialised with an energy E_0.

$$E(t_0) = E_0 \tag{3.1}$$

Every time step, the energy value is decreased by a 'living' cost of $E_{\text{living}} = 1$ unit. Energy *tokens* are scattered in the environment 3.1. If a robot moves over a token, its energy is increased by an amount E_{token}. Equation 3.2 shows the change in energy at each time step, where n is the number of tokens that have been collected in that step. Here, t_0 refers to the time the current genome is activated.

$$E(t+1) = E(t) - E_{\text{living}} + (n \times E_{\text{token}}) \tag{3.2}$$

A robot with no energy ($E(t) = 0$) becomes inactive and remains stationary for the remainder of the generation. To investigate any effects that the necessity of robots to find energy tokens has on their survival, the parameters E_0 and E_{token} have to be selected to be less than the amount required to survive a full generation. This ensures

that there is a selection pressure that originates from the need to locate tokens, as well as spreading the genomes.

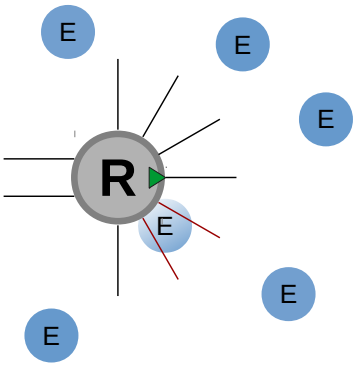


Fig. 3.1 Robot (R) with energy tokens (E) distributed across the environment. The robot's front is indicated by the green arrow and the sensor rays are illustrated as black lines around the circular robot body. The energy token closest to the robot is being detected (red coloured sensor rays) and now being consumed (as indicated by the fading blue colour) as soon as the outline of the robot body touches the token surface.

3.3.2 A Closer Look at the Key Steps

As previously mentioned, there are two ways a robot can become inactive: by running out of energy or not having collected any genomes in the previous generation. Therefore, the population size of the evolutionary algorithm is dynamic and at any point in time equal to the number of unique genomes across the robots' collected genome lists and currently active genomes.

This has an impact on the genetic diversity within the population as well. A harsh environment, caused by a drastic change or initial environmental conditions, can lead to a steep decline in population size. The original experiment from Bredeche and Montanier shows that mEDEA is able to recover from such drops in population size. The availability of energy is the controllable factor for harshness in the environments considered for the experiments. Due to randomly initialised genomes, and therefore neural network weights, the resulting behaviour of robots is random movement. Hence, running over an energy token in a scarcely populated environment is merely down to

```
1  load(currentGenome = randomInitialisedGenome);
2  while iteration ≤ maxIterations do
3      if hasGenome() then
4          if lifetime ≤ maxLifetime & energy > 0 then
5              move();
6              broadcast(currentGenome);
7          else
8              remove(currentGenome);
9          end
10     end
11     genomeList.addIfUnique(receivedGenome);
12     if genomeList.size() > 0 then
13         genome = select<random>(genomeList);
14         load(currentGenome = applyVariation(genome) );
15         genomeList.empty();
16         lifetime = 0;
17     end
18 end
```

Algorithm 1: Pseudo code of the original mEDEA algorithm by Bredeche *et al.* [20] executed within every robot.

chance. The few robots that happen to run over a token, which prolongs their lifetime enough to meet another robot, or that have a good genome, are now the ones carrying the population forward. Here, a good genome is one that leads to the expression of reproductive beneficial behaviours. Robots need to come into communication range to exchange their genomes. If they become inactive in a remote location, it might take longer for another robot to venture into its neighbourhood and pass on its genome.

In an evolutionary algorithm a fitness function can be used in two ways: to select for reproduction and to select for replacement.

The mEDEA algorithm does not possess an explicit selection mechanism that is based on an explicit fitness value. Selection happens implicitly. The selection from the local genome storage is completely random, which means that every genome has an equal chance of being selected. For example, a genome (A) leads its carrying robot to act poorly by circling in a corner of the arena, avoiding energy tokens and other robots. Another robot, with a better genome (B), has a more exploratory behaviour, which seeks

out multiple robots on its quest for energy. The exchange of genomes is reciprocal. If genome A is now chosen in the next generation to replace genome B in the second robot in a more frequented place of the arena than the corner where A originated from, it will likely be passed on to many more robots. Many more scenarios are imaginable in which ill-adapted genomes can spread through the population.

3.4 mEDEA with Relative Fitness

From this section onwards new ideas are described. This section focuses on proposed modifications to the mEDEA algorithm, dubbed $mEDEA_{rf}$ — mEDEA with relative fitness.

Each robot estimates its fitness in terms of its ability to survive based on the balance between energy lost and energy gained, delta Energy (δ_E). This term is initialised to 0 at $t = 0$, when the current genome was activated. It reflects a robot's energy balance and, therefore, undergoes the same changes as stated in equation 3.2. Given δ_E, a robot calculates a fitness value that is relative to robots in a range r according to equation 3.3, where f_i' is the relative fitness of robot i at time t, $\overline{\delta_{\mathrm{Er}_i}}$ is the mean δ_E of the robots within the sub-population defined by all robots in range r of robot i, and $\sigma_{\overline{\delta_{\mathrm{Er}_i}}}$ is the standard deviation of the δ_{Er_i} of the sub-population.

$$f_i'(t) = \frac{\delta_{\mathrm{E}_i}(t) - \overline{\delta_{\mathrm{Er}_i}}(t)}{\sigma_{\overline{\delta_{\mathrm{Er}_i}}}(t)} \tag{3.3}$$

Note that f_i' is defined in relation to the ability of the robot to *survive* in the environment; it records the net energy of a robot, accounting for energy expended and energy gained by locating tokens.

The new explicit fitness function can be exploited in two ways:

- the fitness value can be transmitted with a genome and used by an individual within an explicit selection function.

- it can be used to influence the rate at which a genome is broadcast, thereby indirectly affecting its chances of being selected for reproduction.

The two approaches are described below.

3.4.1 Explicit Selection Mechanisms

The $select_{random}(genomeList)$ method in mEDEA (outlined in the pseudo code in algorithm 1) can easily be replaced with an informed selection method that uses the relative fitness measure to discriminate between genomes. Three well-known selection strategies are investigated:

- tournament selection
- roulette-wheel selection
- elitist select-best strategy

3.4.2 Biasing Broadcasting of Genomes

Alternatively, the spread of genomes can be biased by adapting the *broadcast()* step in algorithm 1. In mEDEA, robots make a random selection from their list of collected genomes at the end of each generation. Two new methods are proposed, both of which bias the spread of genomes throughout the population in favour of higher quality ones, based on a robot's estimation of its fitness f' relative to those in its immediate surroundings.

- *broadcast_range()* adapts the range at which a robot broadcasts depending on f'
- *broadcast_probability()* broadcast within a fixed range r with the probability depending on f'

```
1  load(currentGenome = randomInitialisedGenome);
2  while iteration ≤ maxIterations do
3      if hasGenome() then
4          if lifetime ≤ maxLifetime & energy > 0 then
5              move();
6              if neighbourhood.isNotEmpty() then
7                  rf =calculateRelativeFitness(neighbourhood);    // eq. 3.3
8                  broadcast(currentGenome,rf);
9              end
10         else
11             remove(currentGenome);
12         end
13     end
14     genomeList.addIfUnique(receivedGenomes);
15     if genomeList.size() > 0 then
16         genome = select<selection−strategy>(genomeList);
17         load(currentGenome = applyVariation(genome) );
18         genomeList.empty();
19         lifetime = 0;
20     end
21 end
```

Algorithm 2: Pseudo code of the adapted version of the mEDEA algorithm with relative fitness mEDEA$_{rf}$, used with the explicit selection mechanism.

Given f_i', we define the probability of a robot broadcasting using equation 3.4, which simply describes a function that returns a probability of 0 if f_i' is less than d_0, probability of 1 if f_i' is more than $d_{\max}$ standard deviations away from the mean, and, otherwise, linearly interpolated between 0 and 1.

$$p_i(t) = \begin{cases} 0 & f_i'(t) < d_0 \\ \frac{f_i'(t) - d_0}{d_{\max} - d_0} & d_0 \leq f_i'(t) \leq d_{\max} \\ 1 & f_i'(t) > d_{\max} \end{cases} \tag{3.4}$$

For the *Broadcast_probability()* method, shown in *case*1 in algorithm 3, the probability $p_i(t)$ is used directly to determine whether a robot broadcasts. For *Broadcast_range()*, the probability $p_i(t)$ is converted to a broadcasting range between 0 and a value $r_{\max}$ according to equation 3.5 — the higher the relative fitness, the greater the broadcast range. Note that range increases with the square root of the probability in order to maintain a proportional increase in broadcast *area*.

$$r_i(t) = r_{\max} * \sqrt{p_i(t)} \tag{3.5}$$

Both methods result in robots that have a higher relative fitness, broadcasting their genome more than those with a lower relative fitness, hence, biasing the quality of genomes that a receiving robot collects. At the end of each generation, a random selection of a genome is made from those collected, as in mEDEA.

3.5 Hypotheses

The following hypotheses inform the experimental design and are tested through experimental investigation.

```
1  /* Calculating the relative fitness, step 7 in algorithm 2   */
2  R_i ← getRobotsWithinRange(r_max);
3  if R_i > 0 then
4  |   f'_i ← getRelativeFitness(R);                          // eq. 3.3
5  |   p_i ← getProbability( f'_i);                           // eq. 3.4
6  else
7  |   p_i = 0
8  end
9  /* Varying the broadcasting mechanism, step 8 in algorithm 2
      */
10 switch exp do
11 |   // vary probability
12 |   case 1 do
13 |   |   if p_i > rand() then
14 |   |   |   broadcast(r_max,currentGenome,σ);
15 |   |   end
16 |   end
17 |   // vary broadcast range
18 |   case 2 do
19 |   |   r_i ← adjustRange(p_i);                            // eq. 3.5
20 |   |   foreach robot j in R do
21 |   |   |   if distance(i, j) < r_i then
22 |   |   |   |   broadcast(r_i,currentGenome,σ);
23 |   |   |   end
24 |   |   end
25 |   end
26 end
```

Algorithm 3: Pseudo code of the algorithm to calculate the relative fitness of robot i in its current neighbourhood of range r_{max}, and variation of broadcasting mechanism based on the relative fitness.

Explicit Selection Mechanism

The following alternative hypothesis is used for experiments in which the relative fitness value is used for explicit selection, described in 3.6.2.

Hypothesis 1 *Replacing the random selection method in mEDEA with a selection method that selects based on the relative fitness value will increase both the average maintained level of energy (δ_E) of the population and number of robots active (N_{active}) at the end of the final generation when compared to the original mEDEA algorithm.*

Biasing the Spread of Genomes

The following alternative hypotheses are used for experiments in which the relative fitness value is used for biasing the broadcast of genomes, described in 3.6.3.

Hypothesis 2 *Biasing the spread of genomes via adapting the* probability *that a robot broadcasts based on its relative fitness will improve the average δ_E of the population and N_{active} compared to the original mEDEA algorithm.*

Hypothesis 3 *Biasing the spread of genomes via adapting the* range *over which a robot broadcasts based on its relative fitness will improve the average δ_E of the population and N_{active} compared to the original mEDEA algorithm.*

3.6 Experiments

Three sets of experiments were undertaken, exploring the effects of using the explicit selection mechanism, biasing the spread of genomes through altering the broadcasting mechanism, and finally biasing the spread *and* using explicit selection.

3.6.1 Methodology

All experiments use Roborobo by Bredeche et al. from [22], as in the original simulations described with mEDEA. Roborobo is a multi-platform, highly portable robot simulator for large-scale collective robotics experiments. With respect to other robotic simulators, Roborobo combines (pseudo-)realistic modelling with fast-paced simulation and thus falls somewhere inbetween very realistic frameworks, such as Player/Stage [45], that tend to be very slow, and agent-based tools, such as MASON [74], that are extremely simplified with respect to the environment. It focuses solely on large-scale swarms of robots in a 2D environment and is based on a Khepera/ePuck model. It has already been used in more than a dozen published research papers mainly concerned with evolutionary swarm robotics, including environment-driven self-adaptation and distributed evolutionary optimisation, as well as online onboard embodied evolution and embodied morphogenesis.

The genome defines the weights of an Elman recurrent neural network (RNN) consisting of 43 sensory inputs and 2 motor outputs (translational and rotational speeds). 8 ray-sensors are distributed around the robot's body. They detect the proximity to the nearest object, the presence of walls and other robots, whether it belongs to the same group and the relative orientation between the two robots. An energy level input feeds the current level into the network. A distance and angle sensor gives the direction to the nearest energy token. The RNN has 1 hidden layer with 8 nodes, thus 434 weights are defined by the genome.

Table 3.1 mEDEAs evolutionary algorithm parameters for all experiments in this thesis.

Evolutionary Algorithm parameters	
Variation operator	Gaussian mutation with σ parameter
σmin value	0.01
σmax value	0.5
σinitial value	0.1
α (ie. σ update parameter)	0.35

All parameters used in the experiments are given in table 3.1 and table 3.2. Simulation parameters are based on the original papers. Experimental parameters were chosen following limited empirical tuning. The maximum broadcasting range requires sensible selection and should be chosen according to the arena size.

Table 3.2 Simulation and experimental parameters for all experiments in this chapter.

Simulation parameters	
Arena size	1024 pixel by 1024 pixel
Number of robots	100
Robot lifetime	1500 iterations
Food regrow time	500 iterations
Sensor range	32 pixel
Chromosome length	434
Agent rotational velocity	30deg/step
Agent translational velocity	2 pixel/step
Experimental parameters	
Number of independent runs	30
Maximum generations	500
Number of energy tokens	800
Energy value of token	100
Start energy	1200
Maximum range $r_{\max}$	64
d_0	0
$d_{\max}$	2

3.6.2 Experiment Set 1: Explicit Selection Mechanism

The first set of experiments investigate the hypothesis that replacing the random selection method in mEDEA with a selection method that selects based on the relative fitness value, will increase both the average δ_E of the population and number of robots alive N_{active} at the end of the final generation when compared to the original mEDEA algorithm. Three selection methods are investigated: binary tournament, roulette-wheel and an elitist select-best. These experiments are labelled:

- **E1** (vanilla) mEDEA
- **E1+t** mEDEA + tournament selection
- **E1+rw** mEDEA + roulette-wheel selection
- **E1+b** mEDEA + best selection

to denote the different selection methods.

3.6.3 Experiment Set 2: Varying the Broadcasting Mechanism

The new methods *broadcast_probability()* and *broadcast_range()* introduce *two* modifications compared to the original algorithm: (1) the broadcast probability (and therefore range) is variable across the population and (2) the broadcast probability (and therefore range) is determined by *relative fitness*. In order to show that any improvement in average δ_E can be attributed to the effect of introducing the *relative fitness* term, rather than simply a random variation, additional control experiments are performed as follows:

Rather than calculating the relative fitness of a robot according to equation 3.3, using its own $\delta_{\mathrm{Er}_i}(t)$, it is simply replaced with x_i — a random number drawn from a normal distribution with mean $\Delta(t)$ and $sd\Delta(t)$, where the Δ terms refer to the mean and standard deviation of the fitness of the *global* population. The global fitness is used simply to ensure that the random value is drawn from an appropriate range. New methods *broadcast_randomProbability()* and *broadcast_randomRange()* then use equations 3.4 and 3.5 as previously described. These methods are introduced merely to perform rigorous control experiments: it is not to suggest that this method be used in practice as it requires the calculation of a global parameter, contrary to the distributed nature of the algorithm.

Five different experiments are performed, where E1-E3 are controls and E4 and E5 evaluate the new methods.

- **E1** records the mean δ_E of the robot population and the number of active robots at the end of the final generation using only the original version of mEDEA
- **E2** (control for E4) records the same metrics as E1 using *broadcast_randomProbability()*
- **E3** (control for E5) records the same metrics as E1 using *broadcast_randomRange()*
- **E4** records the same metrics as above using *broadcast_probability()*
- **E5** records the same metrics as above using *broadcast_range()*

3.7 Evaluation and Analysis

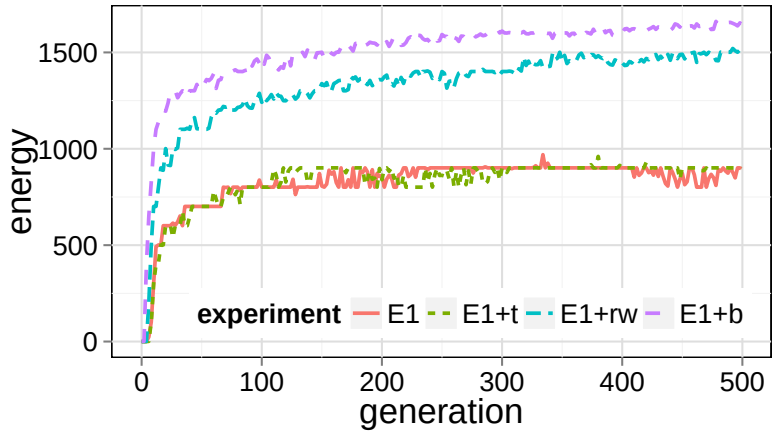
Results for all approaches are compared to the original mEDEA algorithm. Experiments show that both methods perform equally well compared to the original algorithm. Note, however, that broadcasting in the physical world is an energy consuming operation; methods that reduce this energy in order to save battery by either reducing the range or frequency of broadcasting are likely to be of considerable benefit.

3.7.1 Experiment Set 1: Explicit Selection Mechanism

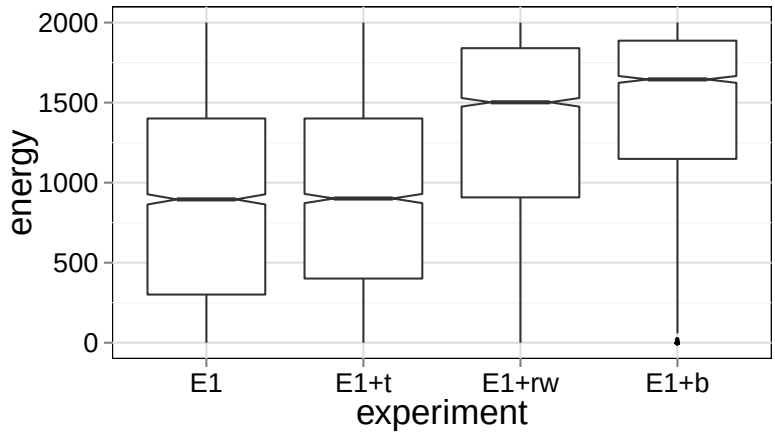
Results from the experiments E1, E1+t, E1+rw, E1+b in which the $select_{random}(genome\_list)$ method in algorithm 1 is replaced with a selection method are shown in figure 3.2 and 3.3, which compares the median[1] energy and agents alive over 30 repeated runs for each of the four experiments listed. Adding an explicit selection method based on a relative fitness value relating to the ability of another robot to survive over the generation has a significant effect in the case of *roulette-wheel* and *best* selection when compared to mEDEA. Both of these methods exert high selection pressure. In contrast, the low-pressure *tournament* selection method shows little difference to the random selection method of mEDEA. Wilcoxon rank-sum tests confirm that the *roulette-wheel*

[1] A Shapiro-Wilk test showed that the results were not normally distributed.

and *best* methods provide significantly different results for both energy and N_{alive}, while no significant difference is observed with the *tournament* selection method for either metric. The highest pressure selection method *best* outperforms *roulette-wheel* with statistically significant results at the 0.05 significance level.

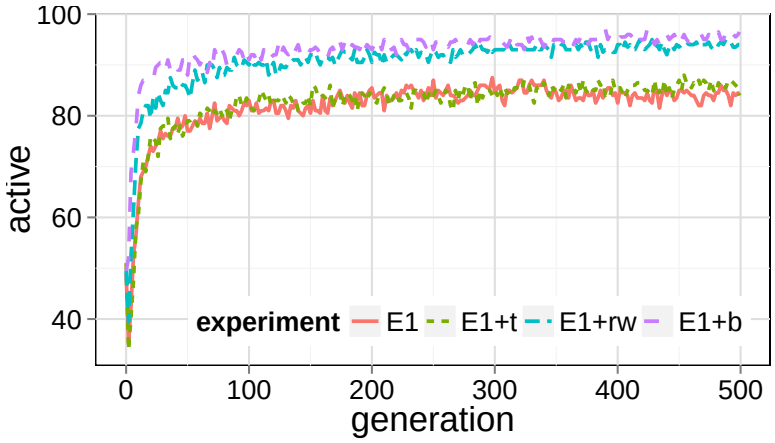


(a) Energy

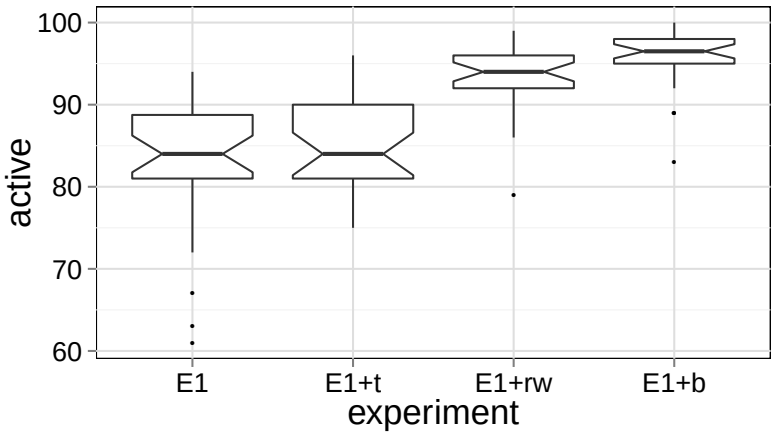


(b) Energy at gen. 500

Fig. 3.2 Explicit selection added to the mEDEA algorithm. Figures show the energy for experiments E1, E1+t, E1+rw, E1+b.



(a) Active robots



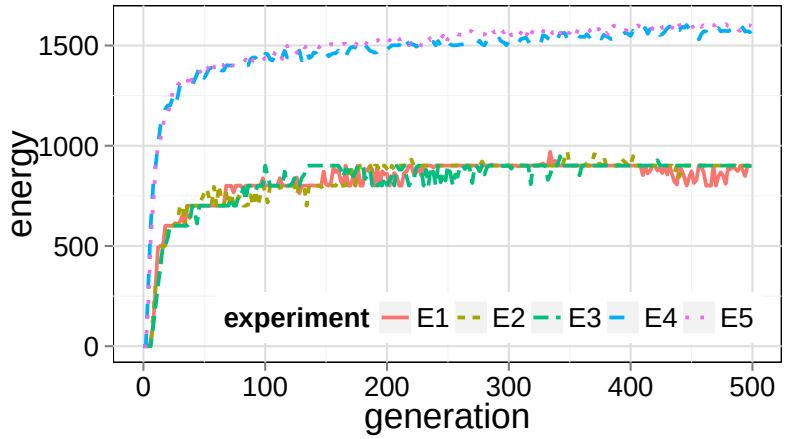
(b) Active robots at gen. 500

Fig. 3.3 Explicit selection added to the mEDEA algorithm. Figures shows the number of active robots for experiments E1, E1+t, E1+rw, E1+b.

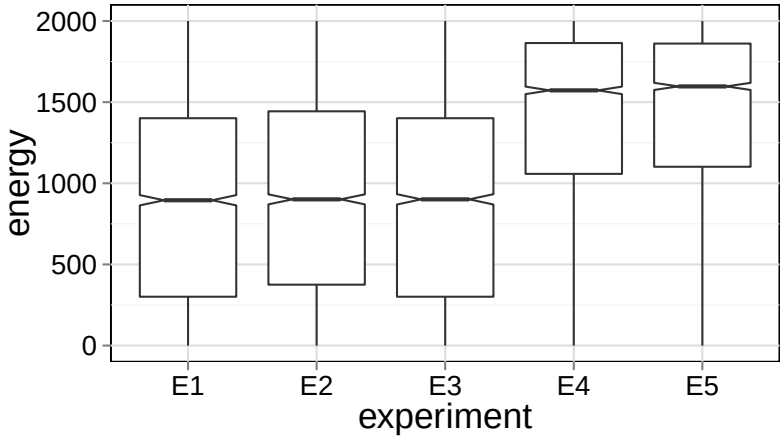
3.7.2 Experiment Set 2: Varying the Broadcasting Mechanism

The second set of experiments examine the results of using the two new broadcasting methods, comparing results to the original mEDEA algorithm. Figures 3.4, 3.5 and 3.6 clearly shows that experiments E4 and E5, which introduce the new broadcasting methods, outperform both the original mEDEA algorithm and the two control experiments. A Wilcoxon rank-sum test with significance level $\alpha = 0.05$, shows that the difference in final energy at generation 500 for both E4 and E5 is statistically different to E1, E2 and E3, but that there is no statistical difference between E4 and E5. The fact that E4 and E5 differ significantly from controls E2 and E3 respectively, shows that the differences

in performance are not simply attributable to varying the broadcast rate or range. They must be related to the fact that the broadcast rate and range are adjusted according to the estimate of fitness f' calculated by each robot. A corresponding pattern is observed when examining the number of active robots. Figure 3.6 clearly show that the number of genomes broadcast significantly decreases with respect to the original methods, but this is compensated for by using the higher environmental pressure achieved by adapting what is broadcast based on the quality estimate f'.

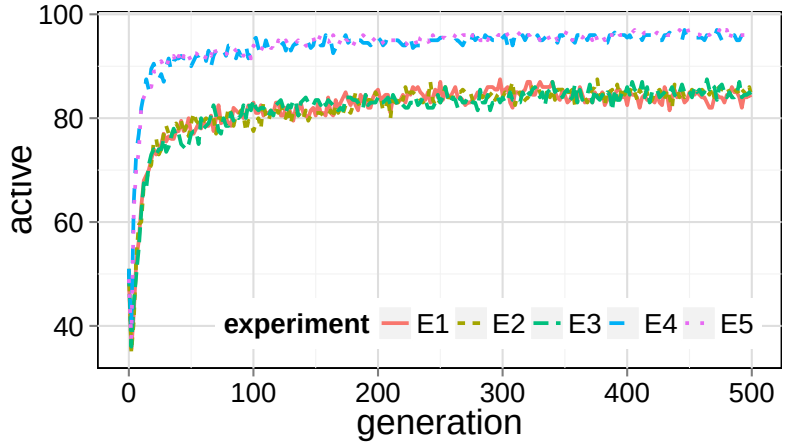


(a) Energy

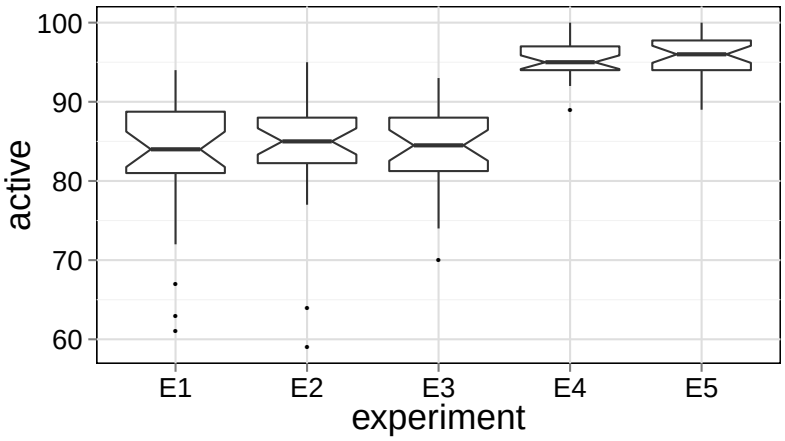


(b) Energy at gen. 500

Fig. 3.4 mEDEA, control experiments and biased broadcasting: figures show the energy for each of the experiments E1-E5.

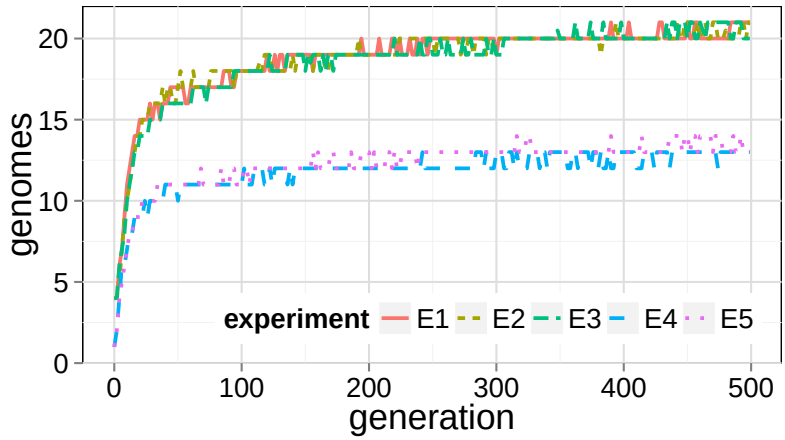


(a) Active robots

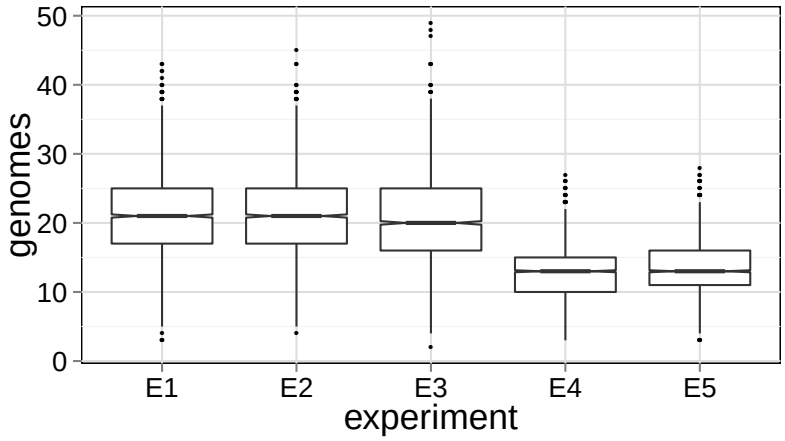


(b) Active robots at gen. 500

Fig. 3.5 mEDEA, control experiments and biased broadcasting: figures show the number of active robots for each of the experiments E1-E5.



(a) Genomes broadcast



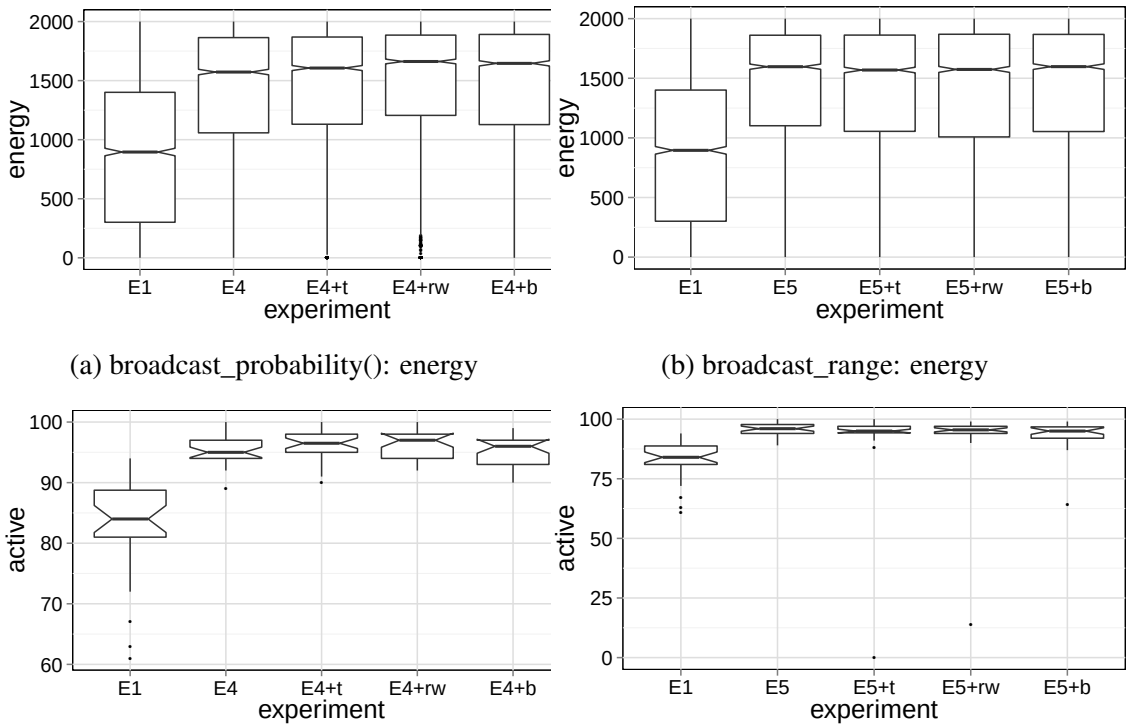
(b) Genomes broadcast at gen. 500

Fig. 3.6 mEDEA, control experiments and biased broadcasting: figures show the number of genomes received for each of the experiments E1-E5.

In the original mEDEA, all robots broadcast indiscriminately at the same fixed range, which creates a very weak selection pressure, resulting in genomes spreading more widely and having more chance of being selected when considering the population as a whole. Behaviours that lead to a robot coming into contact with more robots will result in more spreading of genomes and thus, on average, a higher probability of generating future offspring. In contrast, the more discriminate methods of broadcasting proposed in this chapter create higher selection-pressure: genomes that have a higher relative fitness have more chance of being received by other robots than lower fitness ones, and, thus, are more likely to be randomly selected.

3.7.3 Combining Explicit Selection with Biased Broadcasting

Finally, the effect of combining explicit selection within an individual with biased broadcasting is investigated. Each of the three selection methods are tested in combination with the two biased broadcasting methods in E4 and E5. Figure 3.7 shows box-plots of the results obtained from using the two $mEDEA_{rf}$ variants and vanilla mEDEA. Each of the $mEDEA_{rf}$ variants are significantly better in terms of energy level and active robots compared to vanilla mEDEA using an explicit selection method, confirmed using a Wilcoxon Rank-Sum test with a significance level $\alpha = 0.05$.



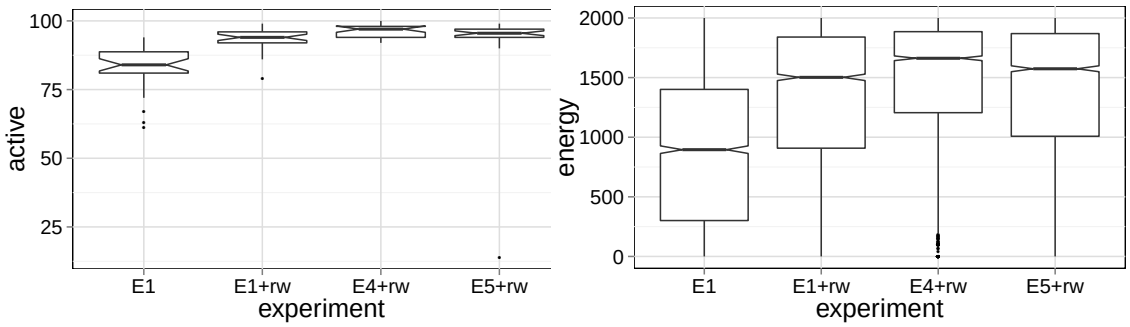
(a) broadcast_probability(): energy

(b) broadcast_range: energy

(c) broadcast_probability: active robots

(d) broadcast_range(): active robots

Fig. 3.7 Combining the biased broadcasting of genomes with explicit selection by individuals.



(a) Active robots at gen. 500

(b) Energy at gen. 500

Fig. 3.8 Comparison between vanilla mEDEA, mEDEA with explicit selection by individuals and $mEDEA_{rf}$ with explicit selection by individuals.

In order to easily contrast the new methods to the original algorithm, figure 3.8 compares mEDEA, mEDEA+rw and the two new broadcasting methods combined with roulette-wheel selection. Further, in table 3.3 we use the Wilcoxon's Rank-Sum test to compare pairs of experiments with and without explicit selection (indicated by $Ei+s$ Ei respectively). Statistically significant results are shown in bold.

Table 3.3 p-values obtained from applying Wilcoxon's Rank-Sum test across pairs of experiments, including biased broadcast only and biased broadcasting coupled with an explicit selection method.

		E1	E4	E5
E4	Energy	**< 2.2e-16**		
	Alive	**3.079e-10**		
E5	Energy	**< 2.2e-16**	0.3521	
	Alive	**4.112e-10**	0.4885	
E1+s	Energy	**< 2.2e-16**	**2.13e-05**	**1.963e-07**
	Alive	**6.207e-08**	0.0684	0.1994
E4+s	Energy	**< 2.2e-16**	**1.227e-07**	**1.007e-05**
	Alive	**1.222e-10**	0.1076	0.2793
E5+s	Energy	**< 2.2e-16**	0.5267	0.1288
	Alive	**3.287e-09**	0.7715	0.7658

3.8 Summary and Conclusion

3.8.1 Summary

The following comments can be made to summarise all experiments. Where claims are made, they are evidenced by data that is statistically significant as shown in table 3.3.

- Coupling the standard mEDEA algorithm with a high-pressure explicit selection method results in a more robust and sustainable population (higher energy and more active robots) than the standard mEDEA. However, using a low-pressure explicit selection method does not result in any statistical difference.

- The new methods of biasing the spread of genomes based on relative fitness combined with a random selection method by individual robots (E4, E5) results in a more robust and sustainable population than mEDEA (higher energy and more active robots). However, there are no discernible differences between the two new methods.

- Coupling the methods for biasing the spread of genomes (E4, E5) with an explicit selection method by individual robots improves on the standard mEDEA, but in most cases does not provide any significant advantage over biasing the spread and using random selection, with the exception of improving energy levels in the case of $E4+s$ when compared to E4 alone.

- Using roulette-wheel selection combined with standard mEDEA outperforms the two experiments in which the spread of genomes is biased but individuals apply random selection in terms of sustaining higher levels of energy within the population. However, this has no significant effect on the number of active robots. Note that using the explicit selection method comes at a potentially high cost in terms of the number and range of broadcasts required to implement this when compared to the biased broadcasting methods.

In summary, the results show that using mEDEA with a relative fitness function that either promotes the spread of good genomes (via biasing what is transmitted) or promotes the selection of genomes with high energy values (explicit selection) results in swarms that sustain high energy levels and high percentages of active robots when compared to the original version.

However, when considering real, physical robots, it should be clear that broadcasting comes with an overhead in terms of the energy required to communicate. Two factors influence the cost of broadcasting in energy terms — the number of broadcasts made and the broadcast range. For the explicit selection and *broadcast_range()* methods, the same number of broadcasts are made — however, *broadcast_range()* results in a range of broadcast distances $r \leq r_{\max}$, whereas the explicit selection method combined with mEDEA always broadcasts at $r_{\max}$, thus utilising greater energy. The *broadcast_probability()* method directly reduces the number of broadcasts made with respect to mEDEA as weaker robots broadcast less on average, thus saving energy.

3.8.2 Conclusion

A number of extensions have been introduced to an existing Environment-driven Distributed Evolutionary Adaptation algorithm — mEDEA. The goal of the work is to show that the integrity of the swarm can be maintained in a more robust manner than in the original work, while still retaining the original distributed and online flavour of the algorithm by using a fitness function that indicates fitness to survive. Having introduced the new fitness function, two new methods are described that adapt either the broadcast range or the probability of broadcasting of a robot, based on its estimate of its own relative fitness.

This biases the spread of genomes through the population. Robots that are relatively fitter than their neighbours are able to spread their genomes more: individual robots perform a random selection from their local store of (now biased) genomes. A thorough

analysis of the experimental results shows that a considerable gain in performance is achieved, both in the number of active robots at the end of a fixed period of evaluation, and in the energy levels sustained by these robots.

The new fitness function was also evaluated within an explicit selection method. Experiments show that this also provides significant improvements over mEDEA, and slightly outperforms the biased broadcast methods in terms of energy sustained.

However, as described above, this comes with a higher cost than either of the biased broadcast methods in terms of the energy used in transmitting. When considering real robots this factor can have a significant impact on survival ability — in many real-life scenarios, the ability to prolong battery life by reducing energy usage might well be critical. An obvious extension to this work is therefore to account for the cost of broadcasting when calculating the net energy of a robot that is used by the fitness function. The next chapter describes an experimental approach to test whether this hypothesis will differentiate results from the two sets of experiments described here.

Chapter 4

Influence of Communication Cost on the Effectiveness of the Broadcasting Variation Mechanism in mEDEA$_{rf}$

In the previous chapter a fitness measure was introduced into mEDEA that is based on the individual's energy level. It was used to vary the broadcasting mechanism for the spread of genomes. In real robot hardware communication comes at a cost, in the form of energy consumption. Varying the broadcasting mechanism will therefore have a direct impact on the amount of energy consumed by a robot. Since the fitness measure is based on an individual's energy level, this creates a circular dependency between fitness measure and the behaviour influencing the fitness.

4.1 Contribution

The work described in this chapter has in parts been published[1] in the proceedings companion of the Genetic and Evolutionary Computation Conference (GECCO 2015) [120]. A copy of the publication can be found in the appendix (A.2).

It is shown from the literature that in real robot hardware communication comes at a cost in the form of energy consumption. In this chapter an energy model for communication is derived from the field of wireless sensor networks research, which accounts for the cost of communication in the robot simulation. The experiments carried out show that accounting for the costs of communication exerts additional environmental pressure. Results show that under these new conditions, biasing the broadcasting of genomes has a negative effect on survivability.

4.2 Introduction

When a simulator is used in evolutionary robotics experiments it is important that the simulation environment provides enough detail of the features that affect the evolutionary process. The underlying evolutionary algorithm always exploits the particular features of the environment. If those features differ in simulation and the real world then it is not guaranteed that insights gained in one world will hold true in the other [83]. This is an important aspect of the reality gap. In real, physical robots, any type of communication comes at a cost in the form of energy. It is therefore important that the simulator reflects these costs as accurately as possible.

The reality gap is not just a problem in evolutionary robotics, but exists wherever a simulator is used to approximate the behaviour of a real system in a simulated environment. In wireless sensor networks research the same type of hardware is used

[1] Some experimental parameters presented in this chapter vary from the published work to enable a better comparison with the experiments of the previous chapter. Details about these changes are given in the experiment methodology section (4.6.1).

that research robot platforms (e.g. ePuck [79]) are commonly equipped with [26]. The field focuses on all aspects of mobile communication using low powered devices, with one main focus to prolong the battery lifetime and therefore lifetime of the application. Hence, accurate simulation of energy consumption plays an important role [57]. Since long term autonomy in robotics application is usually based on battery power, this commonality makes those aspects of research directly transferable to this research.

The relative fitness function that was introduced in the previous chapter uses a robot's energy-level as a fitness measure. The robot's broadcasting mechanism is then varied using this fitness to bias the spread of genomes across the swarm. As mentioned above, all communication comes at a cost in the form of energy, hence the way in which the relative fitness function is used directly influences the fitness measure itself. Focus here lies on reflecting the energy consumption on which the fitness measure is based on, rather then optimising the performance of the adaptation algorithm.

The goal here is to investigate how the previously obtained results change when an improved energy model is used, specifically, when accounting for the energy consumed by communication. To this end a new energy model has been developed and implemented into Roborobo which so far lacked an accurate energy model. A suitable energy model has been borrowed from wireless sensor networks research and the appropriate parameters have been derived using values from real communication modules.

4.3 The Use of Energy in mEDEA$_{rf}$

The current energy model of the simulation software Roborobo [22] is fairly simplistic. Every time-step, the energy value is decreased by a 'living' cost $E_{\text{living}} = 1$ unit. Energy *tokens* are scattered throughout the environment. If a robot moves over a token, its energy is increased by an amount E_{token}.

In the previous chapter about mEDEA$_{rf}$, two methods were introduced to influence the broadcasting of robots: varying the frequency and range depending on the

individual's relative fitness f_i'. As this fitness value is based on the energy level of a robot, which is then used to directly influence its broadcasting mechanism, it is crucial to account for the energy being used for these communications.

Section 3.4 describes the mEDEA$_{rf}$ algorithm in detail. In short, given f_i', the probability of a robot broadcasting is defined using equation 3.4 that simply describes a function that returns a probability of 0 if f_i' is less than d_0, a probability of 1 if f_i' is greater than $d_{\max}$ standard deviations away from the mean, and linearly interpolated between 0 and 1 otherwise.

$$p_i(t) = \begin{cases} 0 & f_i'(t) < d_0 \\ \frac{f_i'(t) - d_0}{d_{\max} - d_0} & d_0 \leq f_i'(t) \leq d_{\max} \\ 1 & f_i'(t) > d_{\max} \end{cases} \qquad \text{(3.4 revisited)}$$

The probability $p_i(t)$ is used in two ways: to determine whether a robot broadcasts, and to convert to a broadcasting range between 0 and a value $r_{\max}$, determined by equation 3.5

$$r_i(t) = r_{\max} * \sqrt{p_i(t)} \qquad \text{(3.5 revisited)}$$

The higher the relative fitness, the greater the broadcast range. Note that range increases with the square root of the probability in order to maintain a proportional increase in broadcast *area*.

Using the above method, the energy a real robot would use for communication would be reflected in its energy level, and therefore directly influence its fitness. This demonstrates the necessity to improve the energy model used in the simulator Roborobo.

4.4 Improving the Energy Model in the Simulator

As outlined in the previous section, the energy model needs to reflect the cost of communication in order to accurately reflect the relative fitness in mEDEA$_{rf}$. To base

the new model on the correct assumptions, first, the energy usage characteristics of real hardware wireless modules is assessed and an established and a widely used model is borrowed from the wireless sensor networks literature. Using the actual energy consumption data, appropriate values are calculated to be used in the simulator.

4.4.1 Energy Consumption Characteristics of Hardware Communication Modules

As argued in 4.3, the relative fitness function introduced in mEDEA$_{rf}$ is based on the robot's energy level and used to influence its communication. Therefore, the energy used for communication would directly influence the robot's fitness and needs to be accounted for. Furthermore, Heinzelman *et al.* [61] demonstrates that by not accounting for the energy usage of communication huge errors can occur in simulations. Although their research was conducted with a focus on communication protocols in the field of wireless sensor networks (WSN), it is highly relevant to the assessment revolving around communication in mobile robots. The technical requirements for data exchange in WSN and swarm robotics platforms are virtually the same: both employ low powered, mostly battery powered, systems with low bandwidth data transmissions between multiple entities to cover a large area [108, 71, 15]. It comes, therefore, as no surprise that both fields of research use the same low power communication technologies, such as IEEE 802.15.4 Wireless Personal Area Network protocol and Bluetooth [64, 99]. For example, Cianci *et al.* [26] used the microchips from Texas Instruments CC2420 and Mondada *et al.* [79] used the Bluetooth module LMX9820a in their experiments using the ePuck robot platform.

Experiments using the new communication energy model, in this and following chapters, are based on Cianci *et al.* [26], where a specific technical setup is used to simulate having only close range communication in a wider field. An additional signal

attenuation module is used that limits its reach (from $r_{\max} \approx 50\text{m}$ to $r_{\max} = 4.85\text{m}$), so that even in a small arena robots can be outside the maximum broadcast range.

When it comes to the actual wireless communication between robots, certain properties need to be considered that influence directly or indirectly the energy consumption of the involved communication partners [89, 61]. Data exchange between robots is purely based on broadcasts. Everyone within reach receives a broadcast message and therefore has to pay the cost for receiving it. In fact, using low-powered communication modules, receiving requires more energy than transmitting [131]. This is due to the low-powered nature of the signal, which requires expensive reconstruction. Even if no communication takes place, a wireless chip in *listen* mode still consumes energy. The same information is often sent and received multiple times, but then discarded as only unique information is kept. For example, this happens when the same genome in mEDEA$_{rf}$ is received multiple times. Here, it is important that every successful transmission incurs the cost of communication, even if the information is discarded. All these effects need to be taken into account when looking for a suitable energy model.

The Free-Space Model [61] is a simple model from the WSN literature that is able to address all of the criteria above. A much more detailed power consumption model was proposed by Wang *et al.* [131] that further considers internal effects of the power source, such as loss in conversion, non-linearities, noise in the channel, etc. However, these finer details go beyond what can be addressed and modelled in the simple robot simulator, Roborobo. When removing these details, the latter model is simplified to the point at which both models are equivalent [131], thus, the Free-Space Model is selected here.

4.4.2 Free-Space-Model

As discussed, a simple model used in wireless sensor network simulations is the Free-Space Model [61]. This model assumes homogeneous network nodes with no

specialised base station or relay nodes. It resembles the ad-hoc networks created during communication among the robots within the communication range. The model assumes quadratic growth with distance for energy consumption. Blom *et al.* [11] compared energy models using quadratic and cubic growth for energy consumption in relation to distance and data size. They concluded that cubic growth is a more accurate model for distances above 103 meters, which is way beyond the range considered here ($r_{\text{max}} = 4.85\text{m}$).

The equations 4.1 and 4.2 are used to calculate the energy required for receiving and transmitting respectively.

$$E_{\text{rx}}(n) = n_{\text{bit}} \times E_{\text{elec}} \tag{4.1}$$

$$E_{\text{tx}}(n,d) = n_{\text{bit}} \times E_{\text{elec}} + n_{\text{bit}} \times \varepsilon_{\text{amp}} \times d^2 \tag{4.2}$$

E_{elec} is the basic charge to run the module, and ε_{amp} is the cost for signal amplification which is multiplied by the $d$istance squared. In both equations, the terms are multiplied by n_{bit}, which represents the number of *bit* in a transmission. Note, n is constant as genome broadcasts only vary in content, not length.

The model assumes that the transmitter and receiver circuits consume the same amount of energy to run the module. However, as discussed in the previous section, this is not the case for the modules used *in silico* swarm robotics experiments. It is therefore differentiated between $E_{\text{rx-elec}}$ and $E_{\text{tx-elec}}$ in equation 4.3 and 4.4 respectively.

$$E_{\text{rx}}(n) = n_{\text{bit}} \times E_{\text{rx-elec}} \tag{4.3}$$

$$E_{\text{tx}}(n,d) = n_{\text{bit}} \times E_{\text{tx-elec}} + n_{\text{bit}} \times \varepsilon_{\text{tx-amp}} \times d^2 \tag{4.4}$$

4.4.3 Calculation of Communication Costs for the Simulation

Based on the chosen representation and parameter settings of the previous experiments in 3.6, the additional energy costs can now be calculated.

Each transmission contains a genome and a few parameter values regarding the sender. As all genomes have the same length and consist of the same data type, the number of transmitted bits can be assumed to be constant. Equations 4.3 and 4.4 can therefore be simplified to

$$E_{\text{rx}} = a_{\text{rx}} \tag{4.5}$$

$$E_{\text{tx}}(d) = a_{\text{tx}} + b_{\text{tx}} \times d^2 \tag{4.6}$$

where $a_{\text{...x}} = n_{\text{bit}} \times E_{\text{...x-elec}}$ is the basic cost for for receiving or transmitting, and $b_{\text{tx}} = n_{\text{bit}} \times \varepsilon_{\text{tx-amp}}$ is an amplification cost that only occurs in the case of a transmission.

Table 4.1 Values taken from the data-sheet for wireless communication module Texas Instruments CC2420 [125].

Parameter	**Value**	**Unit**
Operating Voltage	3	V
Current RX mode	18.8	mA
Current TX mode, min.	8.5	mA
Current TX mode, max.	17.4	mA
Data rate	250	$\frac{\text{kbit}}{\text{s}}$
Distance with min amplification d_{min}	0.5	m
Distance with maximum amplification d_{max}	4.85	m

Using equations 4.5 and 4.6, and the electrical specifications for the communications module used by Cianci et al. [26] (Chipcon CC2420, see table 4.1), the values for a_{rx}, a_{tx} and b_{tx} can be calculated as follows:

$$
\begin{aligned}
E_{\text{rx}} &= n_{\text{bit}} \times E_{\text{rx-elec}} \\
&= a_{\text{rx}} \\
&= 1767 \frac{\text{byte}}{\text{message}} \times 8 \frac{\text{bit}}{\text{byte}} \times \frac{3\text{V} \times 18.8\text{mA}}{250 \frac{\text{kbit}}{\text{s}}} \\
&= 14136 \frac{\text{bit}}{\text{message}} \times 225.6 \frac{\text{nJ}}{\text{bit}} \\
&= 3.189 \frac{\text{mJ}}{\text{message}}
\end{aligned}
\tag{4.7}
$$

Table 4.2 Size of a data message (transmitted genome) in byte.

Variable	**Size in byte**
Genome (434 float values)	1736
Sigma	4
Fitness	4
Age	4
Robot ID	4
Genome ID	4
Wireless Protocol Overhead	11
Total =	1767

According to the proposed energy model, the receiving costs are constant while the costs for transmission have a variable part. Besides the constant base energy required to operate the radio module in transmission mode, the signal can be amplified to cover a greater distance. The further the signal should reach, the more it needs to be amplified and the higher the energy consumption of the radio module. As the power consumption at minimum and maximum amplification levels are known, the fixed (a_{tx}) and variable

part (b_{tx}) of equation 4.6 can be calculated as follows:

$$\begin{aligned} a_{\text{tx}} &= n_{\text{bit}} \times E_{\text{tx-elec}} \\ &= 1767 \frac{\text{byte}}{\text{message}} \times 8 \frac{\text{bit}}{\text{byte}} \times \frac{3\text{V} \times 8.5\text{mA}}{250 \frac{\text{kbit}}{\text{s}}} \\ &= 14136 \frac{\text{bit}}{\text{message}} \times 102 \frac{\text{nJ}}{\text{bit}} \\ &= 1.442 \frac{\text{mJ}}{\text{message}} \end{aligned} \tag{4.8}$$

With maximum amplification the signal can reach $d_{\text{max}} = 4.85m$. $E_{\text{tx}}(d)$ is therefore highest at this distance and b is calculated to reflect the quadratic growth. $\varepsilon_{\text{amp-max}}$ is the energy required for maximum amplification.

$$\begin{aligned} \varepsilon_{\text{amp-max}} &= \frac{3\text{V} \times (17.4 - 8.5)\text{mA}}{250 \frac{\text{kbit}}{\text{s}}} \\ &= 106.8 \frac{\text{nJ}}{\text{bit}} \end{aligned} \tag{4.9}$$

$$\begin{aligned} b_{\text{tx}} &= n_{\text{bit}} \times \frac{\varepsilon_{\text{amp-max}}}{r_{\text{max}}^2} \\ &= 1767 \frac{\text{byte}}{\text{message}} \times 8 \frac{\text{bit}}{\text{byte}} \times \frac{106.8 \frac{\text{nJ}}{\text{bit}}}{(4.85\text{m})^2} \\ &= 14136 \frac{\text{bit}}{\text{message}} \times 4.54 \frac{\text{pJ}}{\text{message} \times \text{m}^2} \\ &= 64.18 \frac{\text{nJ}}{\text{message} \times \text{m}^2} \end{aligned} \tag{4.10}$$

Which leads to the final equation 4.11

$$\begin{aligned} E_{\text{tx}}(d) &= a_{\text{tx}} + b_{\text{tx}} \times d^2 \\ &= 1.442 \frac{\text{mJ}}{\text{message}} + 64.18 \frac{\text{nJ}}{\text{message} \times \text{m}^2} \times d^2 \end{aligned} \tag{4.11}$$

The ePuck's battery has a capacity of $5Wh$ [79], which translates to $18.000J$. Assuming that the battery lasts the length of an experiment with 150 generations, of 1500

iterations each, the cost for broadcasting and receiving a message can be determined in relation to that. In previous experiments the median number of transmitted and received broadcasts per generation and robot, using $d_{\max}$, was 1100 and 2200 respectively. Given the previously calculated values for $E_{\text{tx}}(d_{\max})$ and E_{rx}, the average energy consumption is about $1540J$, leaving $16460J$ to be used for the radio *listen mode* and all other sensors, actuators and computation of the controller. Table 4.3 lists the energy model related values.

Table 4.3 Values for communication costs used in simulator.

Parameter	Value (in Energy Units)
$a_{\text{rx-Simulator}}$	0.04022
$a_{\text{tx-Simulator}}$	0.01818
$b_{\text{tx-amp-Simulator}}$	0.000809
$b_{\text{tx-amp-max-Simulator}}$	0.01904

The amount of energy spent on communication E_{com} is calculated using equation 4.12, where i and j are the number of genomes received and transmitted respectively.

$$E_{\text{com}} = \sum_{k=0}^{i} a_{\text{rx-Simulator}} + \sum_{k=0}^{j} \left(a_{\text{tx-Simulator}} + b_{\text{tx-amp-Simulator}} \times d^2\right) \tag{4.12}$$

4.4.4 Movement Dependent Living Costs

The fixed cost of 'living' $E_{\text{living}} = 1$ Energy Unit is independent of a robot's speed. Equation 4.13 shows how this is changed to stipulate a more diverse range of movements among the evolving robots. There is a fixed cost to 'living' of 0.5 units per time-step, regardless of whether the robot moves or not. A moving robot consumes an amount of energy that is related to its rotational speed v_{rot}, translational speed v_{trans} and their respective maximum values $v_{\text{rot-max}}$ and $v_{\text{trans-max}}$.

$$E_{\text{living}} = 0.5 + \left(\frac{v_{\text{rot}}}{v_{\text{rot-max}}} + \frac{v_{\text{trans}}}{v_{\text{trans-max}}}\right)/4 \tag{4.13}$$

4.4.5 The New Energy Model

A change in energy from one time-step to another now includes the cost of communication, as well as an updated cost for 'living' that partially depends on the increased power consumption through movement. Equation 3.2, from the previous chapter, is amended to account for the cost of communication using E_{com}.

$$\begin{aligned} E_i(t+1) &= E_i(t) - E_{\text{living-i}} + (n_{\text{token-i}} \times E_{\text{token}}) \\ E_i(t+1) &= E_i(t) - E_{\text{living-i}} - E_{\text{com-i}} + (n_{\text{token-i}} \times E_{\text{token}}) \end{aligned} \tag{4.14}$$

Equation 4.14 shows the change in energy at each time-step for robot i, where n is the number of tokens that have been collected in that step. Here, t_0 refers to the time the current genome is activated.

4.5 Hypothesis

After augmenting the algorithm with the derived energy model to account for the cost of communication, its influences are tested using the following hypothesis.

Hypothesis 4 *Biasing the spread of genomes in mEDEA$_{rf}$, combined with an explicit selection, outperforms continuous broadcast in terms of active robots and the maintained energy level at the end of the last generation when accounting for the cost of communication.*

4.6 Experiments

Experiments are carried out using mEDEA$_{rf}$, as described in 3.4. The previously derived energy model is used to account for the energy consumption that stems from communication. Two different methods were introduced in mEDEA$_{rf}$, which change the broadcasting mechanism in mEDEA by a) varying the probability to broadcast and

b) adjusting the broadcast range in proportion to the fitness value. Both mechanisms bias the broadcast towards fitter individuals.

The set of experiments from the previous chapter, described in detail in 3.6, are the basis for the experiments outlined here, with the addition of the newly developed energy model. The same notation is used as in the previous chapter for direct comparison of the results.

E1$_{em}$: baseline experiment, vanilla mEDEA using the energy model with random individual selection;

E4$_{em}$: using a fitness proportionate probability to broadcast;

E5$_{em}$: varying the broadcast range proportionate to the fitness.

Each of the three experiments above were then repeated with the random individual selection method being replaced with a fitness-proportionate selection method (roulette-wheel selection). These experiments are denoted as **E1**$_{em}$**+rw**, **E4**$_{em}$**+rw** and **E5**$_{em}$**+rw**. Note that the experiments E2 and E3 conducted previously were control experiments to determine whether any improvement could be contributed to using the relative fitness value in mEDEA$_{rf}$ to influence the range and probability of a broadcast, rather than a random value. Therefore, they do not need to be repeated here.

4.6.1 Methodology

The experiments outlined above were run in the simulator Roborobo [22] using the energy model derived in the earlier section. Compared to the published experiments in [120], the following adjustments have been made to allow for a direct comparison with the experiments conducted in the previous chapter:

- same value for energy tokens (from 75 to 100 Energy Units)
- same initial energy for robots (from 750 to 1200 Energy Units)

Note, not only has the cost of communication (E_{com}) been introduced, but also the living costs (E_{living}) have been changed in the new energy model. Hence, experiments E1, E4 and E5 from the previous chapter have been re-run to allow for a direct comparison between Ex and Ex_{em}, which only differ in the addition of communication costs.

All parameters used in the experiments are given in table 4.4.

Table 4.4 Simulation and experimental parameters for all experiments in this chapter.

Simulation parameters	
Arena size	1024 px × 1024 px
Number of robots	100
Robot lifetime	1500 iterations
Food regrow time	500 iterations
Sensor range	32 pixel
Chromosome length	434
Agent rotational velocity	30 deg/step
Agent translational velocity	2 pixel/step
Experimental parameters	
Number of independent runs	30
Maximum generations	500
Number of energy tokens	800
Energy value of token	100
Start energy	1200
Maximum range r_{max}	64
d_0	0
d_{max}	2

4.7 Evaluation and Analysis

4.7.1 Methodology

Following 30 runs of each experiment, statistical analysis was conducted based on the method in [107] using a significance level of 5%.

The distributions of two results were checked using a Shapiro-Wilk test. If one of the results followed a non-Gaussian distribution then the p-value is determined using a Kruskal-Wallis rank sum test. Otherwise the homogeneity of variance of the two results was performed using a Levene's test for homogeneity of variance. For unequal variances the p-value was determined using a Welch test, otherwise using an ANOVA test.

4.7.2 Influence of the Energy Model

Table 4.5 quantifies how the median values for *energy*, *active*, *broadcasts : genome*, *age : genome* and *token collected* compare to the result of the corresponding experiment without the energy model.

Figures 4.1, 4.2, 4.3, 4.4 and 4.5 further allows the visual comparisons of the median *energy* values and median number of *active* robots between the three experiments without ($E1_{em}$, $E4_{em}$, $E5_{em}$) and with ($E1_{em}$+rw, $E4_{em}$+rw, $E5_{em}$+rw) the fitness proportionate selection mechanism (roulette-wheel selection). Lineplots are smoothed to emphasise the trends, with the original data plotted in same coloured thinner lines. The violin-boxplots show the respective distribution of values over the last three generations of the experiments.

For vanilla mEDEA ($E1_{em}$) and mEDEA using the explicit selection method ($E1_{em}$+rw) the median energy value and *age : genome* (the average time spent between receiving unique genomes) went up and the number of *broadcasts : genome* (received broadcasts per unique genome) went down. At the same time the number of *token collected* remained unchanged, which points to a change in behaviour that avoids costly communication.

For the experiments using $mEDEA_{rf}$ alone ($E4_{em}$, $E5_{em}$) and with the addition of the explicit selection method ($E4_{em}$+rw, $E5_{em}$+rw), the most prominent change is the decrease in the number of *active* robots at the end of the generation. Directly related to

Table 4.5 Showing median of end values for *energy*, *active*, *broadcasts : genome*, *age : genome* and *token collected* over the last three generations of the experiment. The symbols ↓, ↔, ↑ indicate how the shown values compare to the result of the corresponding experiment without the energy model, with the number of arrows indicating the magnitude level of the effect size, based on a Vargha and Delaney A test (1 = small, 2 = medium, 3 = large).

	energy	active	$\frac{\text{broadcasts}}{\text{genome}}$	$\frac{\text{age}}{\text{genome}}$	token
$E1_{em}$	↑↑ 917.1	↓ 99.0	↓↓↓ 83.2	↑↑↑ 249.8	↔ 8.0
$E1_{em}$+rw	↑↑ 919.7	↓↓ 99.0	↓↓↓ 86.4	↑↑↑ 249.8	↔ 10.0
$E4_{em}$	↑ 964.6	↓↓↓ 99.0	↔ 37.6	↑↑↑ 299.8	↔ 10.0
$E4_{em}$+rw	↔ 931.2	↓↓↓ 99.0	↓ 37.0	↑↑↑ 299.8	↔ 8.5
$E5_{em}$	↓ 946.6	↓↓↓ 94.0	↔ 57.0	↑↑↑ 374.8	↓ 6.0
$E5_{em}$+rw	↓↓↓ 612.8	↓↓↓ 45.5	↑ 54.6	↑↑↑ 374.8	↓↓ 0.0

this is the overall rising values for *age : genome*, which reflects the average lifetime it took to pick up a unique genome. The fewer active robots in the arena, the more time robots have to spend looking for other robots to exchange their genome.

Figure 4.4 (d-f) shows a large spread of values in the number of active robots, with extinction events happening in every experiment using the new energy model. Although the difference for the median value in active robots for $E1_{em}$, $E1_{em}$+rw, $E4_{em}$ and $E4_{em}$+rw are fairly close, they significantly differ from their respective baseline experiment, without the energy model. In contrast, for experiments $E5_{em}$ and $E5_{em}$+rw this difference is quite obvious and figure 4.4 c) shows the reason. Following a sharp drop within the first generations, the numbers recover after a few more generations. Where $E5_{em}$ slowly recovers to over 90%, $E5_{em}$+rw breaks back down to finally settle below 50%.

Both energy and the number of collected tokens are down as well in $E5_{em}$ and $E5_{em}$+rw. The lower number of collected tokens can be explained by the fact that fewer robots are active to collect tokens. The median energy values are calculated across all robots independent of whether they are active. Hence a large number of inactive robots with energy values of zero bring the number down.

The dip in active robots within the first few generations is visible in all experiments, independent of the use of the energy model, however, in experiments using the energy model this effect is much larger. This is due to the increased selection pressure coming from the environment. The "cost of living" under these new conditions is larger, hence it requires better adaptation of the individuals to cope with these conditions. Where individuals previously could just get by, under the new energy model these behaviours are no longer sufficient to survive long enough to spread one's genome.

In E4 and E4+rw, where the frequency of broadcasts with maximum range is varied, no significant change in median energy levels is found compared to experiments without the energy model. However, the distribution of energy levels changed as can be seen in 4.3 due to the increased environmental selection pressure.

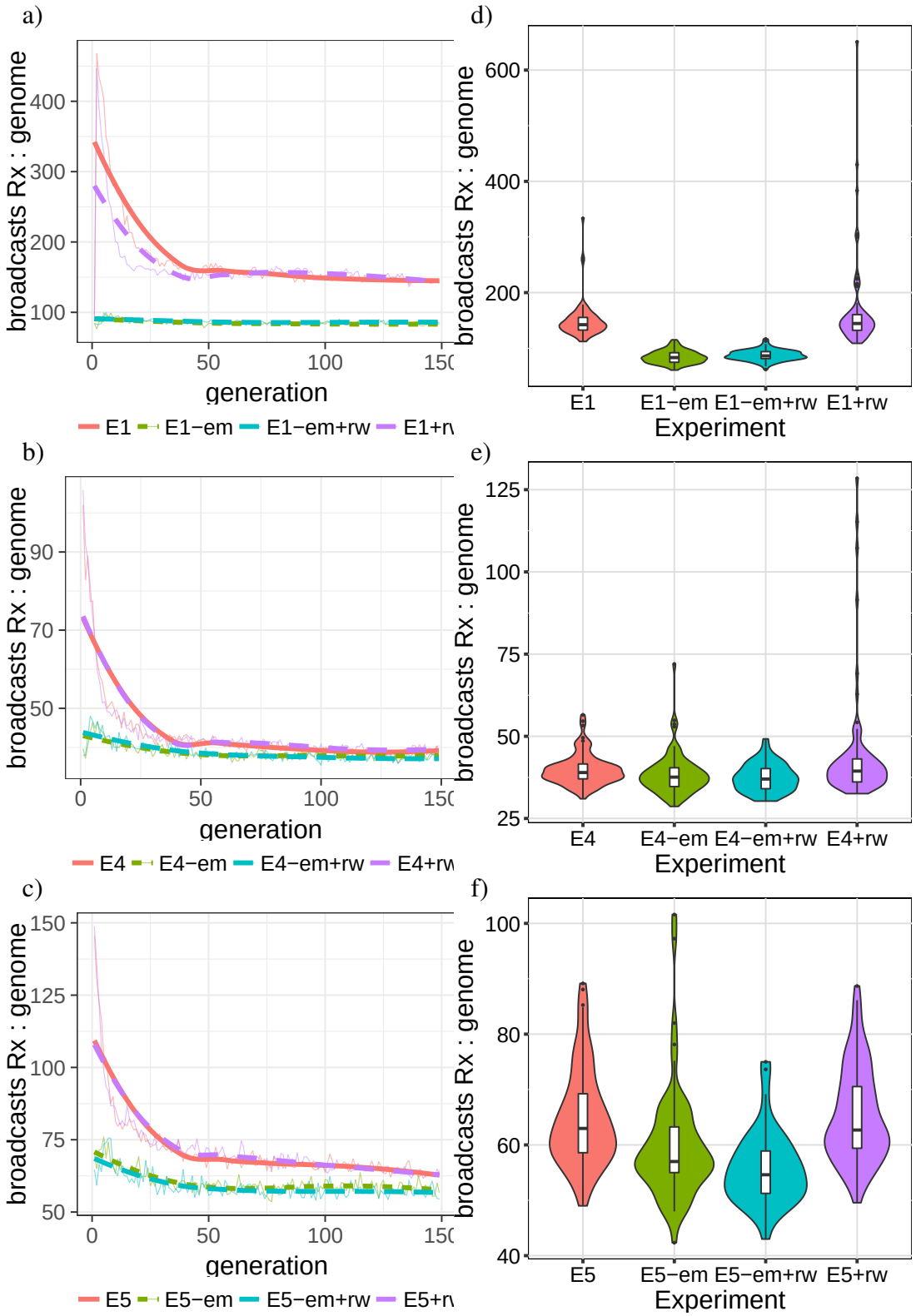


Fig. 4.1 Comparison of the median number of received broadcasts per unique genome between experiments, without (E*x*,E*x*+rw) and with the communication costs (E*x*-em, E*x*-em+rw). Violin and box plots showing the distribution of values over the last three generations of the experiment. Lines are smoothed to better show trends, with the raw data displayed in the same colour and thinner lines.

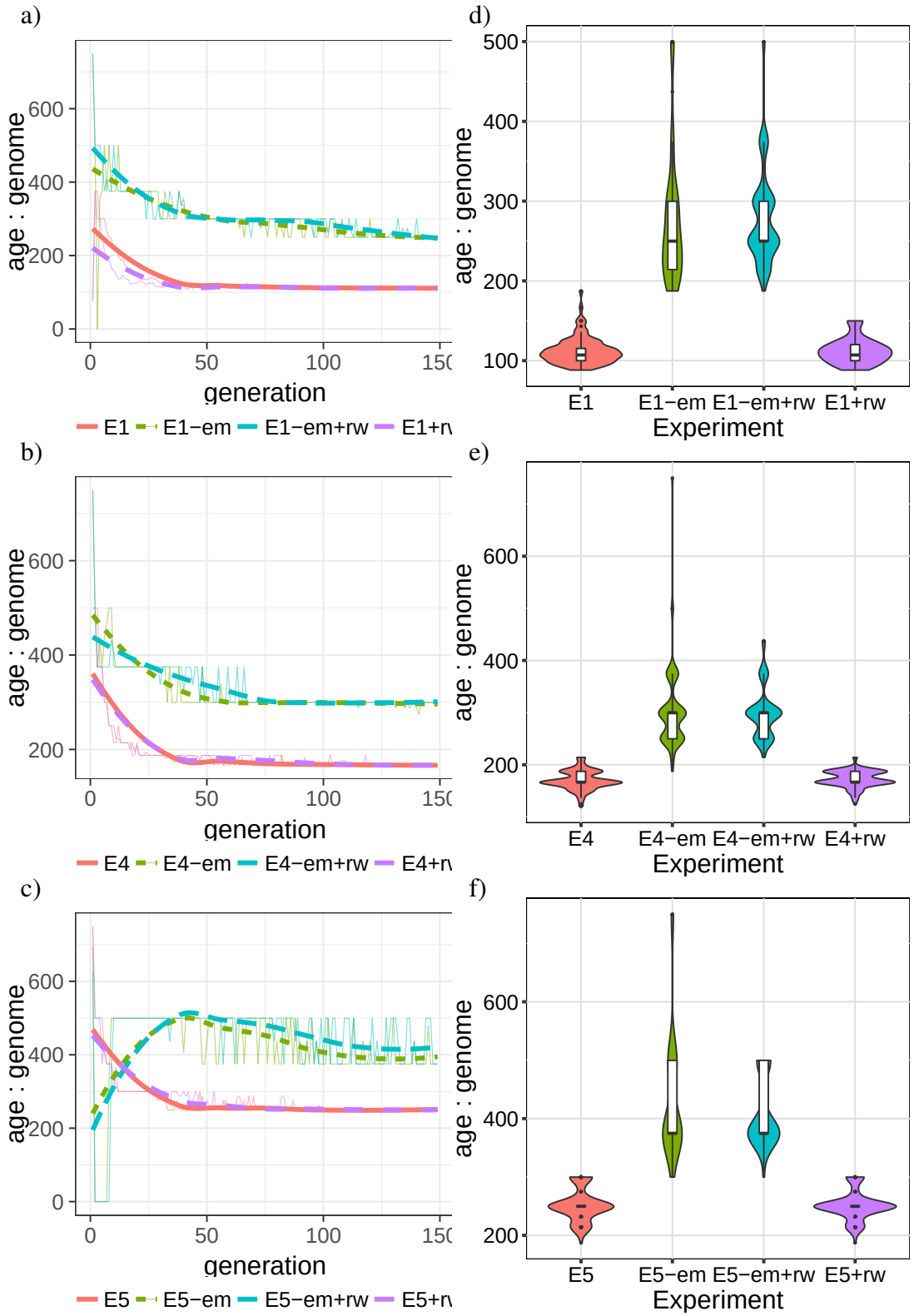


Fig. 4.2 Comparison of the median value for lifetime per unique received genome at the end of a generation between experiments, without (Ex,Ex+rw) and with the communication costs (Ex_{em}, Ex_{em}+rw). Violin and box plots showing the distribution of values over the last three generations of the experiment. Lines are smoothed to better show trends, with the raw data displayed in the same colour and thinner lines.

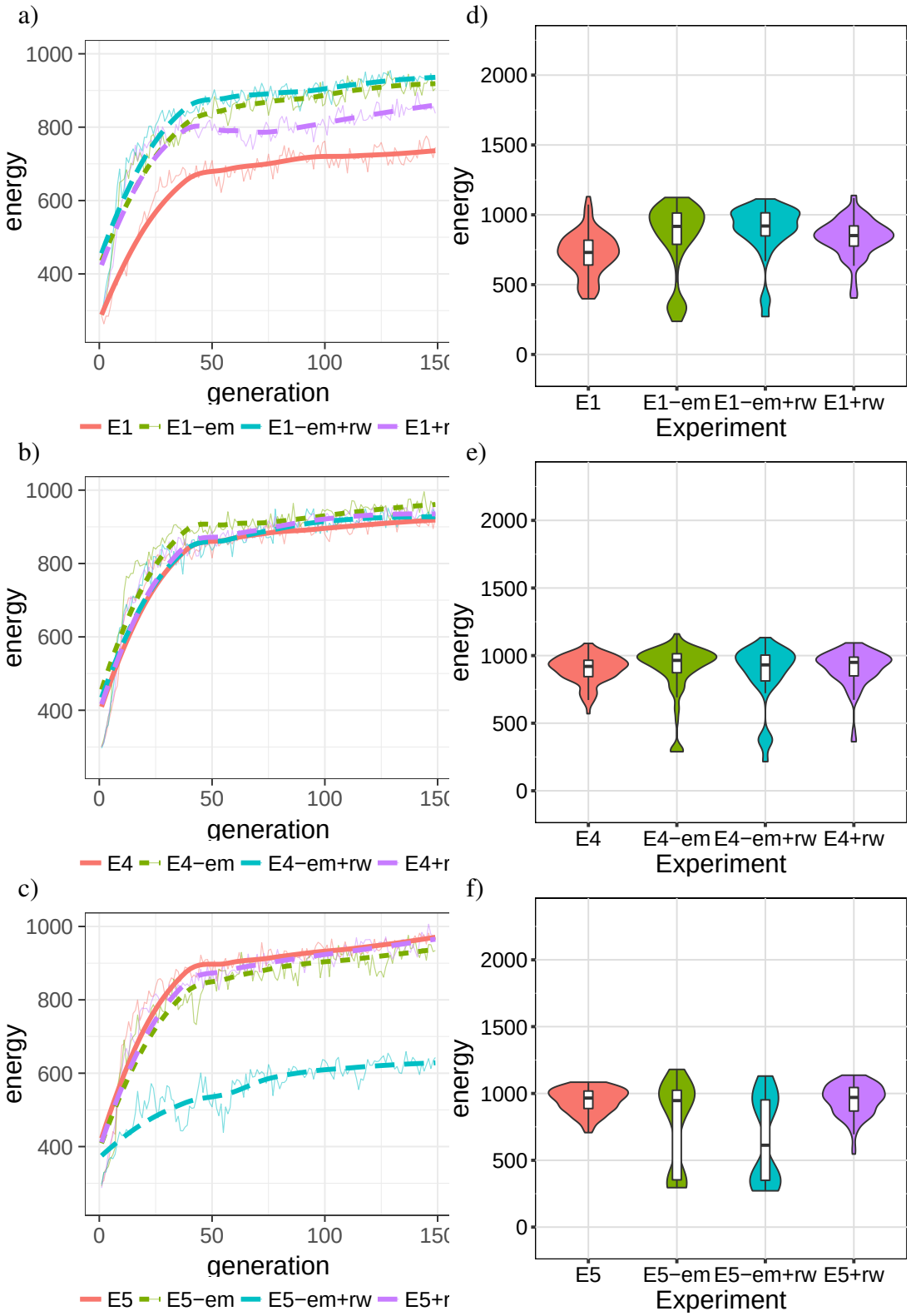


Fig. 4.3 Comparison of the median energy level at the end of a generation between experiments, without (E*x*,E*x*+rw) and with the communication costs (Ex_{em}, Ex_{em}+rw). Violin and box plots showing the distribution of values over the last three generations of the experiment. Lines are smoothed to better show trends, with the raw data displayed in the same colour and thinner lines.

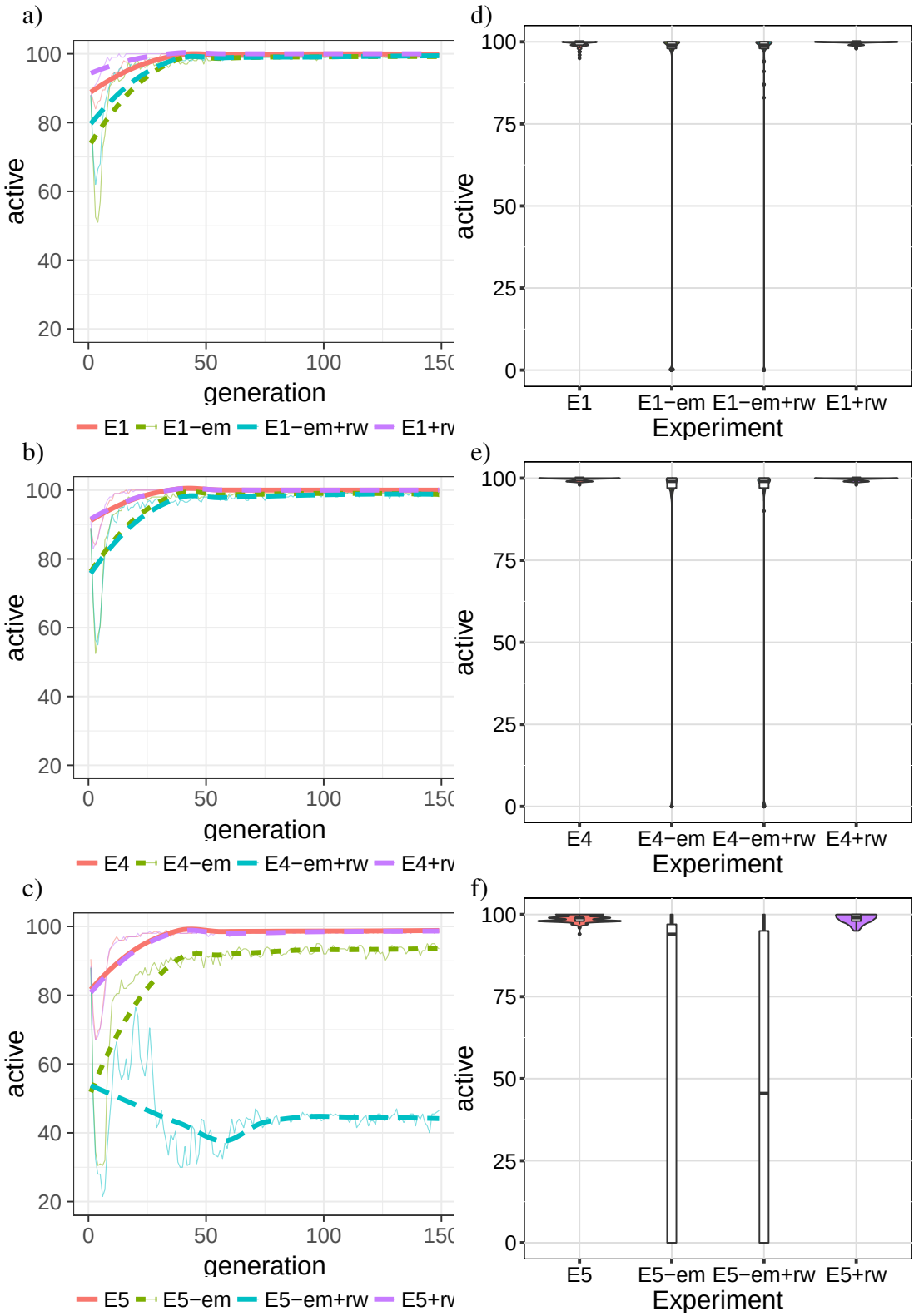


Fig. 4.4 Comparison of the median number of active robots at the end of a generation between experiments, without (E*x*,E*x*+rw) and with the communication costs (Ex_{em}, Ex_{em}+rw). Violin and box plots showing the distribution of values over the last three generations of the experiment. Lines are smoothed to better show trends, with the raw data displayed in the same colour and thinner lines.

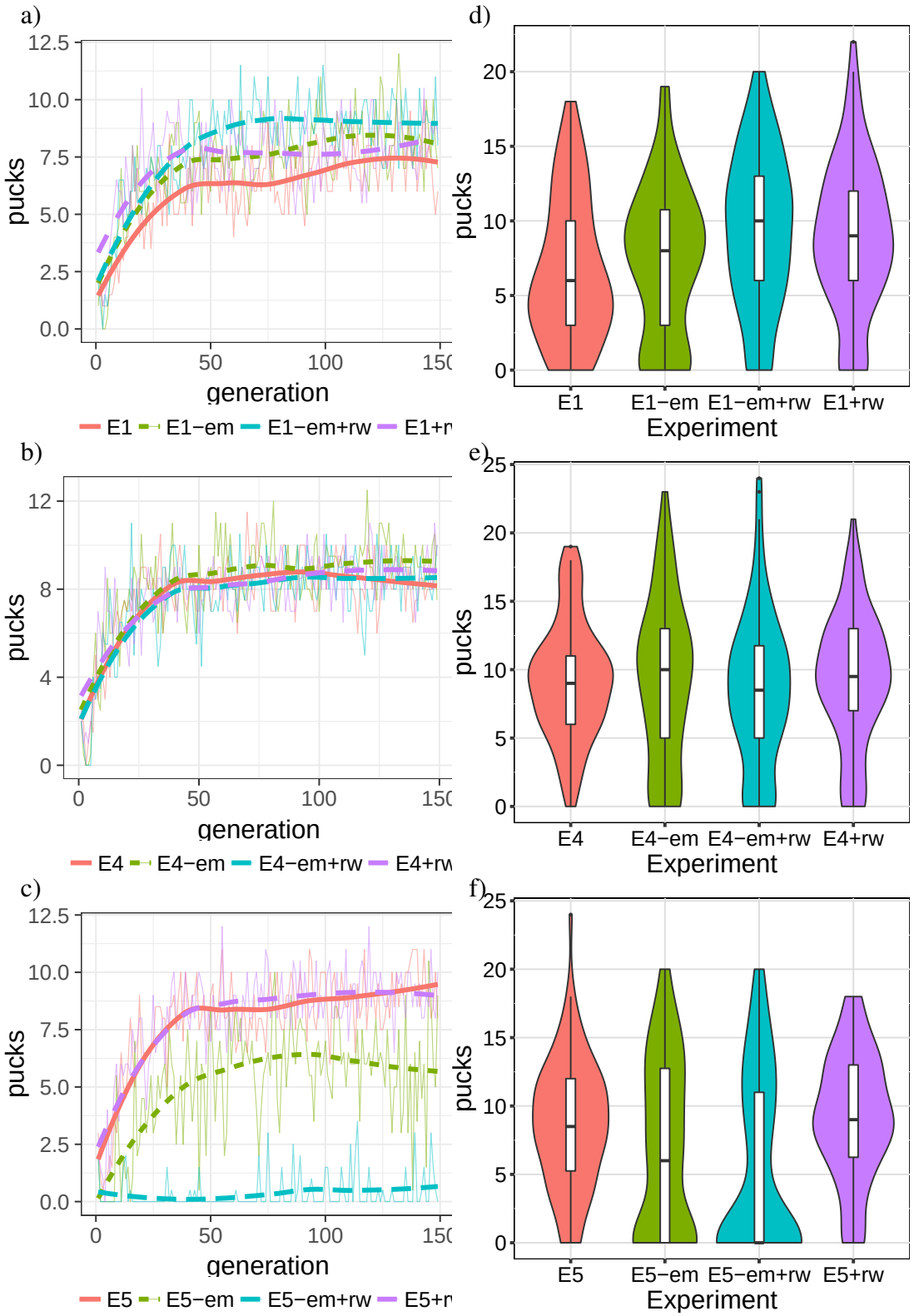


Fig. 4.5 Comparison of the median value for collected tokens at the end of a generation between experiments, without (E*x*,E*x*+rw) and with the communication costs (Ex_{em}, Ex_{em}+rw). Violin and box plots showing the distribution of values over the last three generations of the experiment. Lines are smoothed to better show trends, with the raw data displayed in the same colour and thinner lines.

4.8 Summary and Conclusion

The following comments can be made that summarise all experiments. Where claims are made, they are evidenced by data that is statistically significant as shown in the table in the previous section.

A new energy model based on the Free-Space Model was introduced to account for the cost of communication. Experimental results showed this exerts environmental pressure to implicitly select for genomes that maintain high energy levels. Comparing the method of varying the broadcasting based on fitness to mEDEA alone and with roulette-wheel genome selection shows that although marginal, the latter approaches outperform the mEDEA$_{rf}$ methods. Hence, hypothesis 4 as stated earlier in this chapter is not true. This is due to the fact that biasing the broadcasting based on fitness, which is directly linked to the energy level, partially reverses the effect of the environmental pressure.

The difference in active robots is largest for the experiment where the fitness measure is used for both mechanisms, varying broadcasting and the explicit selection. This effect is smaller for the two experiments E1$_{em}$ and E1$_{em}$+rw, where no or just the explicit selection mechanism is active. The additional environmental pressure of the added cost of communication leads to more efficient behaviour that avoids costly communication.

Although fewer robots are active at the end of the generation, it is argued that the method of biasing what to broadcast has a benefit as it clearly shows that robots spend less time collecting tokens. This is a crucial advantage if a task is added in the future, as more time could be allocated to the task than to maintaining energy levels.

The emergent behaviour of robots in the swarm have changed with the introduction of an energy-model that reflects more accurately the incurred energy costs in reality. It can be seen that the performance of an algorithm depends not just on its own parameters, but the external conditions in which the swarm is deployed. The next chapter explores

the relationship of the parameters defining the environment and the emergent behaviour of the swarm.

Chapter 5

Influence of the Environment on the Emergence of Behaviour

As discussed in chapter 3, a crucial trait of the robots is to be able to maintain the integrity of the swarm. The ability to maintain one's energy level is identified as a goal for individual robots. However, this specification might lead to the emergence of a different behaviour on the swarm level.

It is common in optimisation to explore the relationship between algorithmic parameters and fitness. However, evolutionary robotics adds an additional dimension in that it is not only the algorithm's parameters that influence the behaviour, but also *environmental* parameters. Especially when using an embodied online approach, the environment is the external driving force that shapes the evolutionary path. To function reliably, the algorithm should produce the same overall behaviour independent of the environmental conditions it is exposed to. Most research in the field to date has a shortfall in this regard, as often environmental parameters are chosen adhoc or deliberately to present favourable results.

5.1 Contribution

The work presented in this chapter is based on work presented in [121]. A copy of the publication can be found in the appendix (A.3)

The environment's influence on the emergence of behaviour is analysed in different environmental conditions. This is achieved through a map that represents the performance as a surface plot that further allows the identification of regions of interest. The contribution described in this chapter is the methodology by which this map is created and used to perform the analysis.

5.2 Introduction

Particularly in simulation, it is easy to arbitrarily select environmental parameters, such as the number of available energy sources or their corresponding energy-values. However, arbitrary choices can inadvertently create environments that have a major influence on the evolution of behaviour. For example, assume a researcher wishes to investigate whether individual learning speeds up environment-driven evolution: if an environment is created that has too much energy available then it is unlikely to exert sufficient pressure for individual learning to be beneficial or even emerge. Quantifying 'too much' (or 'too little') is of course difficult and in order to address this an analysis of an environment-driven distributed algorithm is analysed operating in a variable environment. In this context environments are defined by the concentration and availability of energy. These two parameters are varied in order to investigate the effect on the robot's behaviour.

Using a 3-dimensional visualisation, which is created from experimental results of the median-energy balance landscape for the algorithm mEDEA$_{rf}$, it can be shown that:

- the energy landscape contains three distinct regions: energy-poor, energy-neutral and energy-rich, as well as a 'dead-zone' in which robots cannot survive

- the energy-rich region is relatively large compared to other regions, but is very rugged
- that on the energy-neutral line, distinct behaviours evolve at different places along the line

It follows that the energy-neutral region provides the most obvious settings for conducting experimentation that aims to extend a robot's ability to survive or accomplish tasks.

5.3 Hypotheses

The following alternative hypotheses inform the experimental design and are tested through experimental investigation.

Hypothesis 5 *The specific configuration of an environment, defined by availability and concentration of energy, will affect the emergent behaviour at the end of the experiment, measured through the level of δEnergy maintained.*

Hypothesis 6 *The specific configuration of an environment, defined by availability and concentration of energy, will affect the emergent behaviour at the end of the experiment, measured through the number of broadcasts.*

5.4 Experiments

The goal of the experiments is to understand the energy landscape in terms of the median δEnergy of a robot in the population as a function of the energy concentration and distribution in the environment, defined by the two parameters: *count*, the number of energy tokens available, and *value*, the energy value of each token. Energy *tokens* are randomly scattered in the environment. If a robot moves over a token, its energy

Table 5.1 Simulation and experimental parameters for all experiments in this chapter.

Simulation parameters	
Arena size	1024 pixel by 1024 pixel
Max. robot lifetime	2500 iterations
Token re-spawn time	500 iterations
Sensor range	196 pixel
Variable Parameters	
Number of robots	50, 75, 100
Number of tokens (*count*)	0 - 1300 (in steps of 50)
Energy value per token (*value*)	0 - 1400 (in steps of 50)
Experimental parameters	
Number of runs	5
Max. iterations	375,000 (= 150x2500)
Start energy	500
Max. communication range $r_{\max}$	128 pixel

is increased by an amount E_{token}. The energy token disappears when consumed and reappears after a fixed amount of time later at a different random location.

5.4.1 Methodology

Similar to the previous chapters, all experiments are conducted in simulation using Roborobo by Bredeche et al. from [22]. The same algorithm mEDEA$_{rf}$ is used as in previous chapters, without biasing of broadcasting and with roulette-wheel selection as explicit selection method, using the communication energy model derived in the previous chapter (E1$_{em}$+rw).

Table 5.1 shows the ranges of values considered for each parameter. Parameters are set before the beginning of the experiment and remain fixed throughout. Each experiment was repeated over 5 independent runs. This number is rather low for a noisy application of this type, but was chosen to speed up computation due to the high number of experiments that had to be run in total. It is to be noted here again, that the focus of this analysis is not the evaluation of a set of (potentially) randomly chosen parameters, but the influence the environment has on the emergence of behaviours.

The energy model remains the same as used in experiments in the previous chapter. Referring to the derived energy model in section 4.4, equation 4.14 shows the change in energy at each simulation step, where n is the number of tokens that have been collected in that step.

$$E_i(t+1) = E_i(t) - E_{\text{living-i}} - E_{\text{com-i}} + (n_{\text{token-i}} \times E_{\text{token}}) \qquad \text{(4.14 revisited)}$$

There is a fixed cost of 'living' of 0.5 units per timestep, regardless of whether the robot moves or not. A robot moving consumes an amount of energy that is related to its rotational speed v_{rot}, translational speed v_{trans}, and their respective maximum values $v_{\text{rot-max}}$ and $v_{\text{trans-max}}$

$$E_{\text{living}} = 0.5 + \left(\frac{v_{\text{rot}}}{v_{\text{rot-max}}} + \frac{v_{\text{trans}}}{v_{\text{trans-max}}} \right) /4 \qquad \text{(4.13 revisited)}$$

5.4.2 Adaptation of mEDEA$_{rf}$ for Experiments

The algorithm mEDEA$_{rf}$ has been taken from experiments described in the previous chapters. However, certain changes have been made to the implementation of the neural network controller in order to simplify the experimental setup. The number of inputs has been lowered by removing inputs from robots that would pose a challenge to implement in real hardware.

The genome defines the weights of an Elman recurrent neural network (RNN) consisting of 16 sensory inputs, one bias node (feeding into the hidden layer) and 2 motor outputs (translational and rotational speeds). 8 ray-sensors are distributed around the robot's body. They detect the proximity to the nearest object and its type. The RNN has 1 hidden layer with 16 nodes, thus 322 weights are defined by the genome.

So far, robots have chosen a genome after a set amount of time, which leads to a synchronous switch of generations across the swarm. This has been changed to be asynchronous, in keeping with the paradigm of a distributed algorithm without central

control. If a robot runs out of energy and has an empty genome list it temporarily becomes inactive, thus reducing the population size. It remains stationary until it receives a new genome from a passing robot, at which point it starts a new lifetime. Thus at any time-step, each robot potentially has a different 'age'.

Due to the change in length of the genome, which impacts the values used in the derived model for communication costs (E_{com}), the associated values have to be recalculated. Table 5.2 lists those values calculated with the method introduced previously in 4.4.3.

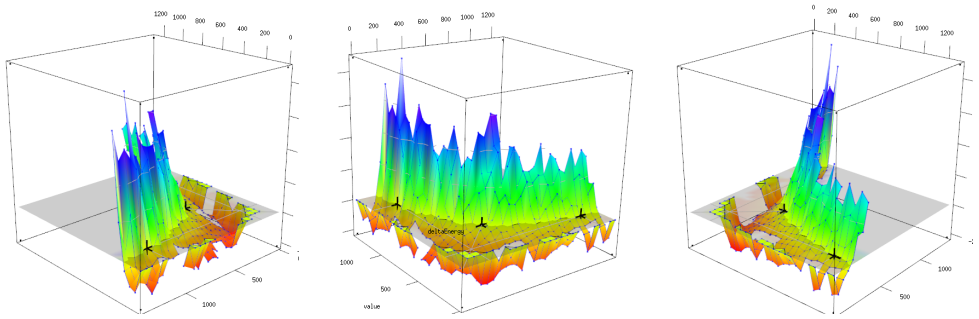
Table 5.2 Communication cost parameters adjusted to reflect change in genome length.

Parameter	Value
Genome length, n	10392 bit
$a_{\text{rx-Simulator}}$	0.02950
$a_{\text{tx-Simulator}}$	0.01334
$b_{\text{tx-amp-Simulator}}$	0.000594

5.5 Evaluation and Analysis

Data is gathered from the robots every 2500 iterations. Recall from section 5.4.2 that each robot chooses a new genome once it has depleted all its energy or reached the maximum lifetime, leading to asynchronous generation changes throughout the population. Hence, the data gathered at each interval represents a snapshot across robots of multiple ages and therefore does not necessarily capture the peak performance of each robot (i.e. it may include very 'young' robots). However, given that the goal of the experiment is to understand the interplay of the specific algorithm and environment under consideration, this is not a relevant factor.

Figure 5.1 shows three rotated 3-dimensional plots of surface obtained using 100 robots after 375,000 iterations. The x and y axes represent the *count* and *value* variables, while the z axis represent the median δ_E of the robot population over the last 2500



(a) rotated 90° right (b) centred (c) rotated 90° left

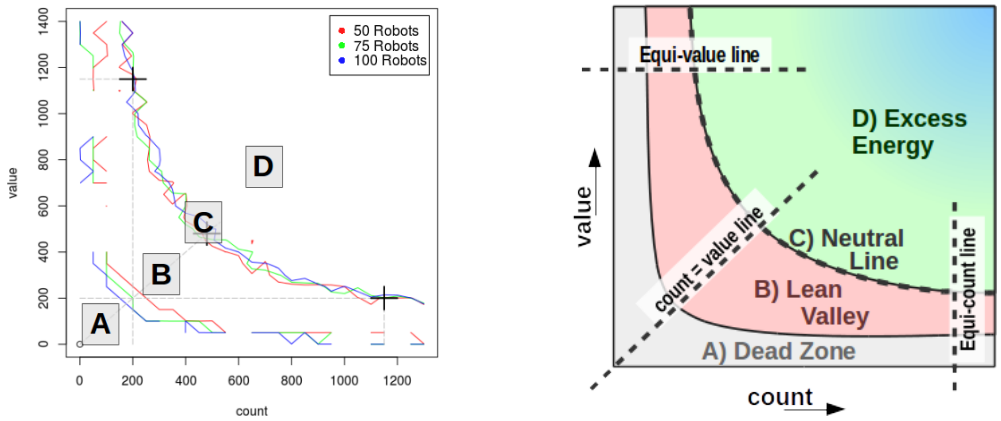
Fig. 5.1 View on the resulting surface from different angles. The figure was created by plotting the median δ_E of the last 2500 iterations of the experiment. The grey plain marks a value for δ_E of zero, at which point robots in an experiment have an energy balance of zero. In other words, the same amount of energy as they started the experiment with. An interactive 3D model can be found at [119]

iterations. The grey plain marks a value for δ_E of zero, at which point robots have an energy balance of zero, i.e. the same amount of energy as they started the experiment with. Three broad regions are noticeable: a large region in which the robots have positive δ_E (green and blue values above the grey plain), a region lying on the plain itself, and finally a region below the plain in which robots are spending more energy than they are collecting, i.e. $\delta_E < 0$. In order to explore this in more detail, a 2-dimensional top-down projection is shown in figure 5.2, obtained from populations of 50, 75 and 100 robots, and is discussed in detail below.

5.5.1 Different Performance Regions

Figure 5.2 shows clearly that the landscape is defined by four different regions:

A) Dead Zone: In this region, the environment does not provide enough energy for the algorithm to evolve a controller that can survive a full run. Low values for both parameters, *count* and *value*, result in the extinction of the whole robot population within a few generations. The random genomes that the controllers are initialised with generally result in a random spinning behaviour rather than movement. This random



(a) Real data plot (b) Cartoon

Fig. 5.2 Overview of the landscape (zero contours from 5.1), as plot of the real data on the left and as a cartoon version on the right. Four different regions are shown: A) Dead Zone, B) Lean Valley, C) Neutral Line, D) Excess Energy.

behaviour, combined with the lack of energy tokens in the immediate vicinity in which the robot is born, means that robots cannot survive given their inability to move.

B) Lean Valley (negative δ_E): This region starts at the edge of the dead zone that marks the point where there is just enough energy available that some robots survive until the end of the experiment, i.e. it marks the point where a robot has spent all its initial energy and started picking up tokens from the environment. Moving down towards the bottom of the valley, an increasing number of robots survive as there is more energy in the environment, with the corollary that each robot has less total energy — the energy available is shared between more robots. The bottom of the valley marks the minimum δ_E that still enables survival. Moving upwards out of the valley on the other side, robots gradually get better in both harvesting energy from the environment and managing their residual energy as a result of evolving better strategies. For example, good strategies optimise movement, or avoid moving towards tokens in which there are other robots close by.

C) Neutral Line (δ_E=0): This line marks the points in the environment where the environment provides exactly enough energy to enable a robot to maintain an energy

balance of zero, i.e. the costs of moving and communicating are just balanced by energy harvested.

D) Excess Energy ($\delta_E > 0$): In the final region, in which both *cost* and *value* are high, robots are able to locate more energy in the environment than is required to maintain their initial energy E_0, either due to the abundance of tokens or the high energy value of tokens.

The main contribution here is the insight gained into the algorithm's behaviour as a function of the parameter defining the environment, which is distilled out in the cartoon figure in 5.2b. Especially the identification of low and high performing regions that surround the narrow band of optimal conditions for the evolutionary process. Within this band, the *neutral line*, the conditions are so that the population of robots can keep operating indefinitely. Here, the environment supplies enough energy for the robots to just create the perfect conditions for creative strategies and behaviours to evolve that make the best use of the current environment.

Hoverd and Stepney [66] also found different environmental configurations in their simulation. They observed the influence of adding an energy model into their simulation of Turk's Sticky Feet and varying the level of available energy on the diversity of evolved creatures. Their findings are similar to those presented in this chapter. Similar to the *neutral line*, they found a 'critical' energy level that leads to the evolution of the widest diversity of creatures. Above this level, in high energy worlds with little evolutionary pressure they observed sessile behaviours. Below the 'critical' level either extinctions or predatory behaviours were observed, which leads to diminished diversity.

The approach introduced in this chapter allows different insights into the algorithm's performance than could be obtained if the algorithm was tweaked to outperform the competition on a single or set of tasks. It demonstrates the diversity of resulting behaviours obtained from the same set of algorithmic parameters and allows a better judgement about the reliability of the algorithm in different environments.

5.5.2 Environmental Influence on Behaviour

In order to properly understand the evolved behaviours that lead to the landscapes just described, a more detailed analysis is required. Figure 5.3 examines pairings of $(count, value)$ along the three dashed lines in 5.2, i.e. equivalent-$value$ (**a-b**), equivalent-$count$ (**c-d**) and the diagonal in which $count = value$ line (**e-f**). The figure shows boxplots of the δ_E values at specific pairings of $(count, value)$ and the ratio of genome broadcasts made to unique genomes received over a lifetime. The latter quantity leads to insights into behaviour as it relates to the number of *unique* robots encountered by an individual robot: a robot will broadcast indiscriminately to any robot in its range but will only collect unique genomes. At the equivalent-count and equivalent-value lines, we fix the parameter $count$ and $value$ respectively, and successively increase the other parameter in steps of 50.

Each graph shows five points. The first point on sub-figure (a) ($c50v150$) corresponds to a total energy E_{tot} that is the same as the first points on the two sub-figures (c) and (e) below[1]. For a specific value of E_{tot}, it is clear that high $value$ combined with low $count$ leads to robots that have increased δ_E when compared to robots with high $count$ but low $value$ (graph (a) compared to graph (e)). Robots must therefore evolve behaviours that enable them to seek out the rare but high-value tokens. These robots also have high broadcast:genome ratios, suggesting the robots are frequently coming into contact with the *same* robots. A possible explanation lies in the fact that the robots appear to travel in small groups, thus broadcasting continually to the same robots; the rare occurrence of tokens leads to many robots having to travel towards the same regions of the space. On the other hand, a high $count$ leads to robots that receive more unique genomes than in the high $value$ case: this is suggestive of a more random movement pattern that enables each robot to encounter more unique robots during its

[1] while this is exactly true for the first and third rows, in the middle row which represents equal count/value it is necessary to approximate.

lifetime. In this case there is low selection pressure to evolve focused movement due to the abundance of tokens.

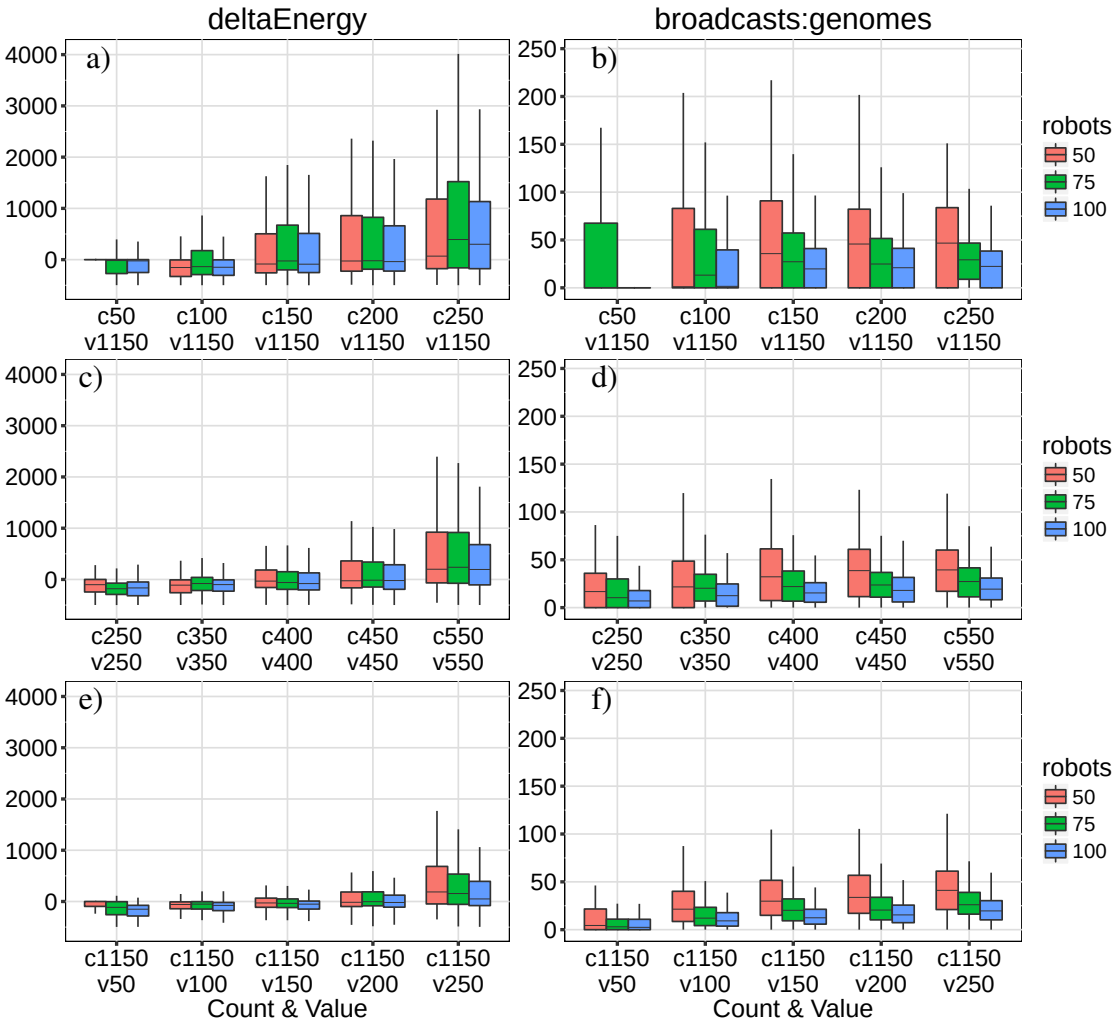


Fig. 5.3 Cuts through different parts of the landscape. Points towards different behaviours in terms of exploration. a-b) *value* = 1150, vary *count*; c-d) *count* = *value*; e-f) *count* =1150, vary *value*.

5.5.3 Behaviours in the Neutral Region

The energy neutral region is of greatest interest for researchers wishing to conduct research moving beyond genetic evolution of survival, for example, using individual or social learning [59] or task-driven research [49]. In this region, on the one hand, robots are able to survive, while on the other, the environment does not *over*-provide, ensuring that there is scope for robots to learn novel behaviours. Three specific points

Table 5.3 The environmental configurations sought out for all future experiments.

Number of tokens	*Value per token*	*Description*
200	1150	High competition environment
500	500	Comfortable environment
1150	200	Cluttered/abundant environment

Table 5.4 Results obtained at three configurations within the neutral region.

		Robots								
		50			75			100		
Count	Value	Age	Age / Genome	Brodcasts / Genome	Age	Age / Genome	Brodcasts / Genome	Age	Age / Genome	Brodcasts / Genome
200	1150	*770*	*107.94*	*46.61*	*767.5*	*63.86*	*28.99*	*667*	*49.97*	*21.18*
500	500	*1026.5*	*86.12*	*39.11*	*1038.5*	*52.39*	*25.93*	*928.5*	*41.26*	*19.67*
1150	200	*1173.5*	*79.83*	*33.43*	*1093*	*47.39*	*21.95*	*1059*	*36.52*	*15.49*

within this region are further investigated where there is approximately the *same* amount of energy available in the environment (table 5.4). The table shows the median age increases with increasing *count* — it is easier to maintain sufficient energy to survive as availability increases. The lower median observed at low *count* reflects the fact that many robots do not survive for very long. The time to find a new unique genome (age:genome) is shortest at high *count*, reflecting frequent encounters with novel robots. Broadcast:genomes is highest at low *count* as observed in the previous section. All three configurations lead to the same energy balance of 0, but diverse behaviours result in the gain in energy being offset by movement and broadcasting in each case.

5.5.4 Results of a Different Environment

The environments considered so far only contain one type of energy token. A straightforward way to introduce a significant change, is to introduce a different type of energy token. By changing the composition of the environment it is shown how crucial an analysis of the resulting landscape is.

Using the method described above, a new set of experiments is conducted in an environment parameterised by two variables: the *number* of energy tokens available,

Table 5.5 Environmental configurations: description refers to the prevalence of energy tokens within the environment.

Number of tokens	*Value per token*	*Description*
300	1150	Scarce
625	625	Balanced
1150	425	Abundant

and the *value* of the energy token. In each environment tested, there are n positive tokens with value v, and n negative tokens with value -400, which is 80% of a robot's initial energy. To reduce the amount of experiments, all environments are only used with 100 robots, compared to the previous setting where 50, 75 and 100 robots were used. All other parameters remain exactly the same as in the previous experiments and are listed in table 5.1.

The delta-energy δ_E, i.e. difference between start and end energy is recorded for multiple points in the parameter space, resulting in the plot shown in figure 5.4. The three crosses off the line toward the axes mark the identified environments of the previous experiments. The crosses on the line mark the environmental configurations where the extension of the equi-value, equi-count and count=value lines cross the neutral line. From the shift of the crosses along the lines it is evident that the environment has changed in comparison to the previously considered environment.

From this plot, we identify three points to conduct experiments along the *energy neutral line*, i.e the region in which robots expend as much energy as they acquire. This represents a region in which selection-pressure from the environment to survive is neither too small or too large to mask the behaviours we are interested in investigating. The points identified are specified in table 5.5.

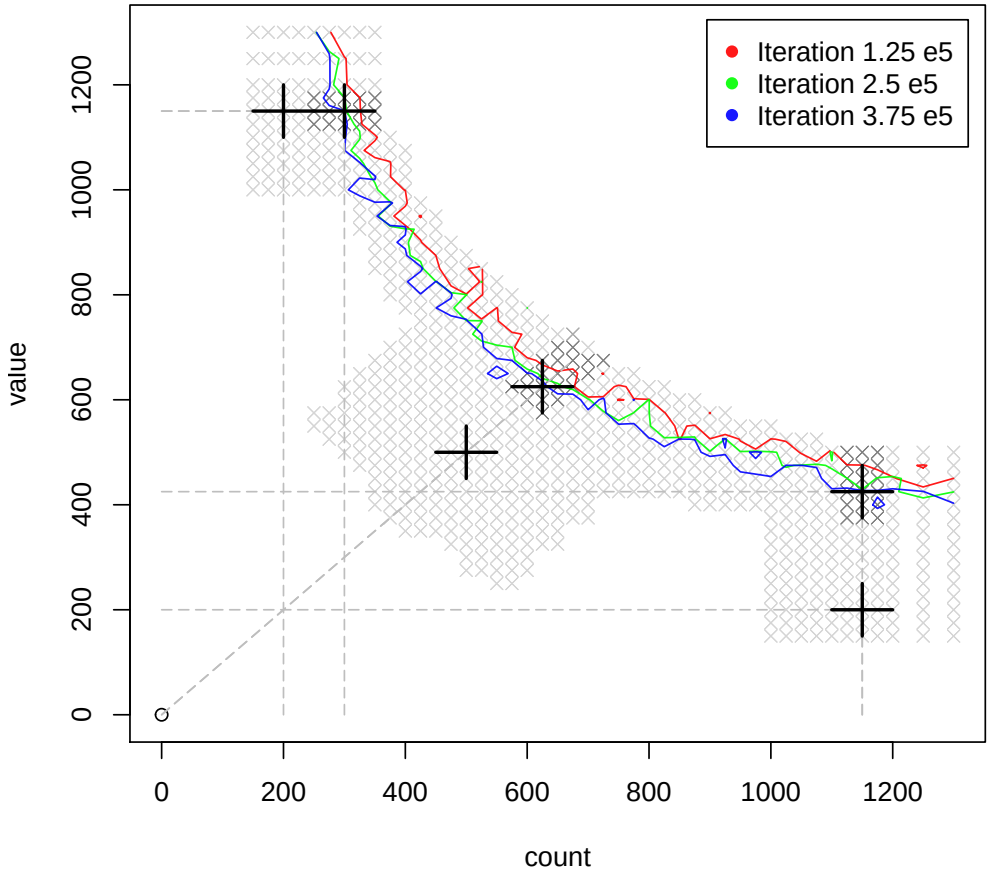


Fig. 5.4 Cuts through the surface landscape of experiments with 100 robots in environments that differ in availability and concentration of energy, and contain equal amounts of positive and negative tokens. Environments are defined by amount (*count*) and *value* per token, with a constant value of -400 for negative tokens. The coloured lines show the Neutral Line, the line where the surface plot crosses a plain drawn at delta-energy (δ_E)=0, averaged over 2500 iterations at different stages of the experimental run. Red, green and blue ending at iteration 125,000, 250,000 and 375,000 respectively. The three crosses, located just off the line, mark the identified environments of the previous experiments. The crosses, located on top of the line, mark the environmental configurations where the extension of the *equi-value*, *equi-count* and *count=value* lines cross the Neutral Line.

5.6 Summary and Conclusion

This chapter introduced a novel methodology to create performance maps for evolutionary robotics algorithms as functions of environmental parameters. The method is used to analyse the performance map that results from running an environment-driven evolutionary algorithm (mEDEA$_{rf}$) in an environment that is parameterised by two values that control the distribution of energy in the environment. Adjusting the availability and value of energy tokens results in the evolution of a range of different behaviours. Rather than arbitrarily selecting parameters in which to study evolution, it is suggested that it is vital to understand *how* these choices will direct evolution, by changing the selection pressure exerted by the environment.

Three distinct regions are observed in which the final energy balance can be negative, neutral, or positive. A fourth region is found in which robots cannot survive. The energy neutral region appears to be a good region in which to undertake experiments. It provides an environment in which robots are able to survive, enabling experimentation, while at the same time, rewarding new behaviours which are able to more efficiently harness energy from the environment. It is clear that the environment plays a key role in influencing what kind of behaviours emerge, in that it is not the total amount of energy available that matters but also the manner in which it is spread.

Introducing a change into the experimental setting showed a shift in the identified regions. From this it is evident that such analysis is beneficial to identify regions of interest.

Chapter 6

Influence of the Environment on the Benefit of Lifetime Adaptation

In the previous chapters adaptation was achieved through evolution alone. However, the time robots spent evaluating genomes while moving through the environment can be used to further adapt the individual. Lifetime adaptation in the form of individual learning has been used many times to augment evolutionary algorithms in evolutionary swarm robotics. Here, the benefits are not in question. The goal is to explore the relationship between the environment and the value of different types of adaptation.

6.1 Contribution

The work presented in this chapter has partly been published in the Genetic and Evolutionary Computation Conference (GECCO 2017) [122]. A copy of the publication can be found in the appendix (A.4).

An individual learning mechanism is developed that doesn't require any *a priori* knowledge of the environment and can help the evolutionary process to adapt. It is investigated how environmental parameters (token count, value and the rate of change)

influence the effectiveness of the adaptation mechanism and how the nature of the adaptation mechanism influences performance in different environments.

The results show that there is a clear link between environmental conditions, specifically the rate of change and the availability of learning opportunities, and the effectiveness of different adaptation mechanisms. Further, the incentive to learn in form of reward or punishment, in other words, the environmental pressure which drives the evolutionary process, must be high enough to make any form of adaptation worthwhile.

6.2 Introduction

Many environments will be unknown to the designer *a priori* and are potentially dynamic, the swarm must be able to continuously adapt its behaviour to ensure it both maintains sufficient energy to survive and to successfully perform tasks.

The importance of being able to adapt over time has been a subject of research within Evolutionary Robotics for some time, e.g. [130]. In previous chapters, evolution was the sole engine for adaptation. In *evolutionary* adaptation, information encoded on the genome adapts through selection and reproductive operators over many generations. To evaluate a genome, the robot carrying the genome needs to experience and interact with the world to derive a fitness value. This time can be used by the robot to further adapt its behaviour using *individual* learning.

The goal here is to investigate the interplay between evolution, individual learning and environmental characteristics in depth. A swarm is considered that undergoes distributed evolution of a neural-network based controller, and is augmented with an individual learning mechanism: this modifies the information gleaned from the environment and fed to the controller over the lifetime of a robot. Specifically, the considered swarm is operating in an environment that is unknown *a priori*, and in which robots must learn relative values of positive and negative energy tokens. Each environment contains n positive and n negative energy tokens. Positive tokens increase

the robot's energy by v units of energy when collected, while negative ones reduce it by a fixed amount. As n and v vary, each environment presents different opportunities for learning in that either there are a small number of high value tokens or a large number of low value tokens. In addition, tokens change their nature across 'seasons', i.e. tokens of a specific colour switch value from negative to positive on a cyclical basis. This forces the swarm to have to re-learn the effect of any given colour of token every season. Various settings for individual learning are investigated in which the learning mechanism is either fixed or has components that can be simultaneously evolved.

The distributed evolutionary algorithm mEDEA$_{rf}$ from chapter 3 is augmented with mechanisms for individual learning in order to conduct experiments. Note that the goal is not to propose a novel method of either individual learning or evolutionary adaptation, but to explore the relationship between the environment and value of different types of adaptation. To this end, it is investigated how environmental parameters (token count, value and the rate of seasonal change) influence the effectiveness of the adaptation mechanisms and how the nature of the adaptation mechanism influences performance in different environments.

6.3 Designing the Lifetime Adaptation Mechanism

6.3.1 Scenario Overview

A swarm operates in a simulated environment in which there are two types of coloured tokens: driving over one colour increases robots' energy while the other decreases it. Robots should learn to avoid negative tokens. However, a "seasonal" change is imposed where the value of tokens is reversed, i.e. red becomes positive and blue negative or vice versa. A robot must thus adapt any previously evolved behaviour. All robots in the swarm evolve a neural network that controls their behaviour through the distributed evolutionary algorithm $mEDEA_{rf}$ (3.3). The algorithm $mEDEA_{rf}$ is

used without biasing of broadcasting and with roulette-wheel selection as explicit the selection method (E1$_{em}$+rw), using the communication energy model derived in the previous chapter. In addition, they can exploit an individual adaptation mechanism that can potentially learn the *current* value of a given token colour. This information modifies an input to the evolved neural network. A number of types of individual learning are investigated in which some components of the learning mechanism can be heritable, fixed or absent.

Here, the same simulation environment (Roborobo by Bredeche *et al.* [22]) as in the previous chapters is used. To briefly recap, the simulated robots have 8 ray-sensors distributed around the body and detect proximity to the nearest object and its type. Each robot is controlled by an evolved Elman recurrent neural network (RNN). The network has 16 sensory inputs and 2 motor outputs (translational and rotational speeds). The 16 inputs comprise of two pieces of information for each of the 8 ray-sensors: proximity and whether or not the object is an energy token. Although the colour/type of the object is also detected by the robot, it is not fed into the RNN as an input, but only used in the adaptation mechanism[1].

6.3.2 The Individual Learning Mechanism

The neural network described above has a set of binary inputs (one for each ray-sensor) that denote the presence (1) or absence (0) of a token (independent of its type). Therefore, in an environment in which there are multiple types of tokens, the only way for an individual to distinguish between them is to pick them up and observe the change in energy. If the environment in which the robot operates is known *a priori*, then clearly, the neural network could be designed in order to include relevant information about each token type. However, if the environment is unknown, then the robot must learn to adapt to the different types and values of tokens it may encounter.

[1]The information cannot be encoded directly to the network without *a priori* knowledge of the number of potential colours.

The lifetime-learning mechanism proposed here adds qualitative information about the detected token instead of just a binary indicator. It extends a robot's ability from previously only detecting the presence of a token to being able to distinguish between different types and their relative importance. This is achieved by adding a multiplier to each of the NN's token detection inputs. Like a set of filters, the appropriate multiplier value is chosen based on the detected token type. The multiplier for each type of token can be a continuous value in the range of -1 and 1 and therefore allow the input to carry more information than just binary.

Every time a previously unseen type of token is encountered (detected by a sensor ray or through consumption), a new value is added to the set. As tokens are usually[2] detected before they are consumed, no information regarding the token's value is known: the robot therefore randomly initialises a value to associate with the type (x) of detected token. Following consumption, the resulting change of energy is detected by the robot and its learning mechanism can modify the corresponding multiplier value (m_x).

All multiplier values are adjusted every time a token is consumed according to equation 6.1:

$$m_x{}' = m_x + LS \times \left(LR - \frac{C_x}{C_{\text{total}}} \right) \times \left(\frac{V_x}{V_{\text{max}} - V_{\text{min}}} \right) \tag{6.1}$$

m_x is the current value for the multiplier for type x; C_x is the number of tokens of type x collected; C_{total} is the total number of all tokens collected; V_x is the value of the token that has just been consumed and is therefore now known to the robot (being equivalent to the change in energy); V_{max} and V_{min} define the minimum and maximum values of all tokens encountered so far. LR is a learning rate that controls the magnitude of the change, and LS is either -1 or $+1$ and simply inverts the direction of change; this is required to adjust the learning mechanism to the internal value notation of the neural

[2] As robots only have a discrete number of ray-sensors and not a full field of detection around their body, only objects crossing a sensor ray can be detected. This can lead to a situation in which a robot drives over a token before any of the ray-sensors detect it.

network and is adapted via evolution alongside the genome. The learning mechanism is shown in algorithm 4.

```
1  if token_x is unknown then
2  |  multipliers.add(token_x);
3  end
4  if token_x is consumed then
5  |  tokenCounter_x.update(token_x);
6  |  totalTokenCount.update();
7  |  tokenValue_x.update(δ_E(t) − δ_E(t − 1));
8  |  totalValueRange.update();
9  |  for m_x in multipliers do
10 |  |  m_x.update();                                  // eq. 6.1
11 |  end
12 end
```

Algorithm 4: Pseudo code of the steps carried out to update all multipliers every time a token is encountered.

Three factors influence the learning mechanism: the initial value assigned to a token V_x, the learning rate LR and the associated sign LS. These factors can be randomly assigned, fixed to some specific value or can themselves be subject to evolution. Allowing the learning sign to co-evolve enables the learning mechanism to self-adapt to the internal value convention of the neural network. Finally, enabling the robot to evolve an appropriate starting value for each type of token based on its experience may speed-up learning in some circumstances. Even though token values change over seasons, inheriting a good starting value may be beneficial and, presumably, dependent on the rate of change of the environment. This algorithm makes use of the Baldwin Effect [2], by intertwining the learning mechanism with the evolutionary process and making the adaptable parameters heritable.

Note that in no case is any Lamarkian evolution used, i.e. although the multiplier starting values are adapted over the course of a lifetime, they are *never* written back to the genome and are therefore not inherited.

6.4 Hypotheses

The following alternative hypotheses inform the experimental design and are tested through experimental investigation.

Hypothesis 7 *The effectiveness of different individual adaptation settings is influenced by the parameters of the environment (token count, token value).*

Hypothesis 8 *The rate of change of a given environment influences the effectiveness of the individual adaptation mechanism.*

Hypothesis 9 *The nature of the individual adaptation mechanism influences performance in different environments.*

6.5 Experiments

Experiments are carried out using mEDEA$_{rf}$ as described in 3.4. Experimental and simulation parameters are given in table 6.1. Parameters associated with the learning mechanism are given in table 6.2. The values for LR_{initial} and LR_{max} were selected following limited empirical exploration.

Table 6.1 Simulation and experimental parameters for all experiments in this chapter.

Simulation parameters	
Arena size	1024 px $\times$ 1024 px
Max. robot lifetime	2500 iterations
Token re-spawn time	500 iterations
Sensor range	196 pixel
Max. communication range r_{max}	128 pixel
Experimental parameters	
Number of independent runs	30
Number of robots	100
Max. iterations	1,000,000
Start energy	500

Table 6.2 Learning parameters with initial values and ranges in which they can change during runtime of the experiment. Method of adaptation (through evolution or lifetime-learning) depends on the experiment.

Parameter	**Init. Value**	**Value Range**
Learning rate, LR	1.02	$[LR_{\min}, LR_{\max}]$
Minimum LR, $LR_{\min}$	1	fixed
Maximum LR, $LR_{\max}$	1.5	fixed
Multiplier of type x, m_x	random	$[-1,1]$
Inherited multiplier, im_x	random	$[-1,1]$
Learning sign, LS	random	$[-1,1]$

6.5.1 Variations of the Adaptation Mechanism

Four variants of the adaptation mechanism are investigated as detailed below.

- **Evo**: evolves multiplier values but has no adaptation during lifetime
- **IL**: pure lifetime learning with random initialisation of multiplier values
- **Evo + IL**: evolves the LR, LS and the multiplier values, and also adjusts the latter during lifetime
- **Baseline**: adaptation mechanism is disabled and robots can only detect the presence of a token but not its type

Note that in the baseline experiments, the multipliers for all tokens have a fixed value of 1 and therefore the robots cannot distinguish between tokens of different types. Table 6.3 lists the heritability of parts of the adaptation mechanism for the four variants of the algorithm and shows how multipliers are initialised when a genome is deployed in a robot. These four variants of the algorithm are investigated in conjunction with the three environments described in the following section.

6.5.2 Selecting Environmental Configurations

In the previous chapter in section 5.5.4, an environment was established that is suitable for an individual learning experiment. Having two different types of tokens provides the

Table 6.3 Learning scenarios investigated showing heritability of information.

	Initial Value of Multiplier	LR	LS
Baseline	1 (all tokens)	none	n/a
IL	random	fixed	evolved
EVO	evolved	none	n/a
EVO+IL	evolved	evolved	evolved

opportunity to learn to distinguish between them. An environment is parameterised by two variables: the *number* of energy tokens available and the *value* of the energy token. In each environment tested, there are n positive tokens with value v, and n negative tokens with value -400, which is 80% of a robot's initial energy.

The delta-energy δ_E, i.e. the difference between start and end energy is recorded for multiple points in the parameter space, resulting in the plot shown in figure 6.1. From this plot, three points have been identified to conduct experiments along the *energy neutral line*, i.e the region in which the robot expends as much energy as it acquires. This represents a region in which selection-pressure from the environment to survive is neither too small or too large to mask the behaviours under investigation. What's more, individual learning has the potential to improve evolved behaviours, but is not essential. The points that were identified in the previous chapter in 5.5.4 are listed in table 6.4.

Table 6.4 Environmental configurations: description refers to the prevalence of energy tokens within the environment.

Number of tokens	*Value per token*	*Description*
300	1150	Scarce
625	625	Balanced
1150	425	Abundant

6.5.3 Sets of Experiments

An experiment is defined by a tuple <*environment, seasonal change rate, algorithm*>. Three environments are specified in section 6.5.2. Three different rates of seasonal

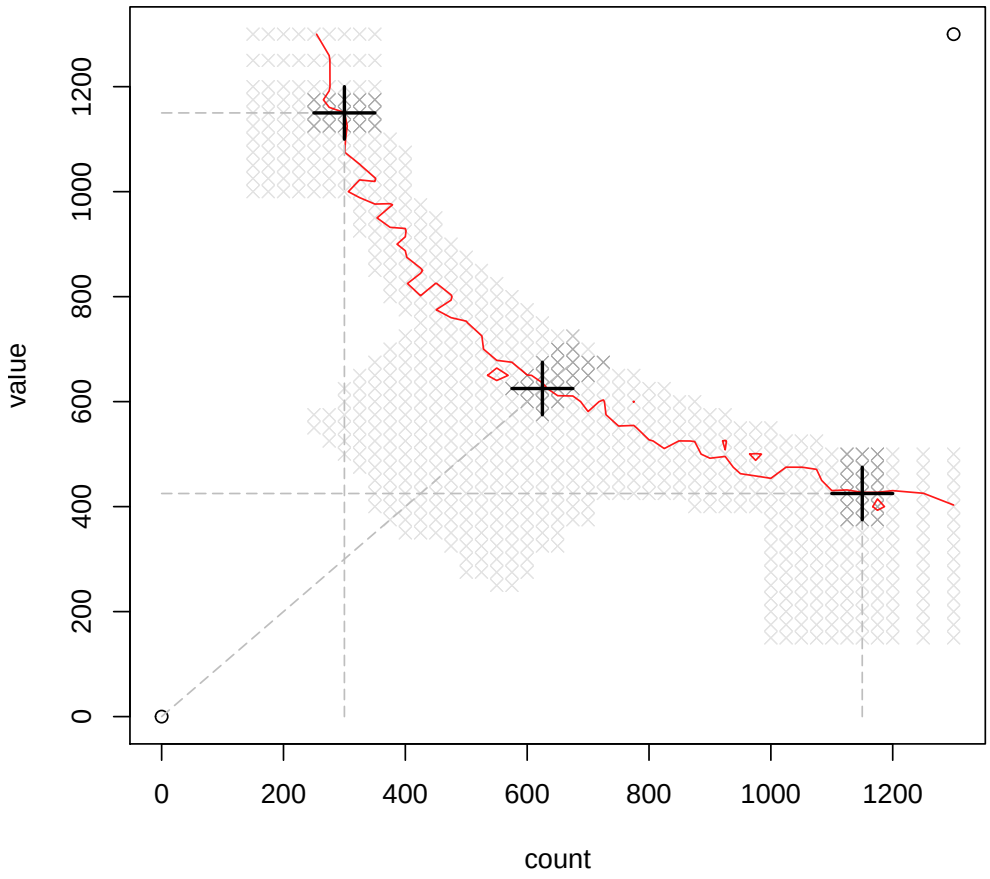


Fig. 6.1 Top down view of the surface plot created using the method introduced in the previous chapter. The red line shows the Neutral Line, the line where the surface plot crosses a plane drawn at delta-energy (δ_E)=0.

change are investigated: 0 (no change, i.e. static environment), every 5000 iterations and every 15000 iterations. Note that the maximum lifetime of a genome before it is replaced is 2500 iterations, so every robot should go through at least one evolutionary generation during the shorter (5000 iterations) season, and at least 5 generations during the 15000 season. In practice, as robots tend to die before their maximum lifetime, more evolutionary cycles are likely to occur.

Four algorithms are investigated as detailed in table 6.3. Thus, in total 36 (=3x3x4) experiments are conducted. In each experiment, the *totalTokenRatio* is recorded at the end of the season. This value is the ratio of the number of collected tokens with positive values divided by the sum of all collected tokens within that season. A ratio of 0.5

shows that an equal amount of positive and negative tokens were collected, below 0.5 more negative and above more positive tokens, respectively.

6.5.4 Energy Model and Updated Communication Costs

Referring to the derived energy model in section 4.4 in an earlier chapter, equation 4.14 shows the change in energy at each simulation step, where n is the number of tokens that have been collected in that step.

$$E(t+1) = E(t) - E_{\text{living}} - E_{\text{com}} + (n_{\text{token}} \times E_{\text{token}}) \qquad \text{(4.14 revisited)}$$

The energy model remains the same as used in the experiments in the previous chapter. To recap from 4.3, there is a fixed cost of 'living' of 0.5 units per timestep, regardless of whether the robot moves or not. A robot moving consumes an amount of energy that is related to its rotational speed v_{rot}, translational speed v_{trans}, and their respective maximum values $v_{\text{rot-max}}$ and $v_{\text{trans-max}}$

$$E_{\text{living}} = 0.5 + \left(\frac{v_{\text{rot}}}{v_{\text{rot-max}}} + \frac{v_{\text{trans}}}{v_{\text{trans-max}}} \right) / 4 \qquad \text{(4.13 revisited)}$$

However, due to the additional genes of the adaptation algorithm the genomes lengths have changed, which impacts the values used in the derived model for communication costs (E_{com}). Recall from 4.4 that the energy required for transmitting and receiving is based on the length of a message (here, the length of the transmitted genome). The genome now includes 1335 byte of information (322 neural network weights, sigma, fitness, age, genome ID, robot ID, learning rate (*LR*), learning sign (*LS*), and the inherited multiplier for type x (im_x) of 4 byte each). The parameters im_x, *LS* and *LR* are always included in the message, but only used in experiments where the learning parameters are inherited.

Using the following equations derived in a previous chapter (section 4.4.3) and the values from the data sheet in table 4.1, the adjusted values listed in table 6.5 can be calculated.

$$\begin{aligned} E_{\text{rx}} &= n_{\text{bit}} \times E_{\text{rx-elec}} \\ &= a_{\text{rx-Simulator}} && \text{(4.5 revisited)} \\ E_{\text{tx}}(d) &= n_{\text{bit}} \times E_{\text{tx-elec}} + n_{\text{bit}} \times \varepsilon_{\text{tx-amp}} \times d^2 \\ &= a_{\text{tx-Simulator}} + b_{\text{tx-amp-Simulator}} \times d^2 && \text{(4.6 revisited)} \end{aligned}$$

Table 6.5 Communication cost parameters.

Parameter	**Value**
Genome length, n	10774 bit
$a_{\text{rx-Simulator}}$	0.03050
$a_{\text{tx-Simulator}}$	0.01379
$b_{\text{tx-amp-Simulator}}$	0.000614

6.6 Evaluation and Analysis

6.6.1 Methodology

Following 30 runs of each experiment, statistical analysis was conducted based on the method in [107], using a significance level of 5%. The distributions of two results were checked using a Shapiro-Wilk test. If one of the results followed a non-Gaussian distribution then the p-value is determined using a Kruskal-Wallis rank sum test. Otherwise, the homogeneity of variance of the two results is performed using a Levene's test for homogeneity of variance. For unequal variances the p-value is determined using a Welch test, otherwise using an ANOVA test.

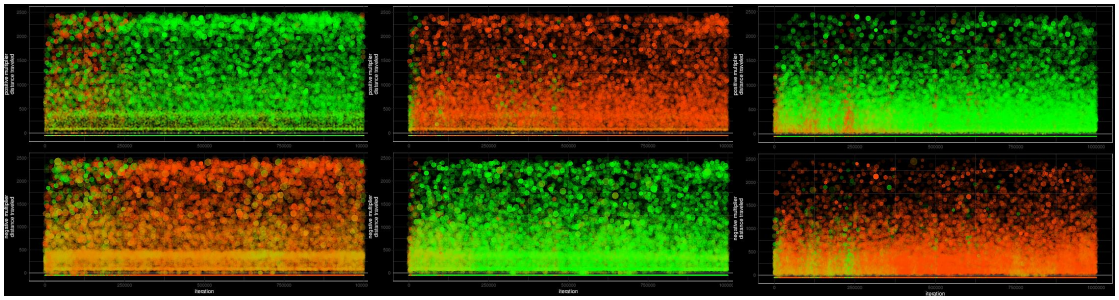
6.6.2 Adaptation Mechanism

Figure 6.2 shows the algorithm variation EVO + IL deployed in three different environments (*scarce*: count = 300, value = 1150; *balanced*: count = 625, value = 625; *abundant*: count = 1150, value = 425) in a static setting and at seasonal change rates of 5k, 15k and 30k. The plots for all application variants of the algorithm can be found in appendix B.

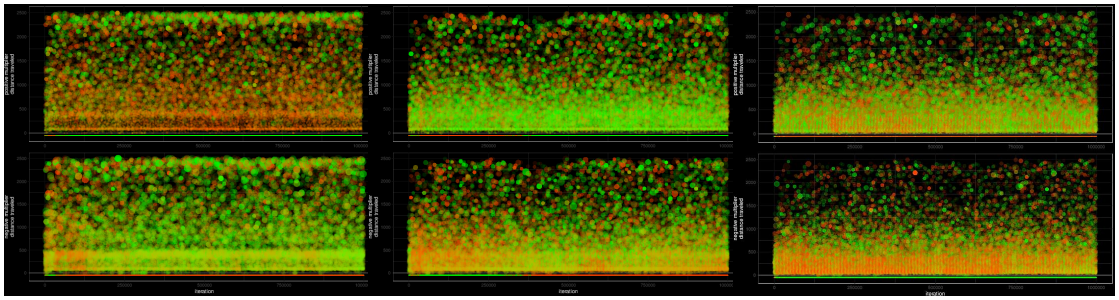
As these plots are quite dense and hard to digest, a larger and more detailed version is shown in 6.3. The plots show data as follows:

Data plotted in the individual scatter plots represents 10% of the population (randomly sampled) and is taken from a single randomly selected run of the experiment. Each of the two sub-graphs is a scatter plot, showing the value of the positive and negative multipliers respectively. Note, the names "positive", for the upper sub-graph, and "negative", for the lower sub-graph, refer to the initial value of the corresponding token at the beginning of the experiment. Every genome has a data point on either of the two sub-graphs. The values end-of-lifetime (x position), distance travelled (y-value) and reproductive success (transparency; measured in number of offspring) are the same in each sub-graph. The value of the multiplier (colour; a continues scale from -1 = red to +1 = green) is unique to the corresponding sub-graph. The size of the dot shows the adaptation success of the individual genome: positive tokens collected per lifetime in the upper sub-graph and the inverse ratio for negative tokens in the lower sub-graph. The red/green lines below the y-axes is made up of dots, each representing the learning sign *LS* of the individuals: red=negative and green=positive. Note that *LS* is negated in the lower sub-graph.

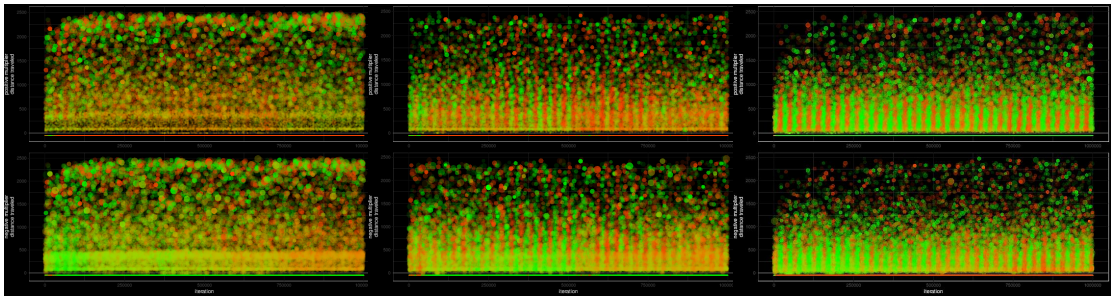
Overall the graph shows that the individual learning mechanism works. In the three static environments shown in the sub-figures (a - c), each multiplier settles on a distinct value. In sub-figure (b) the evolved value notation is reversed compared to (a) and (c),



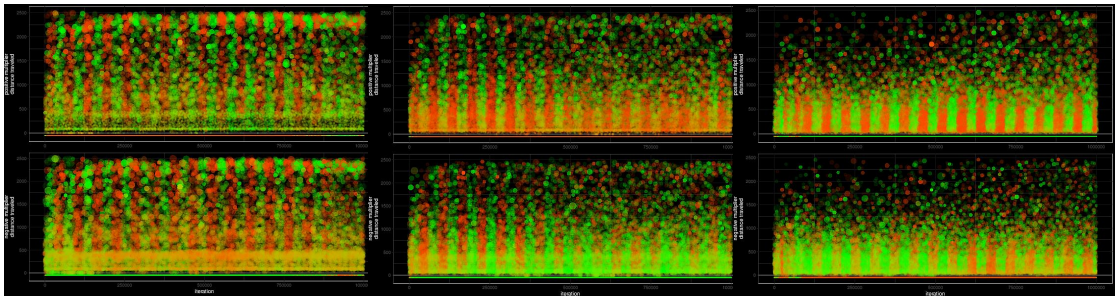
(a) static; c: 300, v: 1150 (b) static; c: 625, v: 625 (c) static; c: 1150, v: 425



(d) s: 5k; c: 300, v: 1150 (e) s: 5k; c: 625, v: 625 (f) s: 5k; c: 1150, v: 425



(g) s: 15k; c: 300, v: 1150 (h) s: 15k; c: 625, v: 625 (i) s: 15k; c: 1150, v: 425



(j) s: 30k; c: 300, v: 1150 (k) s: 30k; c: 625, v: 625 (l) s: 30k; c: 1150, v: 425

Fig. 6.2 Adaptation mechanism and performance values for Evo + IL in three different environments, defined by the quantity (**c**) and value (**v**) of energy tokens available in a *static* setting (a-c) at a seasonal change (**s**) of 5k (d - f), 15k (g - i) and 30k (j - l). These intention of these figures is to demonstrate the general trends and patterns and not the accurate representation of data. Hence, certain elements, such as legends and axis descriptions, have been omitted. The reader is referred to figure 6.3 for a more detailed version of sub figure (j) where the omitted details of the graphs are shown.

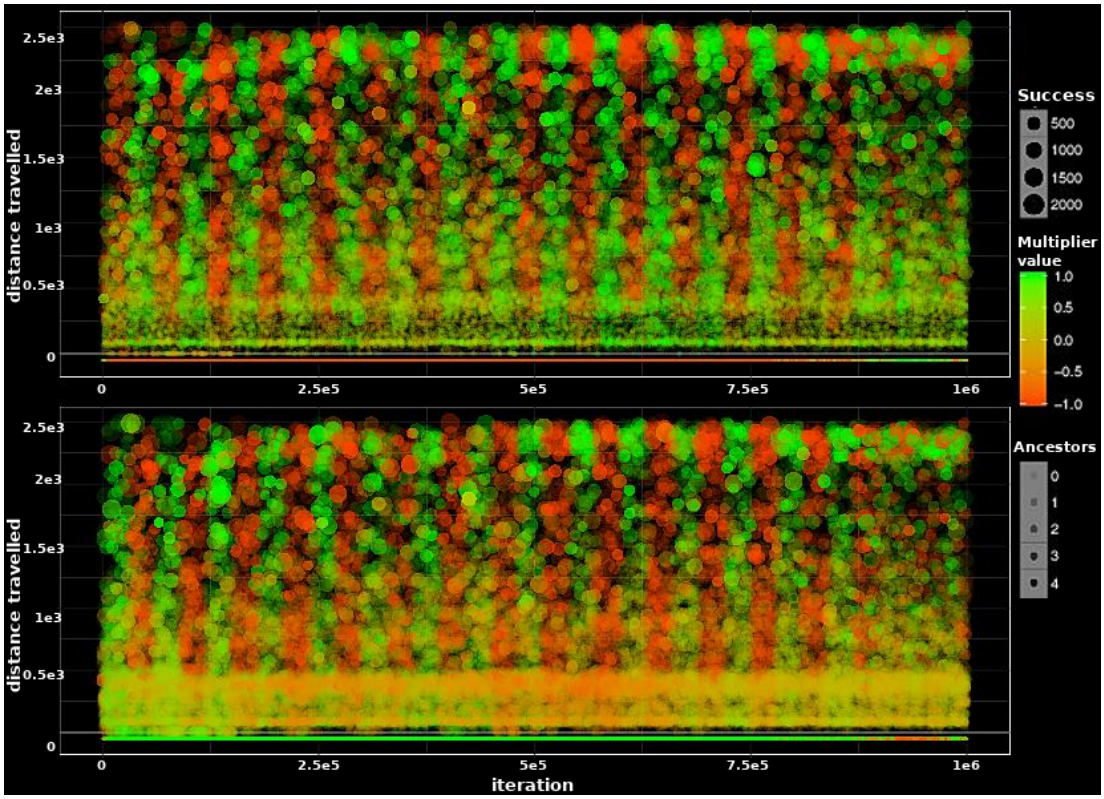


Fig. 6.3 Detailed sample figure for adaptation mechanism and performance values in figure 6.2. The figure shows a randomly selected run of the experiment for the algorithm configuration *Evo + IL* in a scarce environment with a seasonal change rate of 30k. As the resulting plot is quite dense, only 10% of the robots (sampled at random) are shown. The top and bottom half of the figure shows data from the multipliers of the first and second token types respectively. Each dot represents a multiplier value of an individual at the end of its lifetime. Distance travelled by the individual is shown on the y-axis, the colour indicates the value of the respective multiplier. The size represents the success in collecting and avoiding tokens of said types and the transparency represents the reproductive success of the individual using these multipliers. The change in colour from one extreme to another in value (-1: red; 1: green), indicates successful adaptation to the environmental change every 30k iterations when token types swap values (positive becomes negative and vice versa).

which demonstrates the need for the learning sign (*LS*) that co-evolves alongside the neural network weights to adjust the learning mechanism's direction of change.

Comparing the three environments (columns of sub-figures) against each other it can be seen that as the number of tokens increases, the distance robots travel decreases. With an increased number of positive tokens available, the distance an individual has to travel to gather them decreases. There is an equal amount of positive and negative

tokens available. Therefore, the chance to accidentally collect a negative token is also more likely and decreases the lifetime. Recall that the ray-sensors are spaced around the robot's body, which allows for the situation that the robot can drive over a token without detecting it first.

The same correlation between number of available tokens in the environment and distance travelled can be observed for the dynamic environments (d - l). Further, it can be observed that the adaptation mechanism in fact adapts to the change in environment. It allows the robot to adjust the multipliers to correspond with the changed token values after a seasonal change. The resulting colour-bands are most distinct in sub-figures (j - l), representing the longest seasonal change rate of 30k. They also appear in the sub-figures for 5k and 15k, but are less clear than in 30k.

6.6.3 General Observations

Figure 6.4 shows the normalised difference between the number of positive tokens (p) and the number of negative tokens (n) collected per season over time (i.e. p-n). It shows (scarce, balanced, abundant) environments for the two cases in which the values of the tokens change dynamically with the seasons. The solid lines on the graph represent this value combined over both seasons, while the dashed and dotted lines represent the value in season 0 and season 1 respectively. All lines are smoothed over the relevant points. In every experiment the oscillation of the plotted lines can be observed. The following observations offer an explanation for this behaviour.

Recall that tokens respawn after 500 iterations and that tokens waiting to respawn are also affected by seasonal change. Therefore, if a change happens during a token being consumed and being placed back in the arena, the token respawns as the opposite type. The frequency of respawning and the seasonal change interact; like two wave signals with different frequencies. While one oscillates quickly, the other changes slowly. The product of their interaction and the magnitude of the resulting wave will

oscillate as well. At times both waves will cancel each other out, and at others they will lead to a very high amplitude. In this particular case, the faster changing respawn time interacts with the much slower rate of seasonal change. There are times when there is an ever increasing availability of positive tokens until the number of available tokens peaks, followed by a decline. Furthermore, the used adaptation algorithm determines the ability of robots to distinguish between tokens and thus the amount of each type being consumed and consequently respawning.

In the baseline experiment, where robots can't distinguish between different types, there is, theoretically, equal likelihood of either type being picked up. For example, 3 positive and 1 negative is equally likely to 1 positive and 3 negative. However, a negative token takes energy away from a robot. Recall that robots start the evaluation of a new genome with 500 energy units and a negative token is always fixed at -400. Hence, 80% of its initial energy is taken as a result of the encounter with a negative token. Therefore, collecting more negative tokens than positive ones is much less likely than vice versa. Despite being unable to distinguish between token types, on average, this gives a bias towards collecting more positive than negative tokens.

This bias is true for all experiments. Although the adaptation algorithm enables the robot to learn/evolve to distinguish between token types, the results shown reflect the whole population, including ill adapted individuals. To enable a genome to be evaluated a robot begins every generation with a starting energy of 500 energy units. A necessary mechanism that has the unfortunate effect of diluting the selection pressure for the desired behaviour to be able to distinguish.

This given starting energy is free, meaning it is independent of an individual's fitness. It can be used for any kind of behaviour, including only picking up negative tokens. Genomes are selected using a fitness proportionate selection mechanism based on a comparison of the energy balance at the time of the genome exchange. A young genome that might have evolved to consume negative tokens with a token count of zero outranks another genome with the desired behaviour of preferring positive tokens that

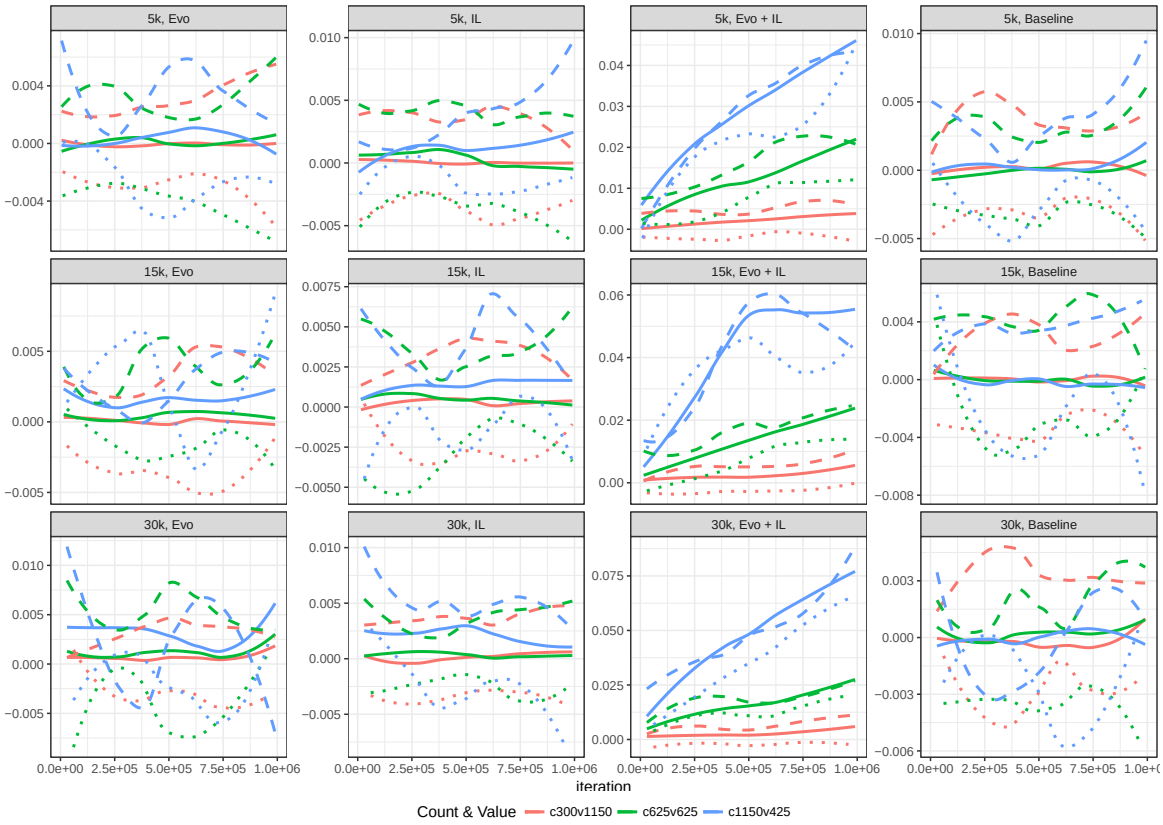


Fig. 6.4 Normalised difference between positive and negative tokens collected. Solid line is the value combined over all seasons, dashed = season 0, dotted = season 1.

has consumed a negative token by accident. Many similar scenarios are imaginable, which could lead to a counter productive strategy spreading through the population.

All these factors lead to varying oscillations that can be observed in figure 6.4.

6.6.4 Influence of the Adaptation Mechanism

Table 6.6 shows the median totalTokenRatio for each of the three adaptation mechanisms (EVO, EVO+IL, IL) in each of the three environments and for each value of seasonal change. The values are compared to the result from the baseline experiment for each case.

The EVO method (that evolves multiplier values but has no adaptation during a lifetime) outperforms the baseline method in all three static environments (season change = 0). Here, evolution is able to determine appropriate values for each multiplier type.

However, in the dynamic environment, evolving the multiplier values is detrimental. In the first season, evolution can find appropriate multiplier values (particularly in a long season). However, as soon as the season changes, these become irrelevant; if these values have spread sufficiently through the population it may take considerable time for evolution to reverse this change, while in the meantime, the robot will continue to collect negative tokens.

In this experimental setup, where a genome is supplied with an amount of energy to start with, it stands to reason that negative token have a higher impact on the adaptation mechanism and facilitate quicker learning.

The IL method (fixed learning rate and random initialisation of values) never outperforms the baseline method in the static environment, and is worse than the baseline in the dynamic environments. The magnitude of the effect is highest in the seasonal change = 5000 environment for a balanced environment. It appears that the learning rate is not sufficient to adapt a randomly initialised multiplier to a suitable value, while the randomness can actually bias the robot towards collecting a particular type. On average, this is worse than the baseline case in which the robot has equal preference for both types.

In contrast, with the exception of the two *dynamic and scarce* environments, the EVO+IL method that evolves the *LR*, *LS* and the multiplier values, and also adjusts the latter during lifetime, a significant improvement is observed with respect to the baseline method. In the *scarce* environments, the robots have little information available to them to inform learning as there are few tokens. When the environment is changing rapidly this is particularly detrimental. In the other environments, there are more tokens to learn from. When this is coupled with the ability to both evolve useful multiplier values *and* adapt them at an appropriate rate, the robots learn to adapt to the changing environments and improve their behaviour in the static environment.

Table 6.7 provides a pairwise comparison of learning mechanisms within different environments. 22/27 comparisons are significant.

In this table and subsequent ones, the symbols $=$, $<$,$>$ indicate whether the median values for totalTokenRatio are not significantly different, significantly smaller or larger respectively. p-values below the significance level of 0.05 are written in bold.

For the *scarce* environment, the general pattern is that EVO+IL outperforms the other two methods in 4/6 cases, with no statistical difference in the other two cases. In the balanced environment, EVO+IL also clearly dominates both EVO and IL. EVO dominates IL in the static and 5k experiments. Finally, in the *abundant* environment, the supremacy of EVO+IL is again observed, while IL dominates EVO in both of the dynamic environments.

Table 6.6 Showing median of end values by seasonal change and the experiment for *totalTokenRatio* over the final 5000 iterations (N:30).

↓, ↔, ↑ indicate whether the value is lower, not different or higher respectively compared to the baseline experiment. The number of arrows corresponds to the magnitude level of the effect size based on a Vargha and Delaney A test. (1 = small, 2 = medium, 3 = large)

Experiment		**Evo**			**IL**			**Evo + IL**		
	count	**300**	**625**	**1150**	**300**	**625**	**1150**	**300**	**625**	**1150**
Season	**value**	**1150**	**625**	**425**	**1150**	**625**	**425**	**1150**	**625**	**425**
0		↑↑↑ 0.5301	↑↑↑ 0.5411	↑↑↑ 0.5662	↔ 0.5034	↔ 0.5056	↓ 0.4997	↑↑↑ 0.5306	↑↑↑ 0.5388	↑↑↑ 0.5705
5k		↔ 0.4995	↓ 0.4982	↓ 0.4989	↓ 0.5006	↓↓ 0.4975	↔ 0.5023	↔ 0.5029	↑↑ 0.5134	↑↑↑ 0.5191
15k		↓↓ 0.496	↓↓ 0.495	↓ 0.4981	↓ 0.4973	↓ 0.4993	↓ 0.5011	↔ 0.4981	↑↑ 0.5136	↑↑↑ 0.5121

*

Table 6.7 Showing p-values of pairwise comparisons of the learning mechanism for *totalTokenRatio* (row vs. column) over the final 5000 iterations.

Environment		**count:300 value:1150**		**count:625 value:625**		**count:1150 value:425**	
Season	**Experiment**	**IL**	**Evo + IL**	**IL**	**Evo + IL**	**IL**	**Evo + IL**
0	**Evo**	> **6.04e-35**	= 6.64e-01	> **8.51e-77**	= 4.32e-01	> **4.44e-82**	< **1.5e-03**
	IL		< **3.6e-26**		< **6.66e-59**		< **6.7e-150**
5k	**Evo**	= 8.45e-01	< **3.18e-05**	> **4.42e-06**	< **3.36e-55**	< **4.22e-15**	< **9.94e-122**
	IL		< **1.27e-04**		< **3.6e-70**		< **2.9e-80**
15k	**Evo**	= 7.84e-02	< **8.55e-04**	< **7.4e-03**	< **1.09e-41**	< **1.28e-08**	< **3.11e-77**
	IL		= 5.27e-02		< **5.83e-42**		< **2.23e-72**

6.6.5 Influence of Environmental Parameters

Table 6.8 provides a pairwise comparison of environments for *totalTokenRatio* obtained at the end of each experiment.

The data clearly indicates that for the methods that include an evolutionary component with the learning algorithm in the static environment: *abundant* > *balanced* > *scarce*. In contrast, when only a fixed individual learning mechanism is used with no adaptation of learning rate, then the reverse appears true: the token ratio is higher in the *balanced* and *scarce* environments than in the *abundant* environment, with no significant difference between *balanced* and *abundant*.

In the slow changing environment (15k), the general trend is that *abundant* > *balanced* > *scarce* for *all* three mechanisms. In the rapidly changing environment, a mixed picture emerges. For the EVO+IL mechanism, it is clear that *abundant* > *balanced* > *scarce*. For EVO, the *scarce* environment does *not* provide significantly different results to the other two, whereas for *IL*, both *scarce* and *balanced* prove harder than *abundant*, but *scarce* outperforms *balanced*.

6.6.6 Influence of Environmental Change

Next, in table 6.9, it is considered how the rate of change of a given environment influences the interaction between environmental parameters and learning mechanisms. In 21/27 pairwise comparisons, statistically significant results are observed.

In the *scarce* environments, there is a general pattern in terms of rate of change: $static > 5k > 15k$ for *all* mechanisms. In the *balanced* environments, the same general pattern is observed, with the exception that for the IL and EVO+IL mechanisms, no statistical differences are noted between the 5k and 15k environments. In the *abundant* environments, the same general pattern as above can be noted, except for IL, where the only significant result shows that 5k>15k, while in contrast, for EVO, 5k<15k.

Table 6.8 Showing p-values of pairwise comparisons of environments for *totalTokenRatio* (row vs. column) over the final 5000 iterations.

	Experiment		**Evo**		**IL**		**Evo + IL**	
	count		**625**	**1150**	**625**	**1150**	**625**	**1150**
Season		**value**	**625**	**425**	**625**	**425**	**625**	**425**
0	**300**	**1150**	$<$ **1.24e-07**	$<$ **3.02e-27**	= 5.78e-01	$>$ **7.33e-05**	$<$ **3.27e-02**	$<$ **1.44e-40**
	625	**625**		$<$ **9.37e-16**		$>$ **7.87e-11**		$<$ **1.33e-32**
5k	**300**	**1150**	= 4.55e-01	= 3.08e-01	$>$ **7.89e-05**	$<$ **1.99e-02**	$<$ **3.87e-19**	$<$ **1.5e-32**
	625	**625**		$>$ **1.22e-03**		$<$ **1.81e-17**		$<$ **5.88e-04**
15k	**300**	**1150**	$<$ **4.78e-02**	$<$ **1.58e-04**	= 6.51e-01	$<$ **4.11e-04**	$<$ **1.94e-16**	$<$ **9.56e-30**
	625	**625**		$<$ **1.09e-08**		$<$ **1.44e-12**		= 2.61e-01

*

Table 6.9 Showing p-values of pairwise comparisons of seasonal change for *totalTokenRatio* (row vs. column) over the final 5000 iterations.

	Environment	**count:300 value:1150**		**count:625 value:625**		**count:1150 value:425**	
Experiment	**Season**	**5k**	**15k**	**5k**	**15k**	**5k**	**15k**
Evo	**0**	$>$ **9.09e-69**	$>$ **6.76e-55**	$>$ **1.2e-131**	$>$ **4.34e-99**	$>$ **6.91e-120**	$>$ **9.94e-88**
	5k		$>$ **1.89e-02**		$>$ **1.67e-05**		$<$ **6.04e-03**
IL	**0**	$>$ **2.54e-05**	$>$ **6.93e-09**	$>$ **4.05e-27**	$>$ **4.43e-20**	= 1.03e-01	= 7.08e-01
	5k		= 7.61e-02		= 8.51e-02		$>$ **8.33e-03**
Evo + IL	**0**	$>$ **2.82e-35**	$>$ **1.37e-31**	$>$ **1.76e-32**	$>$ **4.15e-32**	$>$ **3.41e-178**	$>$ **3.54e-118**
	5k		$>$ **1.7e-02**		= 4.38e-01		= 7.19e-01

6.6.7 Analysis

The previous section shows that the EVO+IL clearly outperforms IL and EVO in all parametrisations of the environment and for all rates of change. Its behaviour can be examined more closely by plotting the normalised difference between the number of positive tokens (p) and the number of negative tokens (n) collected per season over time (i.e. p-n). This is shown in figure 6.4 for the (scarce, balanced, abundant) environments for the two cases in which the values of the tokens change dynamically with the seasons. The solid lines on the graph represent this value combined over both seasons, while the dashed and dotted lines represent the value in season 0 and season 1 respectively. All lines are smoothed over the relevant points. The continuous improvement in this metric is clearly identified for EVO+IL, showing a generally robust response to the changes in token value (i.e. an upward trend). The *abundant* environment proves most straightforward to learn in: having a large *quantity* of information of low-value outweighs the situation in which a small quantity of high-value information is available. In contrast, in the baseline experiment in which *no* information is available as to token value, the (p-n) metric continuously cycles. In this case, the best that evolution can do is learn a token-avoidance behaviour, as there is no means of distinguishing between tokens.

6.7 Summary and Conclusion

In this chapter the performance of a number of adaptation mechanisms that augment evolution of a neural network controller were investigated. Adaptation mechanisms that included heritable and fixed components were analysed in three different environments in which both the number of learning opportunities and the impact of the learning opportunities varied.

It is shown that an adaptation mechanism in which all components evolve and are heritable (EVO+IL) copes well in static and dynamic environments, and is able to learn

to distinguish between tokens of different values. In dynamic environments, the greatest effect is observed when the environment contains a large number of small learning opportunities. The fewer the learning opportunities, the less effective the mechanism becomes, despite the fact that the opportunities provide more energy and therefore more information to the learning mechanism.

In contrast, the EVO and IL mechanisms both prove to be detrimental in a changing environment when compared to the baseline scenario. However, no clear pattern emerges in terms of the magnitude of the effect with respect to the number of learning opportunities present. The IL method never outperforms the baseline experiments, whereas EVO is beneficial only in a static environment. In the latter case, performance is greatest in the environment with most tokens, and decreases as the number of tokens decreases.

The results clearly demonstrate the interaction between the learning mechanism and environmental parameters. This is of particular relevance for distributed algorithms, such as mEDEA, in which environmental pressure influences reproductive abilities. The huge varieties of behaviours that were displayed in different environments highlight how fundamental it is to not just select parameters at random, but to perform a more thorough analysis. The emerging behaviours using a single set of algorithmic parameters varied from giving a huge advantage, to showing no difference, to even being counter productive.

Chapter 7

Conclusion

7.1 Summary

This thesis investigates several aspects of environment-driven adaptation in simulated evolutionary swarm robotics and mEDEA in particular. Firstly, mEDEA is augmented with an explicit fitness measure. It is shown that both methods lead to an improvement over the original algorithm in the swarm's ability to maintain energy over longer periods. Secondly, an energy model is derived from the field of wireless sensor networks research that accounts for communication costs in the robot simulation. Under these new conditions, biasing the broadcasting of genomes has a negative effect on survivability. Thirdly, the environment's influence on the emergence of behaviours at the individual and swarm level is investigated using a 3-dimensional map. This map is used to identify interesting regions for future experimentation, in contrast to ad-hoc methods currently used for parameter settings. These regions provide challenging environments for the swarm while still allowing algorithm specific characteristics to show effect and be explored. Finally, an individual learning mechanism is developed that does not require any *a priori* knowledge of the environment and can help the evolutionary process to adapt. It is investigated how environmental parameters (number of items in the environment, their respective values and the rate of change of said values) influence

the effectiveness of adaptation mechanisms and how the nature of the adaptation mechanisms themselves influences performance in different environments. The results show that there is a clear link between environmental conditions, specifically the rate of change and the availability of learning opportunities, and the effectiveness of different adaptation mechanisms. Further, the incentive to learn in the form of reward or punishment, in other words, the environmental pressure that drives the evolutionary process must be high enough to make any form of adaptation worthwhile.

7.2 Answers to Research Questions

RQ 1: To what extent can adding an explicit relative fitness to mEDEA (creating mEDEA$_{rf}$) increase the maintained energy across the swarm in comparison to the existing algorithm without explicit fitness?

A number of extensions have been introduced to an existing Environment-driven Distributed Evolutionary Adaptation algorithm — mEDEA. The goal of this work is to show that the integrity of the swarm can be maintained in a more robust manner than in the original work, while still retaining the original distributed and online flavour of the algorithm by using a fitness function that indicates fitness to survive. Having introduced the new fitness function, two new methods were described that adapted either the broadcast range or the probability of broadcasting of a robot, based on its estimate of its own relative fitness.

This biases the spread of genomes through the population. Robots that are relatively fitter than their neighbours are able to spread their genomes more: individual robots perform a random selection from their local store of (now biased) genomes. A thorough analysis of the experimental results shows that a considerable gain in performance is achieved, both in the number of active robots

at the end of a fixed period of evaluation, and in the energy levels sustained by those robots.

The new fitness function was also evaluated within an explicit selection method. Experiments showed that this also provided significant improvements over mEDEA, and slightly outperformed the biased broadcast methods in terms of energy sustained.

RQ 2: How does accounting for the cost of communication influence the performance of an evolutionary algorithm that relies on communication?

A new energy model based on the Free-Space model was introduced to account for the cost of communication. Experimental results showed this exerts environmental pressure to implicitly select for genomes that maintain high energy levels. Comparing the method of varying the broadcasting based on fitness to mEDEA alone and with explicit roulette-wheel genome selection shows that although marginal, the latter approaches outperform the mEDEA$_{rf}$ methods as they partially reverse the effect of the environmental pressure.

The difference in active robots is largest for the experiment where the fitness measure is used for both mechanisms, varying broadcasting and the explicit selection. This effect is smaller for the two experiments where no or just the explicit selection mechanism is active. The additional environmental pressure of the added cost of communication leads to more efficient behaviour that avoids costly communication.

Although fewer robots are active at the end of the generation, it is argued that the method of biasing what to broadcast has a benefit as it clearly shows that robots spend less time collecting tokens. A crucial advantage if a task is added in the future.

The emergent behaviours of robots in the swarm has changed with the introduction of an energy-model that reflects more accurately the incurred energy costs in reality. It can be seen that the performance of an algorithm depends not just on its own parameters, but the external conditions in which the swarm is deployed.

RQ 3: To what extent does the specific parametrisation of the environment influence the emergence of behaviour in mEDEA$_{rf}$?

The specific environmental conditions affect the selection pressure exerted onto the evolutionary process. Adjusting the availability and value of energy tokens in the environment results in the evolution of a range of different behaviours. Three distinct regions are observed when analysing the performance map that results from running an environment-driven evolutionary algorithm (mEDEA$_{rf}$) in an environment that is parameterised by two aforementioned values that control the distribution of energy in the environment. In these three distinct regions the final energy balance can be negative, neutral or positive. A fourth region is found in which robots cannot survive. The energy neutral region appears to be a good region in which to undertake experiments. It provides an environment in which robots are able to survive, enabling experimentation, while at the same time rewarding new behaviours that are able to more efficiently harness energy from the environment. It is clear that the environment plays a key role in influencing what kind of behaviours emerge, in that it is not the total amount of energy available that matters, but also the manner in which it is distributed.

Introducing a change into the experimental setting showed a shift in the identified regions. From this it is evident that such analysis is beneficial to identify regions of interest.

RQ 4: To what extent does the environment influence the most appropriate combination of evolutionary and lifetime-adaptation mechanisms in mEDEA$_{rf}$?

Adaptation mechanisms that included heritable and fixed components were analysed in three different environments in which both the number of learning opportunities and the impact of the learning opportunities varied.

It is shown that an adaptation mechanism in which all components evolve and are heritable and are adapted during lifetime (EVO+IL) copes well in static and dynamic environments, and is able to learn to distinguish between tokens of different values. In dynamic environments, the greatest effect is observed when the environment contains a large number of small learning opportunities. The fewer the learning opportunities, the less effective the mechanism becomes, despite the fact that the opportunities provide greater rewards or punishments and therefore more information to the learning mechanism.

In contrast, the EVO and IL mechanisms both prove to be detrimental in a changing environment when compared to the baseline scenario. However, no clear pattern emerges in terms of the magnitude of the effect with respect to the number of learning opportunities present. The IL method never outperforms the baseline experiments, whereas EVO is beneficial only in a static environment. In the latter case, performance is greatest in the environment with most tokens, and decreases as the number of tokens decreases.

The results clearly demonstrate the interaction between the learning mechanism and environmental parameters. This is of particular relevance for distributed algorithms such as mEDEA$_{rf}$, in which environmental pressure influences reproductive abilities. The huge variety of behaviours that were displayed in different environments highlight how fundamental it is to not just select parameters at random, but to perform a more thorough analysis. The emerging behaviours using a single set of algorithmic parameters varied from giving a massive advantage, to showing no difference, to even being counter productive.

7.3 Discussion

A reasonable body of research exists around the mEDEA algorithm [19], which is a distributed evolutionary algorithm for swarm robotics. It has been studied in its original form, with no explicit fitness function [82, 20, 16], and in extended forms with explicit fitness functions [95, 40, 14]. In future applications in remote areas, e.g. the bottom of the ocean or in outer space, local adaptation to unforeseen changes or *a priori* unknown environments is a crucial aspect. In this thesis the mEDEA algorithm was extended with a relative fitness function that is based on a robot's ability to maintain energy in comparison to its current neighbours. Using only local rather than global information and making relative comparisons maintains the distributed nature of the mEDEA algorithm and allows for niching to occur within the population. This is particularly useful under varying conditions across the environment where different traits are beneficial.

The field of evolutionary robotics benefits from recent technological advances, which allows more and more experiments to be performed in hardware [92]. Despite that, many experiments are being carried out in simulation, which can offer many advantages. Experimental settings can be changed with a push of a button, the complexity of the robots is not limited to the available hardware, and huge speed ups can be achieved by running multiple instances in parallel. However, solutions that work in one universe can often not be transferred to the other, a problem referred to as the reality gap [83]. It is caused by differences between the simulated world and reality. One aspect that is often overlooked is the cost of communication, which takes the form of energy consumption in real robots. In common robot simulators where the physics model focuses on the accuracy of movement this is mostly ignored. For applications where communication plays only a minor role, the resulting effect could be negligible. However, in mEDEA$_{rf}$ the exchange of information is a fundamental part of the algorithm. In addition, the small robots have only a limited energy supply and the optimisation criteria is based on

the robots' energy levels. Thus, it is crucial to account for these costs as experiments have shown.

Despite this increased interest in evolutionary robotics, experimental practices are not as rigorous as in other areas of evolutionary computation. For example, routing algorithms or bin-packing heuristics are commonly compared on equal grounds through the use of benchmark sets [36]. In comparison, evolutionary robotics experiments are often performed using a single environment that is chosen ad-hoc, e.g. to suit the experimental setup or maybe even purely random [87]. What follows is an arbitrary presentation of results with respect to environmental parameters, which makes it hard or impossible to compare algorithms without recreating whole sets of experiments in the same environmental setting. This process requires a lot of extra effort and can lead to results that are only representative of a specific area of the problem space, e.g. if the environment is deliberately chosen in a way that demonstrates the superiority of one algorithm over the other.

The method of creating performance maps, which is introduced in this thesis, helps standardise the way results are presented. In addition it can help direct further experimentation and give a better overview of performance in different environments, thus allowing statements about stability and reliability of the algorithm. This has been demonstrated in this thesis on two different occasions where interesting regions for further experimentation have been identified. Although the process of creating performance maps is time consuming and requires effort, it helps represent results in a more consistent way.

7.4 Future Work

While conducting the research presented here, some more or less obvious extensions have presented themselves. Some were natural extensions that stemmed from the limitations of experimental settings, others from illuminating discussions with peers.

The relative fitness function introduced into mEDEA to form mEDEA$_{rf}$ was demonstrated to outperform vanilla mEDEA in a set of experiments (chapter 3). The algorithm could be implemented in a real swarm of robots to investigate its performance under reality constraints.

The evolutionary process in the mEDEA$_{rf}$ algorithm relies heavily on communication. Communication in real robots is not free and comes at a cost in the form of energy usage. However, this is rarely accounted for in simulations. One of the contributions in this thesis was to introduce a communication energy model into the simulator to account for this cost (chapter 4). Hence, a logical step would be to verify the results gathered from simulation with real world experiments.

The method introduced in chapter 5 creates a performance map to link specific environmental parameters to the swarm's performance. Future work could be aimed at understanding the performance map in more detail and, in particular, explaining the ruggedness of some regions. Further, behaviour in different regions of the map could be analysed in greater detail by tracing individual robots and examining their behavioural pattern. It is possible to direct the focus of the research towards the field of ALife and investigate the environmental influence on theories like the optimal foraging theory.

The research into the environmental influence on the usefulness of different adaptation mechanisms currently focuses on a single lifetime adaptation algorithm. Future work could extend the analysis to other mechanisms for adding individual learning and/or adaptation, as well as considering social learning as recently demonstrated by [59, 60] to be effective in some scenarios. Further, it can be analysed whether algorithmic features can be mapped to resulting emergent behaviours. If successful, this could pave the way to using ensemble methods, where the most appropriate mix of adaptation mechanisms is selected from a wider pool based on the current environmental settings.

References

[1] Auerbach, J. E. and Bongard, J. C. (2012). On the relationship between environmental and morphological complexity in evolved robots. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference - GECCO '12*, pages 521–528, New York, New York, USA. ACM Press.

[2] Baldwin, J. M. (1896). A New Factor in Evolution. *The American Naturalist*, 30(354):441–451.

[3] Barca, J. C. and Sekercioglu, Y. A. (2013). Swarm robotics reviewed. *Robotica*, 31(03):345–359.

[4] Bayindir, L. (2016). A review of swarm robotics tasks. *Neurocomputing*, 172:292–321.

[5] Beni, G. (2005). From Swarm Intelligence to Swarm Robotics. In *International Workshop on Swarm Robotics, SR 2004*, pages 1–9. Springer, Berlin, Heidelberg.

[6] Bernard, A., André, J.-B., and Bredeche, N. (2016a). Evolving Specialisation in a Population of Heterogeneous Robots : the Challenge of Bootstrapping and Maintaining Genotypic Polymorphism. *Artificial Life 15: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, pages 1–8.

[7] Bernard, A., André, J.-B., and Bredeche, N. (2016b). To Cooperate or Not to Cooperate: Why Behavioural Mechanisms Matter. *PLOS Computational Biology*, 12(5):e1004886.

[8] Beyer, H.-G. and Schwefel, H.-P. (2001). Evolution strategies – A comprehensive introduction. *Natural Computing*, 1:3–52.

[9] Bianco, R. and Nolfi, S. (2004). Toward open-ended evolutionary robotics: evolving elementary robotic units able to self-assemble and self-reproduce. *Connection Science*, 16(4):227–248.

[10] Bjerknes, J. D. and Winfield, A. F. T. (2013). On Fault Tolerance and Scalability of Swarm Robotic Systems. pages 431–444. Springer, Berlin, Heidelberg.

[11] Blom, M. and Ekström, M. (2008). Extraction of an Energy model for Bluetooth 2.0 depending on transmission distance. Technical Report 1.

[12] Bongard, J. C. (2011). The 'What', 'How' and the 'Why' of Evolutionary Robotics. In Doncieux, S., Bredeche, N., and Mouret, J.-B., editors, *New Horizons in Evolutionary Robotics*, chapter 2, pages 29–35. Springer Berlin Heidelberg.

[13] Bongard, J. C. (2013). Evolutionary robotics. *Communications of the ACM*, 56(8):74.

[14] Boumaza, A. (2017). Phylogeny of Embodied Evolutionary Robotics. In *Proceedings of The Genetic and Evolutionary Computation Conference Companion 2017*. ACM.

[15] Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41.

[16] Bredeche, N. (2014). Embodied Evolutionary Robotics with Large Number of Robots. In *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, number 1, pages 272–273. The MIT Press.

[17] Bredeche, N., Haasdijk, E., and Eiben, A. E. (2009). On-line, on-board evolution of robot controllers. In *Proceedings of the 9th international conference on Artificial evolution*, pages 110–121, Strasbourg. Springer-Verlag.

[18] Bredeche, N., Haasdijk, E., and Prieto, A. (2018). Embodied Evolution in Collective Robotics: A Review. *Frontiers in Robotics and AI*, 5.

[19] Bredeche, N. and Montanier, J.-M. (2010). Environment-driven embodied evolution in a population of autonomous agents. In Schaefer, R., Cotta, C., Kołodziej, J., and Rudolph, G., editors, *Parallel Problem Solving from Nature, PPSN XI*, volume 6239, pages 290–299, Krakov, Poland. Springer Berlin Heidelberg.

[20] Bredeche, N. and Montanier, J.-M. (2012). Environment-driven Open-ended Evolution with a Population of Autonomous Robots. In *Evolving Physical Systems Workshop*, East Lansing, United States.

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

[22] Bredeche, N., Montanier, J.-M., Weel, B., and Haasdijk, E. (2013). Roborobo! a Fast Robot Simulator for Swarm and Collective Robotics. *CoRR*, abs/1304.2.

[23] Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23.

[24] Capodieci, N., Hart, E., and Cabri, G. (2016). Artificial Immunology for Collective Adaptive Systems Design and Implementation. *ACM Transactions on Autonomous and Adaptive Systems*, 11(2):1–25.

[25] Cheney, N., Bongard, J. C., and Lipson, H. (2015). Evolving Soft Robots in Tight Spaces. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15*, pages 935–942, New York, USA. ACM Press.

[26] Cianci, C. M., Raemy, X., Pugh, J., and Martinoli, A. (2007). Communication in a Swarm of Miniature Robots: The e-Puck as an Educational Tool for Swarm Robotics. In *Swarm Robotics*, pages 103–115.

[27] Cliff, D., Husbands, P., and Harvey, I. (1993). Explorations in Evolutionary Robotics. *Adaptive Behavior*, 2(1):73–110.

[28] Cussat-Blanc, S., Harrington, K., and Pollack, J. B. (2015). Gene Regulatory Network Evolution Through Augmenting Topologies. *IEEE Transactions on Evolutionary Computation*, 19(6):823–837.

[29] Dawkins, R. (1976). *The Selfish Gene*.

[30] Doncieux, S., Bredeche, N., Mouret, J.-B., and Eiben, A. E. (2015). Evolutionary robotics: what, why, and where to. *Frontiers in Robotics and AI*, 2(March):4.

[31] Doncieux, S. and Mouret, J.-B. (2014). Beyond black-box optimization: a review of selective pressures for evolutionary robotics. *Evolutionary Intelligence*, pages 1–23.

[32] Dorigo, M., Floreano, D., Gambardella, L. M., Mondada, F., Nolfi, S., Baaboura, T., Birattari, M., Bonani, M., Brambilla, M., Brutschy, A., Burnier, D., Campo, A., Christensen, A. L., Decugniere, A., Di Caro, G., Ducatelle, F., Ferrante, E., Forster, A., Gonzales, J. M., Guzzi, J., Longchamp, V., Magnenat, S., Mathews, N., Montes de Oca, M., O'Grady, R., Pinciroli, C., Pini, G., Retornaz, P., Roberts, J., Sperati, V., Stirling, T., Stranieri, A., Stutzle, T., Trianni, V., Tuci, E., Turgut, A. E., and Vaussard, F. (2013). Swarmanoid: A Novel Concept for the Study of Heterogeneous Robotic Swarms. *IEEE Robotics & Automation Magazine*, 20(4):60–71.

[33] Duarte, M., Costa, V., Gomes, J., Rodrigues, T., Silva, F., Oliveira, S. M., and Christensen, A. L. (2016). Evolution of Collective Behaviors for a Real Swarm of Aquatic Surface Robots. *PLOS ONE*, 11(3):e0151834.

[34] Eiben, A. E. (2015). EvoSphere: The World of Robot Evolution. In Dediu, A.-H., Magdalena, L., and Martín-Vide, C., editors, *Fourth International Conference, TPNC 2015*, pages 3–19. Springer, Berlin, Heidelberg.

[35] Eiben, A. E., Haasdijk, E., and Bredeche, N. (2010). Embodied, On-Line, On-Board Evolution for Autonomous Robotics. In Levi, P. and Kernbach, S., editors, *Symbiotic Multi-Robot Organisms*, chapter 5.2, pages 362–384. Springer, Heidelberg.

[36] Eiben, A. E. and Jelasity, M. (2002). A critical note on experimental research methodology in EC. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 1, pages 582–587. Ieee.

[37] Eiben, A. E. and Smith, J. E. (2003). What is an Evolutionary Algorithm? In *Introduction to Evolutionary Computing*, Natural Computing Series, chapter 2, pages 15–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 1st edition.

[38] Ellefsen, K. O. (2014). The Evolution of Learning Under Environmental Variability. In *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, pages 649–656. The MIT Press.

[39] Erbas, M. D., Winfield, A. F. T., and Bull, L. (2013). Embodied imitation-enhanced reinforcement learning in multi-agent systems. *Adaptive Behavior*, pages 1–20.

[40] Fernández Pérez, I., Boumaza, A., and Charpillet, F. (2014). Comparison of Selection Methods in On-Line Distributed Evolutionary Robotics. In *ALife 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, pages 282–289. The MIT Press.

[41] Floreano, D., Dürr, P., and Mattiussi, C. (2008a). Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62.

[42] Floreano, D., Husbands, P., and Nolfi, S. (2008b). Evolutionary Robotics. In Siciliano, B. and Khatib, O., editors, *Handbook of Robotics*, pages 1423–1451. Springer Berlin Heidelberg, Berlin, Heidelberg.

[43] Floreano, D. and Mondada, F. (1996). Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, 26(3):396–407.

[44] Floreano, D., Zufferey, J.-C., and Nicoud, J.-D. (2005). From Wheels to Wings with Evolutionary Spiking Circuits. *Artificial Life*, 11(1-2):121–138.

[45] Gerkey, B., Vaughan, R. T., and Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR 2003)*, number Icar 2003, pages 317–323, Coimbra, Portugal.

[46] Gomes, J., Duarte, M., Mariano, P., and Christensen, A. L. (2016). Cooperative Coevolution of Control for a Real Multirobot System. In Handl, J., Hart, E., Lewis, P. R., López-Ibáñez, M., Ochoa, G., and Paechter, B., editors, *Parallel Problem Solving from Nature – PPSN XIV: 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings*, pages 591–601. Springer International Publishing, Cham.

[47] Gomes, J., Urbano, P., and Christensen, A. L. (2013). Evolution of swarm robotics systems with novelty search. *Swarm Intelligence*, 7(2-3):115–144.

[48] Gross, R. and Dorigo, M. (2008). Self-Assembly at the Macroscopic Scale. *Proceedings of the IEEE*, 96(9):1490–1508.

[49] Haasdijk, E. (2015). Combining Conflicting Environmental and Task Requirements in Evolutionary Robotics. In *2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems*, pages 131–137. IEEE.

[50] Haasdijk, E., Eiben, A. E., and Winfield, A. F. T. (2013a). Individual, Social and Evolutionary Adaptation in Collective Systems. In Kernbach, S., editor, *Handbook of Collective Robotics - Fundamentals and Challenges*, chapter 12, pages 411–469. Pan Stanford, Germany, 2013 edition.

[51] Haasdijk, E. and Heinerman, J. (2017). Quantifying Selection Pressure. *Evolutionary Computation*, page EVCO_a_00207.

[52] Haasdijk, E., Smit, S. K., and Eiben, A. E. (2012). Exploratory analysis of an on-line evolutionary algorithm in simulated robots. *Evolutionary Intelligence*, 5(4):213–230.

[53] Haasdijk, E., Vogt, P. A., and Eiben, A. E. (2008). Social learning in Population-based Adaptive Systems. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1386–1392. IEEE.

[54] Haasdijk, E., Weel, B., and Eiben, A. E. (2013b). Right on the MONEE. In Blum, C., editor, *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, pages 207–214, Amsterdam, The Netherlands. ACM New York, NY, USA.

[55] Hart, E., Steyven, A., and Paechter, B. (2015). Improving Survivability in Environment-driven Distributed Evolutionary Algorithms through Explicit Relative Fitness and Fitness Proportionate Communication. In Silva, S., editor, *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15*, pages 169–176, New York, New York, USA. ACM Press.

[56] Hauert, S., Zufferey, J.-C., and Floreano, D. (2009). Evolved swarming without positioning information: an application in aerial communication relay. *Autonomous Robots*, 26(1):21–32.

[57] Heidemann, J., Bulusu, N., Elson, J., Intanagonwiwat, C., Lan, K.-c., Xu, Y., Ye, W., Estrin, D., and Govindan, R. (2001). Effects of Detail in Wireless Network Simulation. In *Proceedings of the SCS Multiconference on Distributed Simulation*, pages 3–11, Phoenix, Arizona, USA. Society for Computer Simulation.

[58] Heinerman, J., Drupsteen, D., and Eiben, A. E. (2015a). Three-fold Adaptivity in Groups of Robots: The Effect of Social Learning. In *Proceedings of the 17th annual conference on Genetic and evolutionary computation*, pages 177–183, New York, New York, USA. ACM Press.

[59] Heinerman, J., Rango, M., and Eiben, A. E. (2015b). Evolution, Individual Learning, and Social Learning in a Swarm of Real Robots. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 1055–1062. IEEE.

[60] Heinerman, J., Zonta, A., Haasdijk, E., and Eiben, A. E. (2016). On-line Evolution of Foraging Behaviour in a Population of Real Robots. pages 198–212. Springer, Cham.

[61] Heinzelman, W. R., Chandrakasan, A., and Balakrishnan, H. (2000). Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000*, pages 1–10.

[62] Hinton, G. E. and Nowlan, S. J. (1987). How learning can guide evolution. *Complex systems*, 1(1):495–502.

[63] Hornby, G. S. and Pollack, J. B. (2002). Creating High-Level Components with a Generative Representation for Body-Brain Evolution. *Artificial Life*, 8(3):223–246.

[64] Horneber, J. and Hergenroder, A. (2014). A Survey on Testbeds and Experimentation Environments for Wireless Sensor Networks. *IEEE Communications Surveys & Tutorials*, 16(4):1820–1838.

[65] Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.

[66] Hoverd, T. and Stepney, S. (2011). Energy as a driver of diversity in open-ended evolution. In Lenaerts, T., Giacobini, M., Bersini, H., Bourgine, P., Dorigo, M., and Doursat, R., editors, *Ecal 2011*, pages 356–363. MIT Press.

[67] Karafotias, G., Hoogendoorn, M., and Eiben, A. E. (2015). Parameter Control in Evolutionary Algorithms: Trends and Challenges. *IEEE Transactions on Evolutionary Computation*, 19(2):167–187.

[68] Kodjabachian, J. and Meyer, J.-A. (1998). Evolution and development of neural controllers for locomotion, gradient-following, and obstacle-avoidance in artificial insects. *IEEE Transactions on Neural Networks*, 9(5):796–812.

[69] Kowaliw, T., Bredeche, N., and Doursat, R. (2014). *Growing Adaptive Machines*. Springer Berlin Heidelberg.

[70] Lehman, J. and Stanley, K. O. (2008). Exploiting Open-Endedness to Solve Problems Through the Search for Novelty. In *Artificial Life XI*, pages 329–336. MIT Press.

[71] Li, W. and Shen, W. (2011). Swarm behavior control of mobile multi-robots with wireless sensor networks. *Journal of Network and Computer Applications*, 34(4):1398–1407.

[72] Lipson, H. (2005). Evolutionary Design and Evolutionary Robotics. In Bar-Cohen, Y., editor, *Biomimetics*, chapter 4, pages 129 – 155. CRC Press.

[73] Littman, M. L. (1995). Simulations combining evolution and learning. In *In Adaptive Individuals in Evolving Populations: Models and Algorithms: Santa Fe Institute Studies in the Sciences of Complexity*, pages 465–477. Addison-Wesley.

[74] Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., and Balan, G. (2005). MASON: A Multiagent Simulation Environment. *SIMULATION*, 81(7):517–527.

[75] Macedo, J., Marques, L., and Costa, E. (2017). Robotic odour search: Evolving a robot's brain with Genetic Programming. In Marques, L. and Bernardino, A., editors, *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 91–97, Coimbra, Portugal. IEEE.

[76] Mavrovouniotis, M., Li, C., and Yang, S. (2017). A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation*, 33:1–17.

[77] Millard, A. G., Timmis, J., and Winfield, A. F. T. (2014). Run-time detection of faults in autonomous mobile robots based on the comparison of simulated and real robot behaviour. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3720–3725, Chicago, IL, USA. IEEE.

[78] Mitri, S., Wischmann, S., Floreano, D., and Keller, L. (2012). Using robots to understand social behaviour. *Biological reviews of the Cambridge Philosophical Society*, 88(1):31–9.

[79] Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C. M., Klaptocz, A., Zufferey, J.-C., Floreano, D., and Martinoli, A. (2009). The e-puck, a Robot Designed for Education in Engineering. In *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, pages 59–65, Portugal.

[80] Montanier, J.-M. (2013). *Environment-driven Distributed Evolutionary Adaptation for Collective Robotic Systems*. PhD thesis, Université Paris Sud - Paris XI.

[81] Montanier, J.-M. and Bredeche, N. (2011a). Embedded Evolutionary Robotics: The (1+1)-Restart-Online Adaptation Algorithm. In Doncieux, S., Bredèche, N., and Mouret, J.-B., editors, *New Horizons in Evolutionary Robotics*, volume 341 of *Studies in Computational Intelligence*, chapter 11, pages 155–169. Springer Berlin Heidelberg, Berlin, Heidelberg.

[82] Montanier, J.-M. and Bredeche, N. (2011b). Surviving the Tragedy of Commons: Emergence of Altruism in a Population of Evolving Autonomous Agents. In *European Conference on Artificial Life*, pages 550–557. MIT Press.

[83] Mouret, J.-B. and Chatzilygeroudis, K. (2017). 20 Years of Reality Gap: a few Thoughts about Simulators in Evolutionary Robotics. In *Proceedings of The Genetic and Evolutionary Computation Conference Companion 2017*, Berlin. ACM.

[84] Mouret, J.-B. and Clune, J. (2015). Illuminating search spaces by mapping elites. *CoRR*, abs/1504.0.

[85] Mouret, J.-B. and Tonelli, P. (2014). Artificial Evolution of Plastic Neural Networks : A Few Key Concepts. In Kowaliw, T., Bredeche, N., and Doursat, R., editors, *Growing Adaptive Machines*, chapter 9, pages 251–261. Springer Berlin Heidelberg.

[86] Neal, M. and Timmis, J. (2003). Timidity: A Useful Mechanism for Robot Control? *Informatica Special Issue on Perception and Emotion Based Reasoning*, 27(2):197–204.

[87] Nelson, A. L. (2014). Embodied Artificial Life at an Impasse Can Evolutionary Robotics Methods Be Scaled? In *2014 IEEE Symposium Series on Computational Intelligence (IEEE SSCI'14)*, Orlando, FL, USA. IEEE.

[88] Nelson, A. L., Barlow, G. J., and Doitsidis, L. (2009). Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345–370.

[89] Newport, C., Kotz, D., Yougu Yuan, Gray, R. S., Jason Liu, and Elliott, C. (2007). Experimental Evaluation of Wireless Simulation Assumptions. *SIMULATION*, 83(9):643–661.

[90] Noble, J. and Franks, D. W. (2003). Social learning in a multi-agent system. *Computing and Informatics*, 22(6):1001–1015.

[91] Nolfi, S. (2011). Behavior and Cognition as a Complex Adaptive System: Insights from Robotic Experiments. In Gabbay, D. M., Hooker, C. A., Thagard, P., and Woods, J., editors, *Philosophy of Complex Systems*, chapter 4, pages 443–463. Elsevier.

[92] Nolfi, S., Bongard, J. C., Husbands, P., and Floreano, D. (2016). Evolutionary Robotics. In Siciliano, B. and Khatib, O., editors, *Springer Handbook of Robotics*, pages 2035–2068. Springer International Publishing, Cham.

[93] Nolfi, S. and Floreano, D. (1999). Learning and evolution. *Autonomous robots*, 7(1):89–113.

[94] Nolfi, S. and Floreano, D. (2000). *Evolutionary robotics : the biology, intelligence, and technology of self-organizing machines*. MIT Press.

[95] Noskov, N., Haasdijk, E., Weel, B., and Eiben, A. E. (2013). MONEE: Using Parental Investment to Combine Open-Ended and Task-Driven Evolution. In Esparcia-Alcázar, A. I., editor, *Applications of Evolutionary Computation*, volume 7835, pages 569–578, Berlin Heidelberg. Springer.

[96] Oxford Dictionaries online (2014). Robot.

[97] Pfeifer, R. and Bongard, J. C. (2006). *How the Body Shapes the Way We Think - A New View of Intelligence*. MIT Press.

[98] Pollack, J. B. and Lipson, H. (2000). The GOLEM project: evolving hardware bodies and brains. In *Proceedings. The Second NASA/DoD Workshop on Evolvable Hardware*, pages 37–42, Palo Alto, CA, USA. IEEE Comput. Soc.

[99] Potdar, V., Sharif, A., and Chang, E. (2009). Wireless Sensor Networks: A Survey. In *2009 International Conference on Advanced Information Networking and Applications Workshops*, pages 636–641. IEEE.

[100] Rakshit, P., Konar, A., Bhowmik, P., Goswami, I., Das, S., Jain, L. C., and Nagar, A. K. (2013). Realization of an Adaptive Memetic Algorithm Using Differential Evolution and Q-Learning: A Case Study in Multirobot Path Planning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(4):814–831.

[101] Renda, F., Giorgio-Serchi, F., Boyer, F., and Laschi, C. (2015). Locomotion and elastodynamics model of an underwater shell-like soft robot. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1158–1165. IEEE.

[102] Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH '87*, volume 21, pages 25–34, New York, New York, USA. ACM Press.

[103] Şahin, E. (2005). Swarm robotics: From sources of inspiration to domains of application. In Şahin, E. and Spears, W. M., editors, *Swarm robotics*, pages 10–20. Springer Berlin Heidelberg.

[104] Saska, M., Chudoba, J., Precil, L., Thomas, J., Loianno, G., Tresnak, A., Vonasek, V., and Kumar, V. (2014). Autonomous deployment of swarms of micro-aerial vehicles in cooperative surveillance. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 584–595, Orlando, FL, USA. IEEE.

[105] Sauze, C. and Neal, M. (2013). Artificial endocrine controller for power management in robotic systems. *IEEE Transactions on Neural Networks and Learning Systems*, 24(12):1973–1985.

[106] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

[107] Segura, C., Coello Coello, C. A., Segredo, E., and Aguirre, A. H. (2016). A Novel Diversity-Based Replacement Strategy for Evolutionary Algorithms. *IEEE Transactions on Cybernetics*, 46(12):3233–3246.

[108] Shi, Z., Tu, J., Zhang, Q., Liu, L., and Wei, J. (2012). A Survey of Swarm Robotics System. In Tan, Y., Shi, Y., and Ji, Z., editors, *Advances in Swarm Intelligence: Third International Conference, ICSI 2012, Shenzhen, China, June 17-20, 2012 Proceedings, Part I*, pages 564–572. Springer Berlin Heidelberg, Berlin, Heidelberg.

[109] Silva, F., Correia, L., and Christensen, A. L. (2014a). Towards Online Evolution of Control for Real Robots with odNEAT. In *3rd International Workshop on the Evolution Physical Systems*.

[110] Silva, F., Urbano, P., and Christensen, A. L. (2012a). Adaptation of Robot Behaviour through Online Evolution and Neuromodulated Learning. In Pavón, J., Duque-Méndez, N. D., and Fuentes-Fernández, R., editors, *Advances in Artificial Intelligence – IBERAMIA 2012*, pages 300–309, Cartagena de Indias, Colombia. Springer Berlin Heidelberg.

[111] Silva, F., Urbano, P., and Christensen, A. L. (2014b). Online Evolution of Adaptive Robot Behaviour. *International Journal of Natural Computing Research*, 4(2):59–77.

[112] Silva, F., Urbano, P., Oliveira, S. M., and Christensen, A. L. (2012b). odNEAT: An Algorithm for Distributed Online, Onboard Evolution of Robot Behaviours. In Adami, C., Bryson, D. M., Ofria, C., and Pennock, R. T., editors, *Artificial Life 13*, pages 251–258. MIT Press.

[113] Sims, K. (1994). Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques - SIGGRAPH '94*, pages 15–22, New York, New York, USA. ACM Press.

[114] Smith, C. and Jin, Y. (2014). Evolutionary Multi-Objective Generation of Recurrent Neural Network Ensembles for Time Series Prediction. *Neurocomputing*.

[115] Soltoggio, A., Bullinaria, J. A., Mattiussi, C., Dürr, P., and Floreano, D. (2008). Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In Bullock, S., Noble, J., Watson, R. A., and Bedau, M. A., editors, *ALIFE IX*, volume 2, pages 569–576, Cambridge, MA. MIT Press.

[116] Stanley, K. O., D'Ambrosio, D. B., and Gauci, J. (2009). A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks. *Artificial Life*, 15(2):185–212.

[117] Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127.

[118] Stepney, S. (2007). Embodiment. In Flower, D. and Timmis, J., editors, *In Silico Immunology*, pages 265–288. Springer US, Boston, MA.

[119] Steyven, A. (2016). Interactive 3D Surface of mEDEA_rf Parameter Sweep Landscape.

[120] Steyven, A., Hart, E., and Paechter, B. (2015). The Cost of Communication: Environmental Pressure and Survivability in mEDEA. In Silva, S., editor, *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference - GECCO Companion '15*, pages 1239–1240, New York, New York, USA. ACM Press.

[121] Steyven, A., Hart, E., and Paechter, B. (2016). Understanding Environmental Influence in an Open-Ended Evolutionary Algorithm. In Handl, J., Hart, E., Lewis, P. R., López-Ibáñez, M., Ochoa, G., and Paechter, B., editors, *Parallel Problem Solving from Nature – PPSN XIV*, volume 9921 LNCS, chapter 86, pages 921–931. Springer International Publishing AG.

[122] Steyven, A., Hart, E., and Paechter, B. (2017). An Investigation of Environmental Influence on the Benefits of Adaptation Mechanisms in Evolutionary Swarm Robotics. In *Proceedings of The Genetic and Evolutionary Computation Conference (GECCO 2017)*, Berlin. ACM.

[123] Sutton, R. and Barto, A. (1998). Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054.

[124] Sznajder, B., Sabelis, M. W., and Egas, M. (2012). How Adaptive Learning Affects Evolution: Reviewing Theory on the Baldwin Effect. *Evolutionary Biology*, 39(3):301–310.

[125] Texas Instruments Inc. (2013). TI cc2420 Datasheet.

[126] Trianni, V. and Nolfi, S. (2011). Engineering the evolution of self-organizing behaviors in swarm robotics: a case study. *Artificial life*, 17(3):183–202.

[127] Urzelai, J. and Floreano, D. (2001). Evolution of adaptive synapses: robots with fast adaptive behavior in new environments. *Evolutionary Computation*, 9(4):495–524.

[128] Vargas, P. A., Di Paolo, E. A., Harvey, I., and Husbands, P. (2014). *The horizons of evolutionary robotics*. MIT Press.

[129] Waibel, M., Floreano, D., and Keller, L. (2011). A Quantitative Test of Hamilton's Rule for the Evolution of Altruism. *PLoS Biology*, 9(5):e1000615.

[130] Walker, J. H., Garrett, S. M., and Wilson, M. S. (2006). The balance between initial training and lifelong adaptation in evolving robot controllers. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 36(2):423–32.

[131] Wang, Q., Hempstead, M., and Yang, W. (2006). A Realistic Power Consumption Model for Wireless Sensor Network Devices. In *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, volume 1, pages 286 – 295.

[132] Watson, R. A., Ficici, S. G., and Pollack, J. B. (1999). Embodied evolution: embodying an evolutionary algorithm in a population of robots. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, pages 335–342. IEEE.

[133] 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.

[134] Winfield, A. F. T. (2012). *Robotics: A Very Short Introduction*. Oxford University Press.

Appendix A

Publications

A.1 Improving Survivability in Environment-driven Distributed Evolutionary Algorithms through Explicit Relative Fitness and Fitness Proportionate Communication

Hart, E., Steyven, A., & Paechter, B. (2015). Improving Survivability in Environment-driven Distributed Evolutionary Algorithms through Explicit Relative Fitness and Fitness Proportionate Communication. In S. Silva (Ed.), *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15* (pp. 169–176). New York, New York, USA: ACM Press.

Improving Survivability in Environment-driven Distributed Evolutionary Algorithms through Explicit Relative Fitness and Fitness Proportionate Communication

Emma Hart
School of Computing
Edinburgh Napier University
Edinburgh, Scotland, UK
e.hart@napier.ac.uk

Andreas Steyven
School of Computing
Edinburgh Napier University
Edinburgh, Scotland, UK
a.steyven@napier.ac.uk

Ben Paechter
School of Computing
Edinburgh Napier University
Edinburgh, Scotland, UK
b.paechter@napier.ac.uk

ABSTRACT

Ensuring the integrity of a robot swarm in terms of maintaining a stable population of functioning robots over long periods of time is a mandatory prerequisite for building more complex systems that achieve user-defined tasks. $mEDEA$ is an environment-driven evolutionary algorithm that provides promising results using an implicit fitness function combined with a random genome selection operator. Motivated by the need to sustain a large population with sufficient spare energy to carry out user-defined tasks in the future, we develop an explicit fitness metric providing a measure of fitness that is relative to surrounding robots and examine two methods by which it can influence spread of genomes. Experimental results in simulation find that use of the fitness-function provides significant improvements over the original algorithm; in particular, a method that influences the frequency and range of broadcasting when combined with random selection has the potential to conserve energy whilst maintaining performance, a critical factor for physical robots.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

Keywords

Evolutionary Robotics; Environment-driven; On-line Evolution

1. INTRODUCTION

Recent advances in technology both in hardware and software [8] have fuelled visions of swarms of robots being sent to remote or hazardous environments in which they will need to survive over long periods of time. Environments will be unknown and potentially dynamic, requiring autonomous adaptation by the swarm. This has led to a number of recent efforts to study evolution within a swarm as a mechanism for driving adaptation, as opposed to a mechanism for optimising an explicit fitness function, as is common in much work within evolutionary robotics. Montanier [12] notes that this step is in fact a pre-requisite to studying any kind of user-driven task behaviours within a robotic swarm in an open-environment, as the former cannot be achieved if the integrity of the swarm is compromised.

This type of evolution is often referred to as *environment-driven evolution* [2]. Typical approaches such as [14] remove the need for any central control, resulting in algorithms that perform distributed and online evolution. An additional feature of environment-driven algorithms is that no explicit fitness function is defined: instead, mate selection and reproduction depend on selection pressure provided only by the environment yet need to lead to stable populations. A recent example of this is the mEDEA algorithm (minimal Environment driven Distributed Evolutionary Adaptation) [3, 4, 5]. mEDEA relies on an implicit fitness function that results from two potentially conflicting motivations for a robot: an *extrinsic* motivation to cope with environmental constraints in order to maximise survival ability and an *internal* motivation to spread its genomes across the population in order to survive. A complex trade-off exists in which behaviours that maximise mating opportunities might negatively impact survival efficiency, e.g. failing to maintain stable energy levels; as a result, mEDEA (or an environment-driven EA) must find some equilibrium between the two states.

The original version of mEDEA exploited a simple strategy in which a robot continuously broadcast its genome — this can be received and stored by any robot currently within communication range. At the end of a generation, each robot makes a *random* selection from its set of stored genomes, applies a mutation operator, and then replaces its current genome, exactly as in a (1,1) Evolution Strategy [1]. Although there is no selection pressure on an individual basis, from a global perspective, the most widely spread genomes will be selected more often on average. While this achieved success in evolving stable populations in an open-ended environment, it is of interest to attempt to improve both the size of the swarms maintained and their net energy levels, in order that complex user-defined tasks can be added in future. It is reasonable to assume that spare energy, over and above that required to survive can be exploited to achieve complex tasks, whilst a large swarm offers more potential in terms of the tasks that might be accomplished.

Within mEDEA, the evolutionary mechanism differs from natural evolution that also drives adaptation in a number of respects. Firstly, there is no form or crossover — variation

http://dx.doi.org/10.1145/2739480.2754688

is provided only by a mutation operator — and hence the emphasis is on the spreading of *genomes* rather than *genes* as proposed by the Selfish Gene paradigm of Dawkins [7]. Secondly, as all robots broadcast their genome continuously and with fixed radius, each robot has an equal opportunity to pass on its genome, regardless of its quality. This latter point is clearly not true in many natural systems. In nature, an individual's chance of reproduction is related to its fitness relative to other individuals in its vicinity. Additionally, individuals mate selectively, choosing partners based on some estimation of their quality. Some species broadcast their quality through visual or behavioural displays: a peacock displays its tail feathers, a bird of paradise 'dances'. Fitter individuals can attract the attention of a greater number of potential mates.

Inspired by nature, we investigate the effect of introducing a relative fitness measure into mEDEA. A robot makes an estimate of its fitness to survive relative to those within its broadcast range, thereby maintaining the distributed nature of the algorithm. The value can be used in two ways:

- An individual robot can make an informed rather than random selection from the genomes it has received according to the relative fitness value
- The relative fitness value can be used to influence the broadcasting behaviour of a robot to provide a bias towards the spread of good genomes

The latter point changes the nature of the reproductive strategy used in mEDEA from a 'promiscuous' one in which there is indiscriminate broadcasting of a genome, to one in which the *spread* of a genome (and therefore the probability of it being collected) is dependent on its quality. Two novel methods for influencing broadcasting are introduced: the first causes the robot to adapt the probability with which it broadcasts based on its fitness. The second causes the robot to adapt the range over which it broadcasts its genome. As in the original version of mEDEA, robots still make a random selection from collected genomes; however, due to the biased broadcasting methods, on average good quality genomes are more likely to be collected than poorer ones.

Results for all approaches are compared to the original mEDEA algorithm. Experiments show that both methods perform equally well compared to the original algorithm. Note however that broadcasting in the physical world is an energy consuming operation; methods that reduce this energy in order to save battery by either reducing the range or frequency of broadcasting are likely to be of considerable benefit.

2. RELATED WORK

mEDEA was first proposed in [3]. The system was tested under two scenarios: the first evaluated mEDEA in an environment providing limited pressure in which energy is ignored and an agent survives as long as it collects at least one genome. In the second, environmental pressure is introduced by forcing robots to compete for limited resources in order to gain energy. The algorithm was demonstrated to be both efficient with regard to providing distributed evolutionary adaptation in unknown environments and robust to unpredicted changes in the environment. Furthermore, given its lightweight nature, it was predicted to be suitable for hardware and software setups that have limited computation.

Haasdijk *et al* [10] extended mEDEA so that in addition to surviving and operating reliably in an environment, a robot could also perform user-defined tasks. Survivor selection is driven by the environment, whereas parent selection is driven by task performance. Their new framework MONEE (Multi-Objective aNd open-Ended Evolution algorithm) showed that task-driven behaviour can be promoted without compromising environmental adaptation. Robots accumulate credit for accomplishing particular tasks — this credit value is transmitted along with a genome, and is utilised in a fitness function to select parents, replacing the random selection seen in mEDEA. The basic mEDEA setup is also altered by adding an 'egg' phase that occurs at periodic intervals: during this phase, the stationary robot collects genomes from passing robots — no genomes are collected whilst a robot is moving.

Perez *et al* [13] study the impact of adding explicit selection methods to the mEDEA algorithm in a task-driven scenario. They evaluate four selection methods that induce different intensities of selection pressure, using tasks that include obstacle avoidance and foraging, finding that higher selection pressure results in improved performances, especially in more challenging tasks.

Watson *et al.* [15] proposed an completely decentralised algorithm for embodied evolution (EE). In their Probabilistic Gene Transfer Algorithm (PGTA) robots exchange randomly selected genes through short range communication. The algorithm differs from other approaches in that is doesn't have a dedicated variation and replacement steps. Each robot holds a single genome of which only individual genes are replaced at runtime. Transmission frequency and gene acceptance are based on the explicit fitness value of the respective robot, which reflects its performance on a task.

Another strand of work of work worth mentioning is that of Stanley et al. [11] who introduced the notion of *Novelty Search* that promote the discovery of solutions that differ from the ones already evolved. Solutions are selected for novelty rather than objective fitness hence do not require an explicit fitness function; however, the algorithm requires global knowledge to calculate novelty hence cannot be used as described in a purely distributed evolutionary algorithm.

Our proposal provides a novel contribution to the line of work on mEDEA that started in [3] in adding an explicit fitness function that is defined in relation to survival ability rather than task performance, and using this to providing two methods of influencing the rate of spread of good genomes through the population. By providing a more robust mechanism for maintaining swarm integrity than previous work, we pave the way towards future work in which it is possible to add user-defined tasks to a swarm operating in an unknown environment, thereby increasing its utility.

3. ALGORITHM DESCRIPTION

mEDEA utilises an agent driven by a control architecture whose parameters are defined by the currently active genome. The genome defines the weights of an Elman recurrent neural network (RNN) consisting of 43 sensory inputs and 2 motor outputs (translational and rotational speeds). 8 ray-sensors are distributed around the robot's body. They detect the proximity to the nearest object, the presence of walls and other robots, whether it belongs to the same group and the relative orientation between the two robots. An energy level input feeds the current level into the network. A distance and angle sensor give the direction to the nearest energy token. The RNN has 1 hidden layer with 8 nodes,

thus 434 weights are defined by the genome. This setup is adapted from [3]. The original mEDEA algorithm is defined in algorithm 1 — only the single step *broadcast()* is modified in this paper, therefore the reader is referred to [3] for a detailed description of the concepts that underpin its design and the specification of each of the other methods.

```
genome.randomInitialise();
while forever do
    if genome.isNotEmpty() then
        agent.load(genome);
    end
    for iteration = 0 to lifetime do
        if agent.isAlive() and genome.isNotEmpty()
        then
            agent.move();
            broadcast(genome);
        end
    end
    genome.empty();
    if genomeList.size() > 0 then
        genome =
        applyVariation(select_random(genomeList));
        if agent.isAlive() == false then
            agent.setAliveState(true);
        end
    end
    else
        agent.setAliveState(false);
    end
    genomeList.empty();
end
```

Algorithm 1: Pseudo code of the original mEDEA algorithm by Bredeche *et al.* [4]

In brief: for a fixed period, robots move according to their control algorithm, broadcasting their genome that is received and stored by any robot within range. At the end of this period, a robot selects a random genome from its list of collected genomes and applies a variation operator. This takes the form of a Gaussian random mutation operator, inspired from Evolution Strategies [1] which can be easily tuned through a σ parameter. Robots that have not collected any genomes become inactive, thus reducing the population size.

At the start of each generation, a robot is initialised with an energy E_0. Every time step, the energy value is decreased by one unit. Energy *tokens* are scattered in the environment. If a robot moves over a token, its energy is increased by an amount E_{token}. An robot with energy=0 remains stationary for the remainder of the generation.

The next section describes the proposed modifications to mEDEA algorithm that we dub $mEDEA_{rf}$ — mEDEA with relative fitness.

3.1 mEDEA$_{rf}$

A robot calculates an estimate of its own fitness to survive based on the balance between energy lost and energy gained, δ_E: this term is initialised to 0 at $t = 0$ and is decreased by 1 at each time-step, and increased by E_{token} if it crosses an energy token. Given δ_E, a robot calculates its fitness relative to the robots in a range r according to equation 1, where f'_i is the relative fitness of robot i at time t, $mean_{sub_i}$ is the mean δ_E of the robots within the subpopulation defined by all robots in range r of robot i, and sd_{sub_i} is the standard deviation of the δ_E of the subpopulation.

$$f'_i(t) = \frac{\delta_i(t) - mean_{sub_i}(t)}{sd_{sub_i}(t)} \tag{1}$$

Note that f'_i is defined in relation to the ability of the robot to *survive* in the environment; it records the net energy of a robot, accounting for energy expended and energy gained by locating tokens. It differs from the explicit task-driven fitness functions investigated by [13] that were concerned only with task-driven selection, i.e optimising performance on defined tasks; although one task investigated was a foraging task that involved collecting tokens in a similar manner to the one used here, the tokens did not influence robot survival in that they did not contribute to the energy of the robot and therefore its ability to stay alive.

The new explicit fitness function can be exploited in two ways: it can either be transmitted with a genome and used by an individual within a selection function or it can be used to influence the rate at which a genome is broadcast, thereby indirectly affecting its chances of being selected for reproduction. The two approaches are described below.

3.1.1 Explicit selection mechanisms

The $select_{random}(genomeList)$ method in mEDEA can easily be replaced with an informed selection method that uses the relative fitness measure to discriminate between genomes. We investigate three well-known selection strategies:
tournament-selection, roulette-wheel selection and an elitist select-best strategy.

3.1.2 Biasing broadcasting of genomes

Alternatively, the spread of genomes can be biased by adapting the *broadcast()* step in algorithm 1. In mEDEA, robots make a random selection from their list of collected genomes at the end of each generation. We propose two new methods, both of which bias the spread of genomes throughout the population in favour of higher quality ones based on a robot's estimation of its fitness f' relative to those in its immediate surroundings.

- *broadcast_radius()* adapts the range at which a robot broadcasts depending on f'
- *broadcast_probability()* broadcast at a fixed range r with the probability depending on f'

Given f'_i, we define the probability of a robot broadcasting using equation 2 that simply describes a function that returns a probability 0 if f'_i is less than d_0, probability=1 if f'_i is greater than $d_{\max}$ standard deviations away from the mean, and linearly interpolated between 0 and 1 otherwise.

$$p_i(t) = \begin{cases} 0 & f'_i(t) <= d_0 \\ \frac{f'_i(t) - d_0}{d_{\max} - d_0} & d_0 \leq f'_i(t) < d_{\max} \\ 1 & f'_i \geq d_{\max} \end{cases} \tag{2}$$

For the *Broadcast_probability()* method, shown in *case*1 in algorithm 2, the probability $p_i(t)$ is used directly to determine whether a robot broadcasts. For *Broadcast_radius()*, the probability $p_i(t)$ is converted to a broadcasting range between 0 and a value $r_{\max}$ according to equation 3 — the higher the relative fitness, the greater the broadcast range.

Note that range increases with the square root of the probability in order to maintain a proportional increase in broadcast *area*.

$$r_i(t) = r_{\max} * \sqrt{p_i(t)} \quad (3)$$

Both methods result in robots that have higher relative fitness broadcasting their genome more than those with lower relative fitness, hence biasing the quality of genomes that a receiving robot collects. At the end of each generation, a random selection of genome is made from those collected as in mEDEA.

```
R ← all robots in simulation;
foreach robot i in R do
    N_i ← getRobotsWithinMaxRadius(R);
    if N_i > 0 then
        f'_i ← calculate fitness relative to N_i ;   // eq. 1
        p_i ← convert f'_i to probability;           // eq. 2
    else
        p_i = 0
    end
    switch exp do
        // vary probability
        case 1
            if p_i > rand() then
                broadcast(r_max, currentGenome, σ);
            end
        end
        // vary broadcast radius
        case 2
            r_i ← adjustRadius(p_i);                 // eq. 3
            foreach robot j in N_i do
                if distance(i,j) < r_i then
                    broadcast(r_i, currentGenome, σ);
                end
            end
        end
    endsw
end
```

Algorithm 2: Pseudo code of the algorithm that is executed at every discrete time step of the simulation.

4. EXPERIMENTS

Three sets of experiments were undertaken, exploring the effects of using the explicit selection mechanism, biasing spread of genomes through altering the broadcasting mechanism, and finally biasing spread *and* using explicit selection.

Explicit selection mechanisms.

The first set of experiments investigates the hypothesis that replacing the random selection method in mEDEA with a selection method that selects based on the relative fitness value will increase both the average δ_E of the population and number of robots alive N_{alive} at the end of the final generation when compared to the original mEDEA algorithm. Three selection methods are investigated: binary tournament, roulette-wheel and an elitist select-best. These experiments are labelled E1 (mEDEA), E1+t, E1+rw, E1+b to denote the different selection methods.

Biased broadcasting of genomes.

Experiments were designed to evaluate the following hypotheses:

1. Biasing the spread of genomes via adapting the *probability* that a robot broadcasts based on its relative fitness will improve the average δ_E of the population and N_{alive} compared to the original mEDEA algorithm.

2. Biasing the spread of genomes via adapting the *range* over which a robot broadcasts based on its relative fitness will improve the average δ_E of the population compared to the original mEDEA algorithm.

Note however that the new methods *broadcast_probability()* and *broadcast_range()* introduce *two* adaptations compared to the original algorithm: (1) the broadcast probability (and therefore range) is variable across the population and (2) the broadcast probability (and therefore range) is determined by *relative fitness*. Thus in order to show that any improvement in average δ_E can be attributed to the effect of introducing the *relative fitness* term rather than simply a random variation, we perform additional control experiments as follows:

Rather than calculating the relative fitness of a robot according to equation 1 using its own $\delta_i(t)$, we simply replace it with x_i — a random number drawn from a normal distribution with mean $\Delta(t)$ and $sd\Delta(t)$, where the Δ terms refer to the mean and standard deviation of the fitness of the *global* population (the global fitness is used simply to ensure that the random value is drawn from an appropriate range). New methods *broadcast_randomProbability()* and *broadcast_randomRange()* then use equations 2 and 3 as previously described.These methods are introduced merely to perform rigorous control experiments: we do not suggest that this method would be used in practice as it requires the calculation of a global parameter, contrary to the distributed nature of the algorithm.

Five different experiments are performed, where E1-E3 are controls and E4 and E5 evaluate the new methods.

- E1 records the mean δ_E of the robot population and the number of active robots at the end of the final generation using only the original version of mEDEA
- E2 records the same metrics as above using *broadcast_randomProbability()*
- E3 records the same metrics as above using *broadcast_randomRange()*.
- E4 records the same metrics as above using *broadcast_probability()*.
- E5 records the same metrics as above using *broadcast_range()*.

4.1 Methodology

All experiments use Roborobo! by Bredeche et al. from [6], as in the original simulations described with mEDEA. Roborobo! is a multi-platform, highly portable, robot simulator for large-scale collective robotics experiments. With respect to other robotic simulators, Roborobo! combines (pseudo-)realistic modelling with fast-paced simulation and thus falls somewhere in-between very realistic frameworks such as Player/Stage [9] that tend to be very slow and agent-based tool such as MASON that are extremely simplified with respect to the environment. It focuses solely on large-scale swarms of robots in a 2D environment and is based on a Khepera/ePuck model and has already been used in more than a dozen published research papers mainly concerned

with evolutionary swarm robotics, including environment-driven self-adaptation and distributed evolutionary optimization, as well as online onboard embodied evolution and embodied morphogenesis.

All parameters used in the experiments are given in table 1. Simulation parameters are based on the original papers. Experimental parameters were chosen following limited empirical tuning. The maximum broadcasting range requires sensible selection and should be chosen proportional to the arena size.

Simulation parameters	
Arena size	1024 pixel by 1024 pixel
Number of robots	100
Robot lifetime	1500 iterations
Food regrow time	500 iterations
Sensor range	32 pixel
Chromosome length	434
Experimental parameters	
Number of runs	30
Maximum generations	500
Number of energy tokens	800
Energy value of token	100
Start energy	1200
Maximum range $r_{\max}$	64
d_0	0
$d_{\max}$	2

Table 1: Simulation and Experimental Parameters for all experiments

5. RESULTS AND ANALYSIS

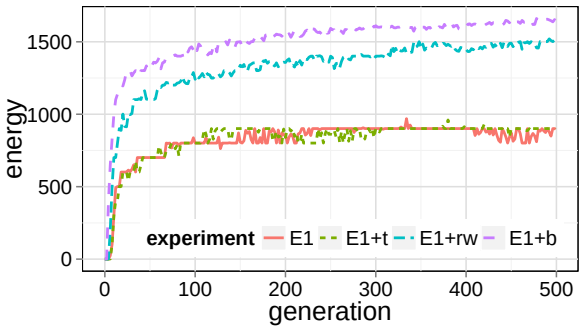
Explicit selection by individuals.

Results from the experiments E1, E1+t, E1+rw, E1+b in which the $select_{random}(genome\_list)$ method in algorithm 1 is replaced with a selection method are shown in figure 1, which compares the median[1] energy and agents alive over 30 repeated runs for each of the four experiments listed. Adding an explicit selection method based on a relative fitness value relating to the ability of another robot to survive over the generation has significant effect in the case of *roulette wheel* and *best* selection when compared to mEDEA. Both of these methods exert high selection pressure. In contrast, the low-pressure *tournament* selection method shows little difference to the random selection method of mEDEA. Wilcoxon rank-sum tests confirm that the roulette-wheel and best methods provide significantly different results for both energy and N_{alive}, while no significant difference is observed with the tournament selection method for either metric. The highest pressure selection method *best'* outperforms *roulette-wheel* with statistically significant results at the 0.05 significance level.

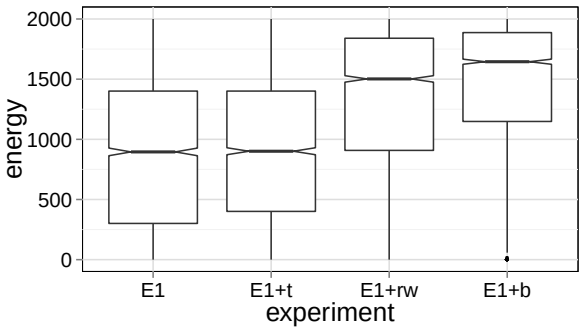
Biasing genome spread.

The next set of experiments examines the results of using the two new broadcasting methods, comparing results to the original mEDEA algorithm. Figure 3 clearly shows that experiments E4 and E5 that introduce the new broadcasting methods outperform both the original mEDEA algorithm and the two control experiments. A Wilcoxon rank-sum test with significance level $\alpha = 0.05$ showed that the difference

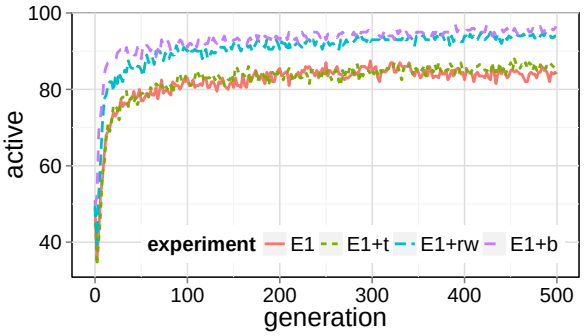
[1]as a Shapiro-Wilk test showed that the results were not normally distributed



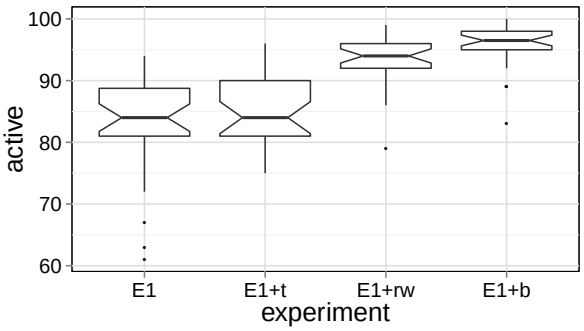
(a) Energy



(b) Energy at gen 500



(c) Active robots



(d) Active robots at gen 500

Figure 1: Explicit selection added to the mEDEA algorithm

in final energy at generation 500 for both E4 and E5 was statistically different to E1, E2 and E3, but that there is no statistical difference between E4 and E5. The fact that E4

and E5 differ significantly from controls E2 and E3 show that the differences in performance are not simply attributable to varying the broadcast rate or range, but must be related to the fact that the broadcast rate and range are adjusted according to the estimate of fitness f' calculated by each robot. A corresponding pattern is observed when examining the number of active robots. Plots (e) and (f) within figure 3 clearly show that the number of genomes broadcast significantly decreases with respect to the original methods, but that is compensated for using the higher environmental pressure achieved by adapting what is broadcast based on the quality estimate f'.

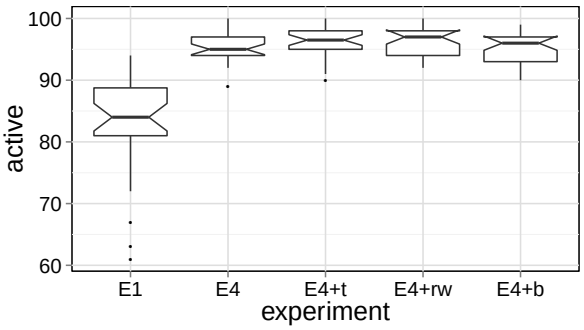
In the original mEDEA, as all robots broadcast indiscriminately at the same fixed range, a very weak selection pressure is created that results in genomes that have spread more widely having more chance of being selected if we consider the population as a whole. Behaviours that lead to a robot coming in contact with more robots will result more spreading of genomes and thus on average, a higher probability of generating future offspring. In contrast, the more discriminate methods of broadcasting proposed in this paper create higher selection-pressure: genomes that have higher relative fitness have more chance of being received by other robots than lower fitness ones and thus are more likely to be randomly selected.

Combining explicit selection with biased broadcasting. Finally, we investigate the effect of combining explicit selection within an individual with biased broadcasting, testing each of the three selection methods in combination with E4 and E5. Figure 2 shows boxplots of the results obtained from using the two $mEDEA_{rf}$ variants and mEDEA. Each of the $mEDEA_{rf}$ variants is significantly better in terms of energy level and active robots compared to standard mEDEA using an explicit selection method, confirmed using a Wilcoxon Rank-Sum test with a significance level $\alpha = 0.05$.

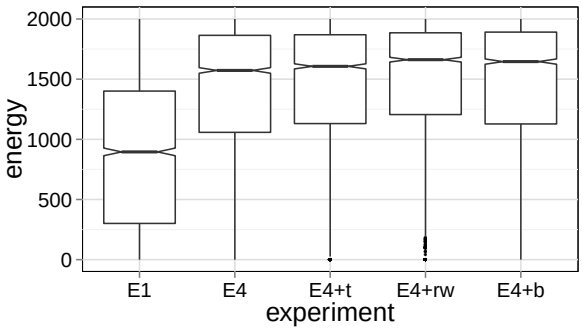
In order to easily contrast the new methods to the original algorithm, figure 4 compares mEDEA, mEDEA+rw, and the two new broadcasting methods combined with roulette wheel selection and in table 2 we use the Wilcoxon test to compare pairs of experiments with and without explicit selection (indicated by $Ei + s\ Ei$ respectively). Statistically significant results are shown in bold.

The following comments can be made that summarise all experiments. Where claims are made, they are evidenced by data that is statistically significant as show in the table.

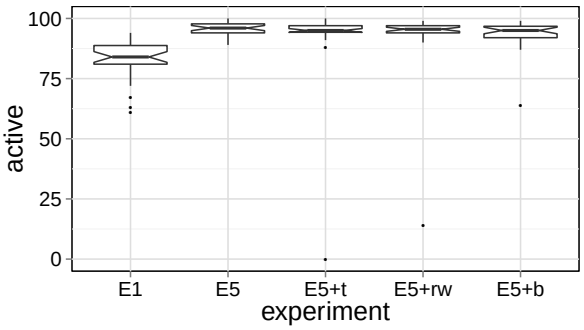
- Coupling the standard mEDEA algorithm with a high-pressure explicit selection method results in a more robust and sustainable population (higher energy and more alive robots) than the standard mEDEA. However, using a low-pressure explicit selection method does not result in any statistical difference.
- The new methods of biasing the spread of genomes based on relative fitness combined with a random selection method by individiual robts (E4, E5) result in a more robust and sustainable population than mEDEA (higher energy and more alive robots). However there are no discernible differences between the two new methods.
- Coupling the methods for biasing spread of genomes (E4, E5) with an explicit selection method by individuals robots improves on the standard mEDEA but in most cases does not provide any significant advantage over biasing the spread and using random selection,



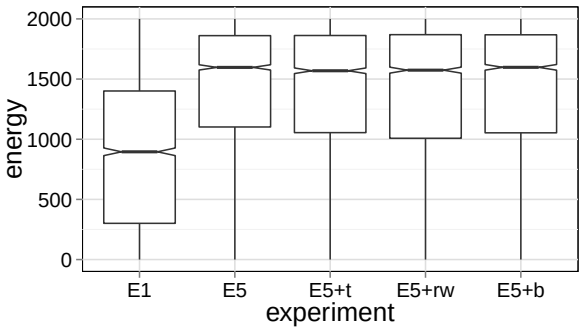
(a) broadcast_probability(): active robots



(b) broadcast_probability:energy

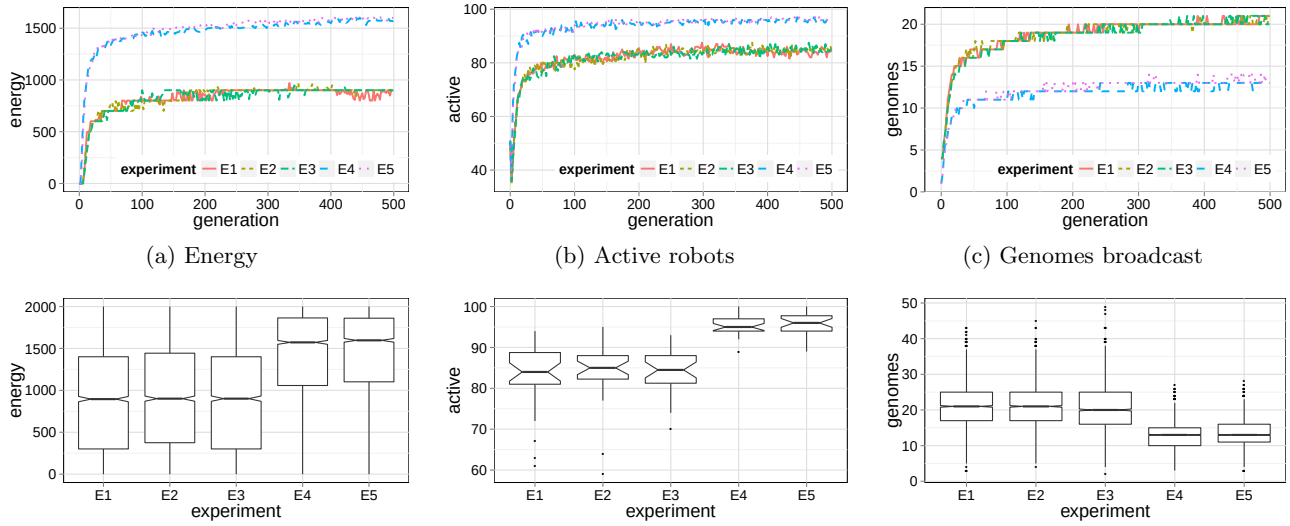


(c) broadcast_range:active robots



(d) broadcast_range():energy

Figure 2: Combining the biased broadcasting of genomes with explicit selection by individuals.



(a) Energy (b) Active robots (c) Genomes broadcast

(d) Energy at gen 500 (e) Active robots at gen 500 (f) Genomes broadcast at gen 500

Figure 3: mEDEA, control experiments and biased broadcasting: figures show the energy, number of active robots and genomes received for each of the experiments E1-E5

with the exception of improving energy levels in the case of $E4 + s$ when compared to E4 alone.

- Using roulette-wheel selection combined with standard mEDEA outperforms the two experiments in which the spread of genomes is biased but individuals apply random selection in terms of sustaining higher levels of energy within the population, but has no significant effect on the size of the sustained population. However, note that using the explicit selection method comes at a potentially high cost in terms of the number and range of broadcasts required to implement this when compared to the biased-broadcasting methods.

In summary, the results show that using mEDEA with a relative fitness function that either promotes spread of good genomes (via biasing what is transmitted) or promotes selection of genomes with high energy values (explicit selection) result in swarms that sustain high energy levels and high percentages of active robots when compared to the original version.

However, when considering real, physical robots, it should be clear that broadcasting comes with an overhead in terms of the energy required to communicate. Two factors influence the cost of broadcasting in energy terms — the number of broadcasts made and the broadcast range. For the explicit selection and *broadcast_range()* methods, the same number of broadcasts are made — however *broadcast_range()* results in a range of broadcast distances $\leq r_{\max}$, whereas the explicit selection method combined with mEDEA always broadcasts at $r_{\max}$, thus utilising greater energy. The *broadcast_probability()* method directly reduces the number of broadcasts made with respect to mEDEA as weaker robots broadcast less on average, thus saving energy. Hence, although both methods of influencing genome choice can provide similar results, the methods that modulate the broadcasting behaviours are preferable in reducing communication overhead. When considering real-robots this factor can have a significant impact on survival ability — in many real-life scenarios, the ability to prolong battery life by reducing energy usage might well be critical.

		E1	E4	E5
E4	Energy	**< 2.2e-16**		
	Alive	**3.079e-10**		
E5	Energy	**< 2.2e-16**	0.3521	
	Alive	**4.112e-10**	0.4885	
E1+s	Energy	**< 2.2e-16**	**2.13e-05**	**1.963e-07**
	Alive	**6.207e-08**	0.0684	0.1994
E4+s	Energy	**< 2.2e-16**	**1.227e-07**	**1.007e-05**
	Alive	**1.222e-10**	0.1076	0.2793
E5+s	Energy	**< 2.2e-16**	0.5267	0.1288
	Alive	**3.287e-09**	0.7715	0.7658

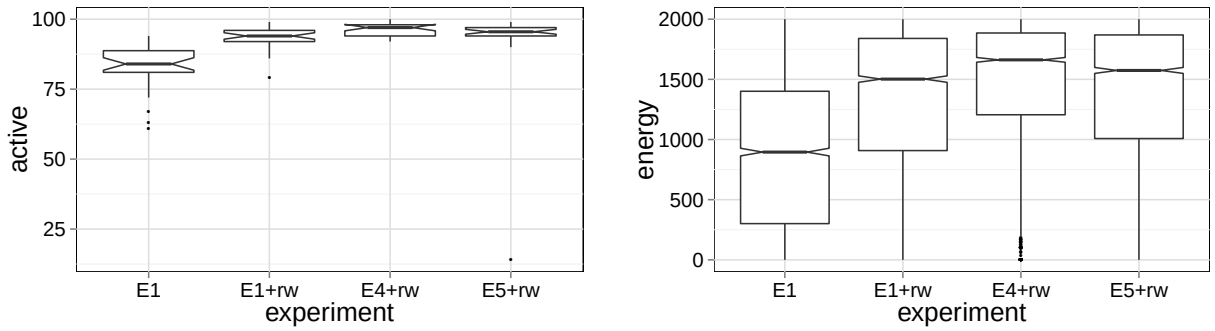
Table 2: p-values obtained from applying Wilcoxon's Rank Sum Test across pairs of experiments, including biased-broadcast only and biased broadcasting coupled with an explicit selection method

6. CONCLUSION AND FUTURE WORK

The paper has provided a number of extensions to an existing Environment Driven Evolutionary Adaptation algorithm — mEDEA. The goal of the work is to show that the integrity of the swarm can be maintained in a more robust manner than in the original work, while still retaining the original distributed and online flavour of the algorithm by using a fitness function that indicated fitness to survive. Having introduced the new fitness function, two new methods were described that adapted either the broadcast range or the probability of broadcasting of a robot, based on its estimate of its relative fitness.

This biases the spread of genomes through the population, with robots that are relatively fitter than their neighbours able to spread their genomes more effectively: individual robots perform a random selection from their store of (now biased) genomes. A thorough analysis of the experimental results shows that a considerable gain in performance is achieved, both in the number of active robots at the end of a fixed period of evaluation, and in the energy levels sustained by those robots.

The new fitness function was also evaluated within an explicit selection method. Experiments showed that this



(a) Active robots at gen 500

(b) Energy at gen 500

Figure 4: Comparison between vanilla mEDEA, mEDEA with explicit selection by individuals and $mEDEA_{rf}$ + explicit selection by individuals

also provided significant improvements over mEDEA, and slightly outperformed the biased broadcast methods in terms of energy sustained. However, as we described above, this comes with a higher cost than either of the biased broadcast methods in terms of the energy used in transmitting. This might be detrimental in a number of real-world scenarios and hence the lower energy-cost methods are preferred.

An obvious extension to this work will include accounting for the cost of broadcasting when calculating the net energy of a robot that is used by the fitness function, to test whether this will differentiate results from the two sets of experiments desribed in this paper. The results provide a robust platform for future experimentation in which user-defined tasks can be added to examine the effects of mixing environment and task driven evolution with a more robust swarm. Given the lightweight nature of the algorithm, an obvious way forward would also include testing on a real hardware platform in the near future.

Acknowledgements

The authors would like to thank Nicolas Bredeche for providing the improved version of Roborobo!, the base of the mEDEA algorithm implementation on which the experiments are based on and the inspirational discussions at the FOCAS summer school 2014.

7. REFERENCES

[1] H.-G. Beyer and H.-P. Schwefel. Evolution strategies – a comprehensive introduction. *Natural Computing*, 1:3–52, 2001.

[2] R. Bianco and S. Nolfi. Toward open-ended evolutionary robotics: evolving elementary robotic units able to self-assemble and self-reproduce. *Connection Science*, 16(4):227–248, 2004.

[3] N. Bredeche and J.-M. Montanier. Environment-driven embodied evolution in a population of autonomous agents. In R. Schaefer, C. Cotta, J. Kołodziej, and G. Rudolph, editors, *Parallel Problem Solving from Nature, PPSN XI*, volume 6239, pages 290–299, Krakov, Poland, 2010. Springer Berlin Heidelberg.

[4] N. Bredeche and J.-M. Montanier. Environment-driven Open-ended Evolution with a Population of Autonomous Robots. In *Evolving Physical Systems Workshop*, East Lansing, United States, 2012.

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

[6] N. Bredeche, J.-M. Montanier, B. Weel, and E. Haasdijk. Roborobo! a Fast Robot Simulator for Swarm and Collective Robotics, *CoRR*, abs/1304.2, Apr. 2013.

[7] R. Dawkins. *The Selfish Gene*. 1976.

[8] A. E. Eiben. Grand Challenges for Evolutionary Robotics. *Frontiers in Robotics and AI*, 1(4):74, June 2014.

[9] B. Gerkey, R. T. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR 2003)*, number Icar 2003, pages 317–323, Coimbra, Portugal, 2003.

[10] E. Haasdijk, B. Weel, and A. E. Eiben. Right on the MONEE. In C. Blum, editor, *GECCO2015*, pages 207–214, Amsterdam, The Netherlands, 2013. ACM New York, NY, USA.

[11] J. Lehman and K. O. Stanley. Exploiting Open-Endedness to Solve Problems Through the Search for Novelty. In *Artificial Life XI*, pages 329–336. MIT Press, 2008.

[12] J.-M. Montanier. *Environment-driven Distributed Evolutionary Adaptation for Collective Robotic Systems*. PhD thesis, Université Paris Sud - Paris XI, Mar. 2013.

[13] I. n. F. Pérez, A. Boumaza, and F. Charpillet. Comparison of Selection Methods in On-line Distributed Evolutionary Robotics. In *ALife2014*, 2014.

[14] F. Silva, P. Urbano, S. Oliveira, and A. L. Christensen. odNEAT: An Algorithm for Distributed Online, Onboard Evolution of Robot Behaviours. In C. Adami, D. M. Bryson, C. Ofria, and R. T. Pennock, editors, *Artificial Life 13*, pages 251–258. MIT Press, July 2012.

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

A.2 The Cost of Communication: Environmental Pressure and Survivability in mEDEA

Steyven, A., Hart, E., & Paechter, B. (2015). The Cost of Communication: Environmental Pressure and Survivability in mEDEA. In S. Silva (Ed.), *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference - GECCO Companion '15* (pp. 1239–1240). New York, New York, USA: ACM Press.

The Cost of Communication: Environmental Pressure and Survivability in mEDEA

Andreas Steyven
School of Computing
Edinburgh Napier University
Edinburgh, Scotland, UK
a.steyven@napier.ac.uk

Emma Hart
School of Computing
Edinburgh Napier University
Edinburgh, Scotland, UK
e.hart@napier.ac.uk

Ben Paechter
School of Computing
Edinburgh Napier University
Edinburgh, Scotland, UK
b.paechter@napier.ac.uk

ABSTRACT

We augment the mEDEA algorithm to explicitly account for the costs of communication between robots. Experimental results show that adding a costs for communication exerts environmental pressure to implicitly select for genomes that maintain high energy levels. We compare our two methods which vary broadcasting based on the individuals fitness to vanilla mEDEA bundled with an explicit selection method under these new conditions and find that biasing broadcasting has a negative effect on survivability.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

Keywords

Evolutionary Robotics; Environment-driven; Online Evolution

1. INTRODUCTION

In previous research [4] we implemented an explicit fitness measure into mEDEA [1] that influences the spread of genomes through the population in order to increase survivability, thus, ensure the integrity of the swarm. We dubbed the algorithm mEDEA$_{rf}$ — mEDEA with relative fitness.

In mEDEA a robot's controller is encoded in its genome which is constantly broadcast in a close range. A random selection from a list of the received genomes during the generation determines the controller for the next generation. In mEDEA$_{rf}$ the broadcasting is varied either by reducing the range or probability of broadcasting depending on the explicit fitness value to favour the spread of genomes that produce controller which maintain a higher energy balance. We compared mEDEA$_{rf}$ to mEDEA where the random selection is replace with an explicit selection method.

Our results suggested that although using the relative fitness value in either way improved survivability, adjusting the broadcasting mechanism should conserve energy. In this paper we derive an energy model to account for the cost for communication and implemented it into both algorithms to test this hypothesis. We give a briefly overview of the experiments conducted and the results obtained.

http://dx.doi.org/10.1145/2739482.2768489

2. METHOD

Two different methods were introduced in mEDEA$_{rf}$ that change the broadcasting mechanism in mEDEA by varying a) the probability to broadcast and b) adjusting the broadcast radius, in proportion to the fitness value. Both mechanisms bias the broadcast towards fitter individuals. For an in-depth description of the mEDEA algorithm and the details of our proposed modifications the reader is referred to [1] and [4] respectively.

To calculate the cost of communication for the simulated ePuck robots in Roborobo [2], we derive an energy model based on the Free-Space Model [5] (equation 1) which is used in wireless sensor network simulations. This field of research makes extensive use of the low power communication modules used in experiments[1] using the ePuck robot platform [6].

$$E_{\text{tx}}(n, d) = n \times E_{\text{elec}} + n \times \epsilon_{\text{amp}} \times d^2 \quad (1)$$

E_{elec} is the basic charge to run the module, ϵ_{amp} the costs for signal amplification which is multiplied by the *d*istance squared and n represents the number of *bits* in a transmission. n is constant as genome broadcasts only vary in content, not length. Values for $E_{\text{tx}}(r_{\max}) = 0.075$ and $E_{\text{tx}}(r_{\min}) = 0.028$ have been chosen following limited empirical testing. The energy required for receiving is constant and 7% higher than the $E_{\text{tx}}(r_{\max})$, due to the low-power nature of the signals which requires signal reconstruction circuits [7].

In order to evaluate the dependence on energy we amend the experimental setup as follows: maximum energy per robot adjusted from 2000 to 15000, initial energy of a robot lowered to 750 (enough to survive half a generation) and energy pucks limited to 75. Further, the algorithm was adapted to prevent robots from broadcasting in empty neighbourhoods to prevent fruitless genome distribution attempts.

3. EXPERIMENTS

Experiments were designed to evaluate the following hypothesis: When accounting for the cost of communication, biasing the spread of gnomes in mEDEA$_{rf}$ outperforms continuous broadcast combined with an explicit selection in

[1]TI CC2420 in [3] and Bluetooth module LMX9820a in [6]

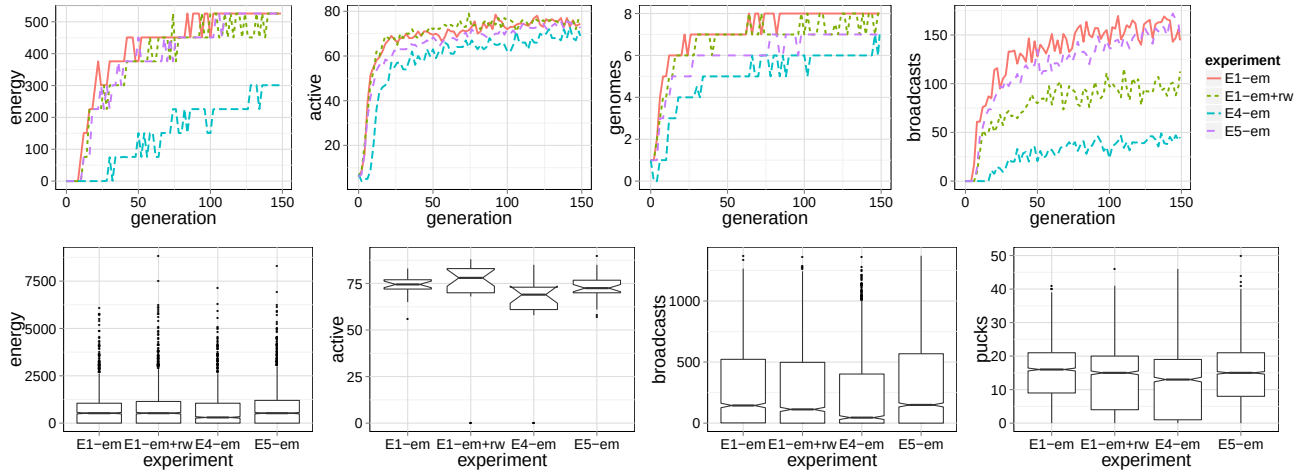


Figure 1: Figures show the energy, number of active robots and genomes received for each of the experiments $\mathbf{E1}_{em}$, $\mathbf{E1}_{em}$+rw, $\mathbf{E4}_{em}$ and $\mathbf{E5}_{em}$. Box plots showing values at generation 150.

terms of active robots and the maintained energy level at the end of the last generation.

Four different experiments were conducted based on the experiments in [4]: $\mathbf{E1}_{em}$: baseline experiment, vanilla mEDEA using the energy model with random individual selection; $\mathbf{E1}_{em}$**+rw**: as $\mathrm{E1}_{em}$, but using roulette-wheel selection as individual selection method; $\mathbf{E4}_{em}$: using a fitness proportionate probability to broadcast; $\mathbf{E5}_{em}$: varying the broadcast radius proportionate to the fitness.

4. RESULTS AND ANALYSIS

Figure 1 shows the median[2] result over 30 repeated runs at the end of the generation.

In $\mathrm{E1}_{em}$ energy can be maintained by reducing broadcasting or gathering energy pucks. In cases where the broadcasting rate is fixed, avoiding others is the only option, which, however, is not conducive to spreading the genome.

The results[3] show that:

$\mathrm{E1}_{em}$ broadcasts most and gathers the most pucks. At the other extreme $\mathrm{E4}_{em}$ (frequency variation) broadcasts least and also gathers fewest pucks: it experiences the least pressure to collect as it can maintain energy by reducing broadcasting. However, the reduced environmental pressure leads to a much slower increase in energy and fewer active robots compared to the unbiased broadcast experiments.

In $\mathrm{E5}_{em}$, although there is the same amount of broadcasting as in $\mathrm{E1}_{em}$, it does not lead to a significant difference in active robots c.f. $\mathrm{E4}_{em}$. The evolved behaviour leads to significantly fewer pucks being collected.

The mechanisms in $\mathrm{E4}_{em}$ and $\mathrm{E5}_{em}$ lower the environmental pressure by preventing less fit individuals from communicating. High fitness individuals will broadcast with high probability or full range hence bear high costs; in contrast low fitness robots rarely broadcast thus save energy. The two methods differ in that in $\mathrm{E5}_{em}$ even with $r = 0$ according to eq. 1 there is still a basic cost.

[2]as a Shapiro-Wilk test showed that the results were not normally distributed

[3]A Wilcoxon Rank-Sum test with a significance level $\alpha = 0.05$ was used to determine statistical significance.

5. CONCLUSION

We introduced an energy model based on the Free-Space model to account for the cost of communication in mEDEA_{rf}. Experimental results showed this exerts environmental pressure to implicitly select for genomes that maintain high energy levels. Comparing the method of varying the broadcasting based on fitness to mEDEA alone and with roulette-wheel genome selection shows that although marginal, the latter approaches outperform the mEDEA_{rf} methods, as they partially reverse the effect of the environmental pressure.

6. REFERENCES

[1] N. Bredeche and J.-M. Montanier. Environment-driven Open-ended Evolution with a Population of Autonomous Robots. In *Evolving Physical Systems Workshop*, East Lansing, United States, 2012.

[2] N. Bredeche, J.-M. Montanier, B. Weel, and E. Haasdijk. Roborobo! a Fast Robot Simulator for Swarm and Collective Robotics. *CoRR*, abs/1304.2, Apr. 2013.

[3] C. M. Cianci, X. Raemy, J. Pugh, and A. Martinoli. Communication in a Swarm of Miniature Robots: The e-Puck as an Educational Tool for Swarm Robotics. In *Swarm Robotics*, pages 103–115, 2007.

[4] E. Hart, A. Steyven, and B. Paechter. Improving Survivability in Environment-driven Distributed Evolutionary Algorithms through Explicit Relative Fitness and Fitness Proportionate Communication. Accepted at GECCO 2015, Madrid, 2015. ACM.

[5] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000*, pages 1–10, 2000.

[6] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. M. Cianci, A. Klaptocz, J.-C. Zufferey, D. Floreano, and A. Martinoli. The e-puck, a Robot Designed for Education in Engineering. In *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, pages 59–65, Portugal, 2009.

[7] Texas Instruments Inc. TI cc2420 Datasheet, 2013.

A.3 Understanding Environmental Influence in an Open-Ended Evolutionary Algorithm

Steyven, A., Hart, E., & Paechter, B. (2016). Understanding Environmental Influence in an Open-Ended Evolutionary Algorithm. In J. Handl, E. Hart, P. R. Lewis, M. López-Ibáñez, G. Ochoa, & B. Paechter (Eds.), *Parallel Problem Solving from Nature – PPSN XIV* (pp. 921–931). Springer International Publishing AG.

Understanding Environmental Influence in an Open-Ended Evolutionary Algorithm

Andreas Steyven**, Emma Hart, and Ben Paechter

School of Computing, Edinburgh Napier University,
10 Colinton Road, Edinburgh, Scotland, United Kingdom
{a.steyven,e.hart,b.paechter}@napier.ac.uk

Abstract. It is well known that in open-ended evolution, the nature of the environment plays in key role in directing evolution. However, in Evolutionary Robotics, it is often unclear exactly how parameterisation of a given environment might influence the emergence of particular behaviours. We consider environments in which the total amount of energy is parameterised by availability and value, and use surface plots to explore the relationship between those environment parameters and emergent behaviour using a variant of a well-known distributed evolutionary algorithm (mEDEA). Analysis of the resulting landscape show that it is crucial for a researcher to select appropriate parameterisations in order that the environment provides the right balance between facilitating survival and exerting sufficient pressure for new behaviours to emerge. To the best of our knowledge, this is the first time such an analysis has been undertaken.

Keywords: evolutionary robotics; parameter selection; environment-driven evolution; distributed online adaptation

1 Introduction

Due to technological advances in both hardware and software, the vision of sending swarms of robots into unchartered terrains to monitor and map environments is becoming much closer to being realised. This brings significant new challenges for evolutionary robotics, with the need for completely distributed evolutionary algorithms to evolve controllers that enable robots to survive for long-periods of time. The issue of survival is key if robots are to effectively accomplish any kind of task: user-driven tasks cannot even be achieved if the integrity of the swarm is compromised through lack of ability to survive.

A number of recent algorithms tackle this issue, notably mEDEA [1] and its variations e.g. mEDEA$_{rf}$ [7] and MONEE [6,4]. However, the emerging behaviours arising from the interactions of an open-ended evolutionary algorithm with its environment are not well understood, perhaps in part due to the time-consuming experimentation that needs to be done to conduct sweeps of the parameters that define the environment. It is common in optimisation to explore

the relationship between algorithmic parameters and fitness. However, evolutionary robotics adds an additional dimension in that it is not only the algorithms parameters that change but also the *environmental* parameters.

Given that it is the environment that provides the pressure to adapt in a purely open-ended scenario, it is crucial to gain some understanding of these landscapes. Particularly in simulation, it is easy to arbitrarily select environmental parameters such as the number of available energy sources or their corresponding energy-values. However, arbitrary choices can inadvertently create landscapes which have a major influence on the evolution of behaviour. For example, assume a researcher wishes to investigate whether individual learning speeds up environment-driven evolution: if an environment is created that has too much energy available then it is unlikely to exert sufficient pressure for individual learning to be beneficial or even emerge. Quantifying 'too much' (or 'too little') is of course difficult. In order to address this, we conduct an analysis of an open-ended evolution algorithm operating in a variable environment. To the best of our knowledge, this is the first time this has been attempted.

Using an open-ended evolutionary algorithm, mEDEA$_{rf}$ [7], we consider evolved behaviours in environments in which the total energy available is parameterised by two variables that determine the availability and value of energy pellets within in the environment. Using a 3-dimensional visualisation of the energy landscape for mEDEA$_{rf}$ we show:

- the energy landscape contains three distinct regions: energy-poor, energy-neutral and energy-rich, as well as a 'dead-zone' in which robots cannot survive
- the energy-rich region is relatively large compared to other regions but is very rugged
- that on the energy-neutral line, distinct behaviours evolve at different places along the line

We propose that the energy-neutral region provides the most obvious settings for conducting experimentation that aims to extend a robots ability to survive or accomplish tasks.

2 Related Work

The completely distributed evolutionary algorithm for open-ended evolution *mEDEA* was first proposed in [1]. It was tested using a scenario in which environmental pressure forces robots to compete for limited resources in order to gain energy. The algorithm was demonstrated to be both efficient with regard to providing distributed evolutionary adaptation in unknown environments, and robust to unpredicted changes in the environment. The basic algorithm has been extended in a number of ways.

Haasdijk *et al* [6] extended mEDEA so that in addition to surviving and operating reliably in an environment, a robot could also perform user-defined tasks. Their new framework MONEE (Multi-Objective aNd open-Ended Evolution algorithm) showed initially that task-driven behaviour can be promoted without

compromising environmental adaptation. More recently, they investigated the trade-off between the survival and task-accomplishment that evolution must establish when the task is detrimental to survival, finding that task-based selection exerts a higher pressure than the environment. Fernandez *et al* [3] study the impact of adding explicit selection methods to the mEDEA algorithm in a task-driven scenario. They evaluate four selection methods that induce different intensities of selection pressure, using tasks that include obstacle avoidance and foraging, finding that higher selection pressure results in improved performances, especially in more challenging tasks. Hart [7] also extended mEDEA by including selection based on a fitness value that was calculated relative to those robots in the immediate vicinity, thus maintaining the decentralised nature of the algorithm, and additionally using this relative fitness value to control the frequency and range of broadcasting. Parameter tuning of *algorithmic* parameters to optimise algorithmic task-performance was investigated by [5]. However, to the best of our knowledge, no methodical investigation of *environment* parameter settings has been conducted: researchers tend to select arbitrary values or simply use those defined in previous papers.

3 Algorithm Description

Evolution of robot controllers is performed by the mEDEA$_{rf}$, first introduced in [7]. The algorithm is an extension of the original mEDEA algorithm of Bredeche *et al* [1] with the addition of an explicit fitness measure. This influences the spread of genomes through the population in order to increase survivability, thus ensuring the integrity of the swarm.

mEDEA$_{rf}$ utilises an agent driven by a control architecture whose parameters are defined by the currently active genome. The genome defines the weights of an Elman recurrent neural network (RNN) consisting of 16 sensory inputs, one bias node (feeding into the hidden layer) and 2 motor outputs (translational and rotational speeds). 8 ray-sensors are distributed around the robot's body. They detect the proximity to the nearest object and its type. The RNN has 1 hidden layer with 16 nodes, thus 322 weights are defined by the genome. This setup is adapted from [1]. An overview of the algorithm is given in Algorithm 1 and reader is referred to [7] for more detail. In brief, for a fixed period, robots move according to their control algorithm, broadcasting their genome that is received and stored by any robot within range. At the end of this period, a robot uses roulette-wheel selection to choose a genome from its list of collected genomes according to a relative fitness value, and applies a variation operator. This takes the form of a Gaussian random mutation operator, inspired from Evolution Strategies. Robots that have not collected any genomes temporarily become inactive, thus reducing the population size.

Each robot estimates its fitness in terms of its ability to survive based on the balance between energy lost and energy gained, delta Energy (δ_E): this term is initialised to 0 at $t = 0$ (when the current genome was activated) and is decreased by 1 at each time-step, and increased by E_{token} if it crosses an energy token. Given δ_E, a robot calculates a fitness value which is relative to those robots in

a range r according to equation 1, where f_i' is the relative fitness of robot i at time t, $mean_{sub_i}$ is the mean δ_E of the robots within the subpopulation defined by all robots in range r of robot i, and sd_{sub_i} is the standard deviation of the δ_E of the subpopulation.

$$f_i'(t) = \frac{\delta_i(t) - mean_{sub_i}(t)}{sd_{sub_i}(t)} \tag{1}$$

Note that evolution is asynchronous, in keeping with the paradigm of a distributed algorithm without central control. If a robot runs out of energy and has an empty genome list, it remains stationary until it receives a new genome from a passing robot at which point it starts a new lifetime. Thus at any time-step, each robot potentially has a different 'age'.

```
genome.randomInitialise();
agent.load(genome);
while forever do
    if genome.isNotEmpty() then
        while lifetime < maxLifetime and energy > 0 do
            agent.move();
            if neighbourhood.isNotEmpty() then
                rf = agent.calculateRelativeFitness(neighbourhood);   // eq. 1
                broadcast(genome,rf);
            end
        end
        genome.empty();
    end
    if genomeList.size() > 0 then
        genome = applyVariation(select_{rhoulette-wheel}(genomeList));
        agent.load(genome);
        genomeList.empty();
    end
end
```

Algorithm 1: Pseudo code of our adapted version of the mEDEA algorithm based on vanilla mEDEA by Bredeche *et al.* [1]

4 Method

All experiments are conducted in simulation using Roborobo! by Bredeche et al. from [2]. A static environment is created, using an arena previously described in [3,6,4]. The robot cannot pass through the outer and inner walls, however, it is possible to broadcast through an obstacle. Energy *tokens* are randomly scattered in the environment. If a robot moves over a token, its energy is increased by an amount E_{token}. The energy token disappears when consumed and reappears after a fixed amount of time later at a different random location. Fixed parameters describing the simulation are given in table 1.

Energy is consumed in three ways. There is a fixed cost to 'living' of 0.5 units per timestep, regardless of whether the robot moves or not. A robot moving consumes an amount of energy E_m that is related to its rotational speed v_{rot}, translational speed v_{trans}, and their respective maximum values $v_{\text{rot}_{\text{MAX}}}$ and $v_{\text{trans}_{\text{MAX}}}$, and is given by

$$E_m = (v_{\text{rot}}/v_{\text{rot}_{\text{MAX}}} + v_{\text{trans}}/v_{\text{trans}_{\text{MAX}}})/4 \quad (2)$$

Finally, a robot consumes energy when communicating. This is an important factor in the real-word but one that it is often overlooked in simulation models. The model used is exactly as described in [10], with an energy cost of $E_{RX} = 0.082$ units for receiving and a cost of $E_{TX}(r) = 0.075$ units for transmitting.

The goal of the experiments is to understand the energy landscape in terms of the median δEnergy of a robot in the population as a function of the two environmental parameters: *count*, the number of energy tokens available, and *value*, the energy value of each token. Table 1 shows the ranges of values considered for each parameter. Parameters are set before the beginning of the experiment and remain fixed throughout. Each experiments was repeated for 5 independent runs. This number is rather low for a noisy application of this type but was chosen to speed up computation due to the high number of experiments that had to be run in total.

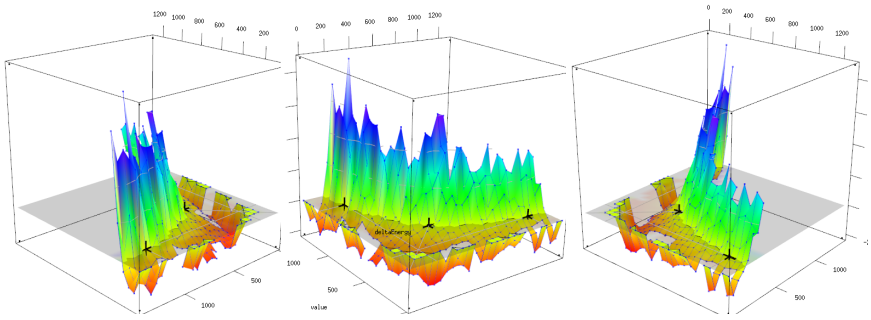
Table 1: Simulation and Experimental Parameters for all experiments

Simulation parameters	
Arena size	1024 pixel by 1024 pixel
Max. robot lifetime	2500 iterations
Token re-spawn time	500 iterations
Sensor range	196 pixel
Variable Parameters	
Number of robots	50, 75, 100
Number of tokens (*count*)	0 - 1300 (in steps of 50)
Energy value per token (*value*)	0 - 1400 (in steps of 50)
Experimental parameters	
Number of runs	5
Maximum iterations	375000 (= 150x2500)
Start energy	500
Maximum range r_{max}	128

Data is gathered from the robots every 2500 iterations. Recall from section 3 that each robot chooses a new genome once it has depleted all its energy or reached the maximum lifetime, leading to asynchronous generation changes throughout the population. Hence, the data gathered at each interval represents a snapshot across robots of multiple ages and therefore does not necessarily capture the peak performance of each robot (i.e. it may include very 'young' robots). However, given that the goal of the experiment is to understand the interplay of the specific algorithm and environment under consideration, this is not a relevant factor.

5 Analysis

Figure 1 shows three rotated 3-dimensional plots of surface obtained using 100 robots after 375,000 iterations. The x and y axes represent the *count* and *value* variables, while the z axis represent the median δ_E of the robot population over the last 2500 iterations. The grey plane marks a value for δ_E of zero, at which point robots have an energy balance of zero, i.e. the same amount of energy as they started the experiment with. Three broad regions are noticeable: a large region in which the robots have positive δ_E (green and blue value above the grey plane), a region lying on the plane itself, and finally a region below the plane in which robots are spending more energy than they are collecting, i.e. $\delta_E < 0$. In order to explore this in more detail, a 2-dimensional top-down projection is shown in figure 2 obtained from populations of 50, 75 and 100 robots, and is discussed in detail below.



(a) rotated 90° right (b) centred (c) rotated 90° left

Fig. 1: View on the resulting surface from different angles. The figure was created by plotting the median δ_E of the last 2500 iterations of the experiment. The grey plane marks a value for δ_E of zero, at which point robots in an experiment have an energy balance of zero. In other words, the same amount of energy as they started the experiment with. A 3D model can be found at [9]

5.1 Different performance regions

Figure 2 shows clearly that the landscape is defined by four different regions:

A) Dead Zone: In this region, the environment does not provide enough energy for the algorithm to evolve controller that can survive a full run. Low values for both parameters, *count* and *value* result in the extinction of the whole robot population within a few generations. The random genomes that the controllers are initialised with generally result in a random spinning behaviour, rather than movement. This random behaviour, combined with the lack of energy tokens in the immediate vicinity in which the robot is born, mean that robots cannot survive given its inability to move.

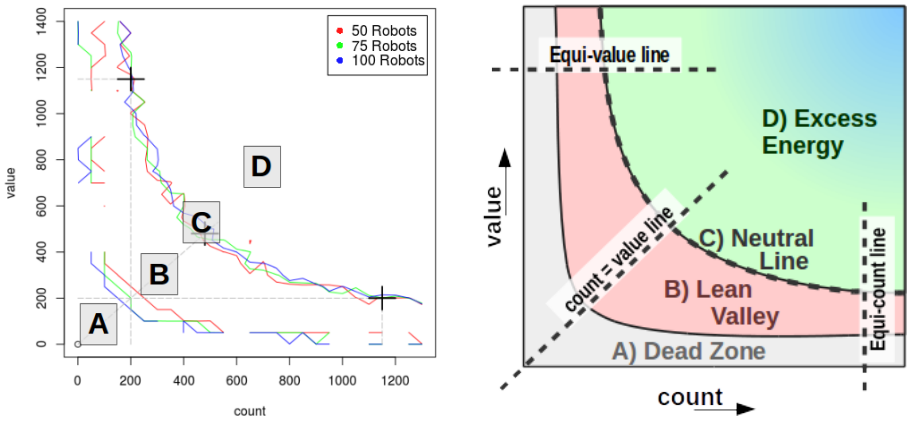


Fig. 2: Overview of landscape, as plot of the real data on the left and as a cartoon version on the right. 4 different regions are shown: A) Dead Zone, B) Lean Valley, C) Neutral Line, D) Excess Energy.

B) Lean Valley (negative δ_E): This region starts at the edge of the dead zone that marks the point where there is just enough energy available that some robots survive until the end of the experiment, i.e. it marks the point where a robot has spent all its initial energy and started picking up tokens from the environment. Moving down towards the bottom of the valley, an increasing number of robots survive as there is more energy in environment, with the corollary that each robot has less total energy — the energy available is shared between more robots. The bottom of the valley marks the minimum δ_E that still enables survival. Moving upwards out of the valley on the other side, robots gradually get better in both harvesting energy from the environment and managing their residual energy as a result of evolving better strategies. For example, good strategies optimise movement, or avoid moving towards tokens in which there are other robots close by.

C) Neutral Line (δ_E=0): This line marks the points in the environment where the environment provides exactly enough energy to enable a robot to maintain an energy balance of zero, i.e. the costs of moving and communicating are just balanced by energy harvested.

D) Excess Energy ($\delta_E > 0$): In the final region, in which both *cost* and *value* are high, robots are able to locate more energy in the environment than is required to maintain their initial energy E_0, either due to the abundance of pucks or the high energy value of pucks.

5.2 Environmental Influence on Behaviour

In order to properly understand the evolved behaviours that lead to the landscapes just described, a more detailed analysis is required. Figure 3 examines

pairings of $(count, value)$ along the three dashed lines in 2, i.e. equivalent-*value* (**a-b**), equivalent-*count* (**c-d**) and the diagonal in which $count = value$ line (**e-f**). The figure shows boxplots of the δ_E values at specific pairings of $(count, value)$ and the ratio of genome broadcasts made to unique genomes received over a lifetime. The latter quantity leads to insights into behaviour as it relates to the number of *unique* robots encountered by an individual robot: a robot will broadcast indiscriminately to any robot in its range but will only collect unique genomes. At the equivalent-count and equivalent-value lines, we fix the parameter *count* and *value* respectively, and successively increase the other parameter in steps of 50.

5 points are shown. The first point on a) corresponds to a total energy E_{tot} that is the same as the first points on graphs (c) and (e) below it etc.[1]. For a specific value of E_{tot}, then is clear that high *value* combined with low *count* leads to robots that have increased δ_E when compared to robots with high *count* but low *value* (graph (a) compared to graph (e)). Robots must therefore evolve behaviours that enable them to seek out the rare but high-value pucks. These robots also have high broadcast:genome ratios, suggesting the robots are frequently coming into contact with the *same* robots. A possible explanation lies in the fact that the robots appear travel in small groups, thus broadcasting continually to the same robots; the rare occurrence of pucks leads to many robots having to travel towards the same regions of the space. On the other hand, a high *count* leads to robots that receive more unique genomes than in the high *value* case: this is suggestive of a more random movement pattern that enables each robot to encounter many unique robots during its lifetime. In this case there is low selection pressure to evolve focused movement due to the abundance of pucks.

5.3 Behaviours in the neutral region

We propose that the energy neutral region is of greatest interest for researchers wishing to conduct research moving beyond genetic evolution of survival, for example using individual or social learning [8] or task-driven research [4]. In this region, on the one hand, robots are able to survive, while on the other, the environment does not *over*-provide, thus ensuring that there is scope for robots to learn novel behaviours. We further investigate three specific points within this region there is approximately the *same* amount of energy available in the environment (table 2). The table shows the median age increases with increasing *count* — it is easier to maintain sufficient energy to survive as availability increases. The lower median observed at low *count* reflects the fact that many robots do not survive long. The time to find a new unique genome (age:genome) is shortest at high *count*, reflecting frequent encounters with novel robots. Broadcast:genomes is highest at low *count* as observed in the previous section. All three configurations lead to the same energy balance of 0, but diverse behaviours result in the gain in energy being offset by movement and broadcasting in each case.

[1] while this is exactly true for the first and third rows, in the middle row which represents equal count/value it is necessary to approximate

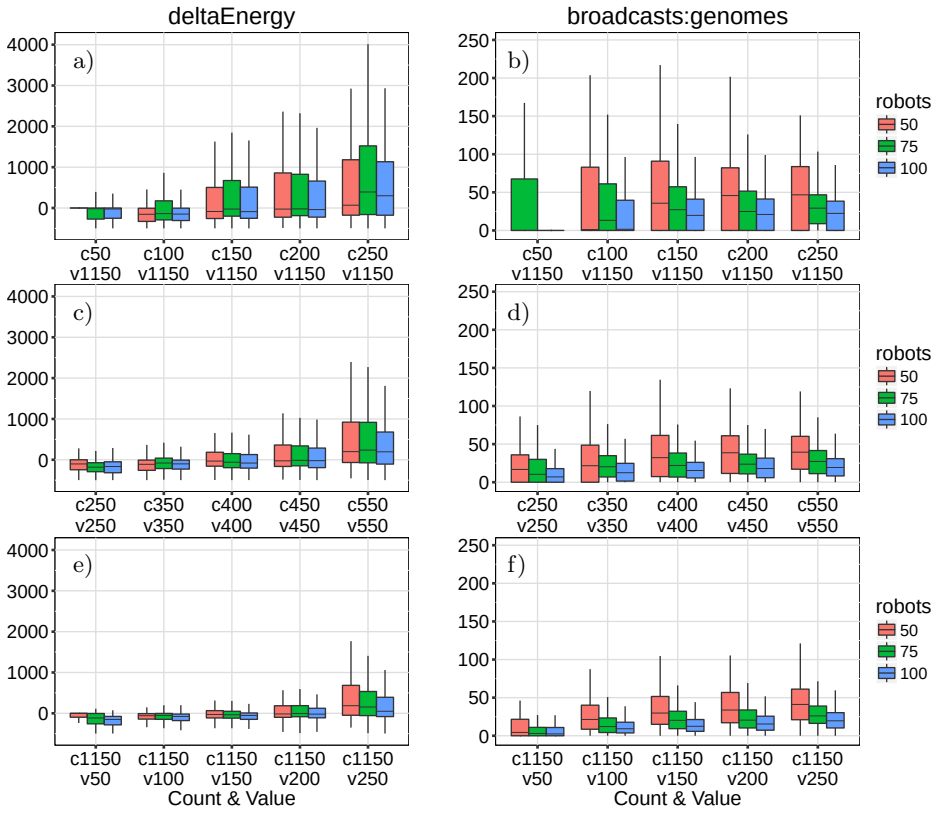


Fig. 3: Cuts through different parts of the landscape. Points towards different behaviours in terms of exploration. a-b) *value* = 1150, vary *count*; c-d) *count* = *value*; e-f) *count* =1150, vary *value*.

6 Conclusion

We have presented the first analysis of the fitness landscape (as a function of environmental parameter) that results from running an open-ended evolutionary algorithm (mEDEA$_r f$) in an environment that is parameterised by two values that control the distribution of energy in the environment. Adjusting the availability and value of energy pucks results in the evolution of a range of different behaviours. Rather than arbitrarily selecting parameters in which to study evolution, we suggest that it is vital to understand *how* these choices will direct evolution, by changing the selection pressure exerted by the environment.

Three distinct regions are observed in which the final energy balance can be negative, neutral, or positive. A fourth region is found in which robots cannot survive. We propose that the energy neutral region is a good region in which to undertake experiments. It provides an environment in which robots are able to survive, enabling experimentation, while at the same time, will reward new behaviours which are able to more efficiently harness energy from the environ-

Table 2: Results obtained at three configurations within the neutral region

		Robots								
		50			75			100		
Count	Value	Age	Age / Genome	Brodcasts / Genome	Age	Age / Genome	Brodcasts / Genome	Age	Age / Genome	Brodcasts / Genome
200	1150	*770*	*107.94*	*46.61*	*767.5*	*63.86*	*28.99*	*667*	*49.97*	*21.18*
500	500	*1026.5*	*86.12*	*39.11*	*1038.5*	*52.39*	*25.93*	*928.5*	*41.26*	*19.67*
1150	200	*1173.5*	*79.83*	*33.43*	*1093*	*47.39*	*21.95*	*1059*	*36.52*	*15.49*

ment. It is clear that the environment plays a key role in influencing what kind of behaviours emerge, in that it is not the total amount of energy available that matters but also the manner in which it is spread. Future work should be aimed at understanding the landscape in more detail, and in particular, explaining the ruggedness of some regions.

References

1. Bredeche, N., Montanier, J.M.: Environment-driven embodied evolution in a population of autonomous agents. In: Schaefer, R., Cotta, C., Koodziej, J., Rudolph, G. (eds.) PPSN XI. vol. 6239, pp. 290–299. Springer Berlin Heidelberg (2010)
2. Bredeche, N., Montanier, J.M., Weel, B., Haasdijk, E.: Roborobo! a fast robot simulator for swarm and collective robotics. CoRR abs/1304.2 (4 2013)
3. Fernández Pérez, I., Boumaza, A., Charpillet, F.: Comparison of selection methods in on-line distributed evolutionary robotics. In: ALife'14. pp. 282–289. MIT Press (2014)
4. Haasdijk, E.: Combining conflicting environmental and task requirements in evolutionary robotics. In: 2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems. pp. 131–137. IEEE (9 2015)
5. Haasdijk, E., Smit, S.K., Eiben, A.E.: Exploratory analysis of an on-line evolutionary algorithm in simulated robots. Evolutionary Intelligence 5(4), 213–230 (2012)
6. Haasdijk, E., Weel, B., Eiben, A.E.: Right on the MONEE. In: Blum, C. (ed.) Proceedings of GECCO '13. pp. 207–214. ACM Press (2013)
7. Hart, E., Steyven, A., Paechter, B.: Improving survivability in environment-driven distributed evolutionary algorithms through explicit relative fitness and fitness proportionate communication. In: Silva, S. (ed.) Proceedings of GECCO '15. pp. 169–176. ACM Press (2015)
8. Heinerman, J., Rango, M., Eiben, A.E.: Evolution, individual learning, and social learning in a swarm of real robots. In: 2015 IEEE Symposium Series on Computational Intelligence. pp. 1055–1062. IEEE (2015)
9. Steyven, A.: Interactive 3D model of mEDEA_rf parameter sweep fitness landscape (2016), `http://research.steyven.de/conf/ppsn2016/`
10. Steyven, A., Hart, E., Paechter, B.: The cost of communication. In: Silva, S. (ed.) GECCO Companion '15. pp. 1239–1240. ACM Press (2015)

A.4 An Investigation of Environmental Influence on the Benefits of Adaptation Mechanisms in Evolutionary Swarm Robotics

Steyven, A., Hart, E., & Paechter, B. (2017). An Investigation of Environmental Influence on the Benefits of Adaptation Mechanisms in Evolutionary Swarm Robotics. *Proceedings of the 2017 on Genetic and Evolutionary Computation Conference - GECCO '17* (8 pages). New York, New York, USA: ACM Press.

An Investigation of Environmental Influence on the Benefits of Adaptation Mechanisms in Evolutionary Swarm Robotics

Andreas Steyven
Edinburgh Napier University
10 Colinton Road
Edinburgh, Scotland, UK
a.steyven@napier.ac.uk

Emma Hart
Edinburgh Napier University
10 Colinton Road
Edinburgh, Scotland, UK
e.hart@napier.ac.uk

Ben Paechter
Edinburgh Napier University
10 Colinton Road
Edinburgh, Scotland, UK
b.paechter@napier.ac.uk

ABSTRACT

A robotic swarm that is required to operate for long periods in a potentially unknown environment can use both evolution and individual learning methods in order to adapt. However, the role played by the environment in influencing the effectiveness of each type of learning is not well understood. In this paper, we address this question by analysing the performance of a swarm in a range of simulated, dynamic environments where a distributed evolutionary algorithm for evolving a controller is augmented with a number of different individual learning mechanisms. The learning mechanisms themselves are defined by parameters which can be either fixed or inherited. We conduct experiments in a range of dynamic environments whose characteristics are varied so as to present different opportunities for learning. Results enable us to map environmental characteristics to the most effective learning algorithm.

CCS CONCEPTS

•**Computing methodologies** → **Mobile agents;**

KEYWORDS

Evolutionary Swarm Robotics, Environment, Learning

ACM Reference format:
Andreas Steyven, Emma Hart, and Ben Paechter. 2017. An Investigation of Environmental Influence on the Benefits of Adaptation Mechanisms in Evolutionary Swarm Robotics . In *Proceedings of GECCO '17, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3071178.3071232

1 INTRODUCTION

Recent advances in technology are driving novel research in swarm robotics, envisioning future applications in which swarms might be sent to remote or hazardous environments and in which they will need to survive over long periods of time. As these environments will be unknown to the designer *a priori* and are potentially dynamic, the swarm must be able to continuously adapt its behaviour to ensure it both maintains sufficient energy to survive, and to successfully perform tasks.

The importance of being able to adapt over time has been a subject of research within Evolutionary Robotics for some time [20]. Adaptation often takes one or all of three forms: evolutionary, individual and social learning. In *evolutionary* adaptation, information encoded on the genome adapts through selection and reproductive operators over many generations. In *individual* learning, a robot can adapt its own behaviour during the course of its lifetime, for example, updating weight values in a neural network controller. Finally in *social* learning, robots can exchange information during a lifetime.

The relative benefits of mixing the different types of adaptation have been studied both in simulation [3, 6] and hardware [4, 10–12]. Typically, experiments are conducted in single environment related to a specific task, therefore the role of the environment in influencing the result is not made explicit. An exception is recent work from Haasdjik [5] who explicitly studied the effect of combining conflicting environmental and task requirements in a simulated system. This showed that high selective pressure exerted by a task can outweigh any selective pressure from the environment. However, an arbitrary environment was defined to conduct experiments in, leaving open the question of whether the same effects would be observed in a different environment.

The goal of this paper is to investigate the interplay between evolution, individual learning and environment characteristics. We consider a swarm which undergoes distributed evolution of a neural-network based controller and is augmented with an individual learning mechanism: this modifies the information gleaned from the environment and fed to the controller over the lifetime of a robot. Specifically, we consider a swarm operating in an environment which is unknown *a priori* and which robots must learn relative values of positive and negative energy tokens. Each environment contains n positive and n negative energy tokens. Positive tokens increase the robot's energy by v units of energy, while negative ones reduce it by a fixed amount. As n, v vary, each environment presents different opportunities for learning in that there are a small number of high value tokens, or a large number of low value tokens. In addition, tokens change their nature across 'seasons', i.e. tokens of a specific colour switch value from negative to positive on a cyclical basis. This forces the swarm to have to re-learn the effect of any given colour of token every season. Various settings for individual learning

GECCO '17, Berlin, Germany

are investigated in which the learning mechanism is either fixed or has components that can be simultaneously evolved. The following questions are investigated:

- How do the parameters of the environment (token count, token value) influence the effectiveness of different individual learning settings?
- How does the rate of change of a given environment influence the effectiveness of individual learning mechanisms?
- How does the nature of the individual learning mechanism influence performance in different environments?

We augment a distributed evolutionary algorithm previously described in [9] with mechanisms for individual learning in order to conduct experiments. Note that the goal is not to propose a novel method of either individual learning or evolutionary adaptation but to explore the relationship between the environment and value of different types of adaptation.

2 RELATED WORK

A reasonable body of research exists in relation to combining learning and evolution, and factors that influence this relationship [7, 13, 14]. The relationship of the two methods in a swarm environment in which it is necessary to simultaneously learn behaviours which enable reproduction in addition to task performance is less well studied however. Haasdjik *et al* propose a framework for evolution, individual and social learning in collective systems, and consider the interaction of evolution and individual learning in which the latter is achieved by *reinforcement learning* [19]. Their experiments show that in a collective system, it is possible for learning to counteract evolution. A *hiding-effect* can occur in which individual learning acts to mask the ill-adapted nature of non-optimal agents and is therefore counter-productive. Although a number of environments were investigated which essentially modified the reward system, all environments were static, and the relationship of the learning framework to specific parameterisations of the environmental features was not examined.

A dynamically changing reward system was investigated in [1] who proposed mEDEA, a completely distributed evolutionary algorithm for open-ended evolution. Here, efficient adaptation in a changing environment was demonstrated using a set up that switched phases: in the *free-ride* phase, there is no cost to movement therefore a robot only needs to meet a single other robot to pass on its genome, while in the alternating phase the robot is required to harvest energy in order to move and therefore creating opportunties for passing on its genome. Haasdijk *et al* [8] extended mEDEA to add explicit task-selection in the MONEE framework [15]. In [5] they examine in more detail the relative selection pressures induced by task performance and survival in different environments, finding that task performance is optimised even if it reduces the lifetime of robots (and therefore their ability to reproduce). Heinermann *et al* investigate the relationship between evolution, individual and social learning in real swarm [10–12]. Here, the evolutionary part focuses on evolving a suitable sensory layout, while the individual learning runs an evolution strategy to learn the network weights during the robot lifetime. Learnt weight vectors are broadcast to other robots during the social learning phase. The main focus of this work was to investigate the impact of social learning. Individual learning is *required* to learn a controller and hence cannot be omitted.

In contrast to the above, we consider scenarios in which individual learning has the potential to improve evolved behaviours, but is not essential. We investigate the relative benefits of evolution and individual learning using a variety of learning mechanisms and in a range of environments with different features. The goal is to specifically relate the roles of evolution and individual learning performance to features of the environment.

3 OVERVIEW

A swarm operates in an open environment in which there are two types of coloured tokens: driving over one colour increases the robots energy while the other decreases it. Robots should learn to avoid the negative token. However, a "seasonal" change is imposed where the value of the token is reversed, i.e. red becomes positive and blue negative or vice versa. A robot must thus adapt any previously evolved behaviour. All robots in the swarm evolve a neural network that controls their behaviour through a distributed evolutionary algorithm [9] In addition, they can exploit an individual learning mechanism which can potentially learn the *current* value of a given colour of token. This information modifies an input to the evolved neural network. We investigate a number of types of individual learning in which some components of the learning mechanism can be either heritable, fixed or absent.

Experiments are conducted using the Roborobo simulator [2]. The robots have 8 ray-sensors distributed around the body and detect proximity to the nearest object and its type. Each robot is controlled by an evolved Elman recurrent neural network (RNN). The network has 16 sensory inputs and 2 motor outputs (translational and rotational speeds). The 16 inputs comprise of two information of each of the 8 ray-sensors, proximity and whether or not this object is an energy token. Although the colour/type of the object is also detected by the robot, it is not fed into the RNN as an input, but only used in the adaptation mechanism[1].

3.1 mEDEA

Using the inputs and outputs just described, an RNN with 1 hidden layer containing 16 nodes is evolved by a distributed evolutionary algorithm [9]. This algorithm is an extension of $mEDEA$ [1], and incorporates a selection mechanism based on relative fitness. In brief, for a fixed period, robots move according to their control algorithm, broadcasting their genome that is received and stored by any robot within range. At the end of this period, a robot uses fitness-proportionate

[1] the information cannot be encoded directly to the network without *a priori* knowledge of the number of potential colours

selection to choose a genome from its list of collected genomes according to a relative fitness value, and applies a variation operator. This takes the form of a Gaussian random mutation operator, inspired from Evolution Strategies. Pseudo-code is given in Algorithm 1.

```
load(currentGenome = randomInitialisedGenome);
while iteration ≤ maxIterations do
    if hasGenome() then
        if lifetime ≤ maxLifetime & energy > 0 then
            move();
            if neighbourhood.isNotEmpty() then
                rf =calculateRelativeFitness();  // eq.1
                broadcast(currentGenome,rf);
            end
        else
            remove(currentGenome);
        end
    end
    genomeList.addIfUnique(receivedGenomes);
    if genomeList.size() > 0 then
        genome = select_{roulette-wheel}(genomeList);
        load(currentGenome =
          applyVariation(genome));
        genomeList.empty();
        lifetime = 0;
    end
end
```

Algorithm 1: Pseudo code of the adapted version of the mEDEA algorithm with relative fitness mEDEA$_{rf}$ as introduced in [9] used with roulette-wheel as explicit selection mechanism

Each robot estimates its fitness in terms of its ability to survive based on the balance between energy lost and energy gained, denoted (δ_E): this term is initialised to 0 at $t = 0$ (when the current genome was activated) and is decreased according an energy-model described below that accounts for both movement and the cost of communicating for evolution, and increased by E_{token} if it crosses an energy token. Given δ_E, a robot calculates a fitness value which is relative to those robots in the neighbourhood of range r. according to equation 1, where f'_i is the relative fitness of robot i at time t, $mean_{sub_i}$ is the mean δ_E of the robots within the subpopulation defined by all robots in range r of robot i, and sd_{sub_i} is the standard deviation of the δ_E of the subpopulation.

$$f'_i(t) = \frac{\delta_i(t) - mean_{sub_i}(t)}{sd_{sub_i}(t)} \tag{1}$$

There is a fixed cost to living of 0.5 units per timestep, regardless of whether the robot moves or not. A robot moving consumes an amount of energy that is related to its rotational speed v_{rot}, translational speed v_{trans}, and their respective maximum values $v_{\text{rot-max}}$ and $v_{\text{trans-max}}$

$$E_{\text{step}} = 0.5 + \left(\frac{v_{\text{rot}}}{v_{\text{rot-max}}} + \frac{v_{\text{trans}}}{v_{\text{trans-max}}}\right)/4 \tag{2}$$

The amount of energy spent on communication E_{com} is calculated using equation 3, where i and j are the number of genomes received and transmitted respectively. The values $a_{\text{rx}} = 0.0305$, $a_{\text{tx}} = 0.01379$ and $a_{\text{tx-amp}} = 0.000614$ were determined based on the method described by [18]; the reader is referred to this publication for a description of their approach.

$$E_{\text{com}} = \sum_{k=0}^{i} a_{\text{rx}} + \sum_{k=0}^{j} \left(a_{\text{tx}} + b_{\text{tx-amp}} \times d^2\right) \tag{3}$$

Equation 4 shows the change in energy at each simulation step, where n is the number of tokens that have been collected in that step.

$$E(t+1) = E(t) - E_{\text{step}} - E_{\text{com}} + (n_{\text{token}} \times E_{\text{token}}) \tag{4}$$

3.2 Environment

In Evolutionary Robotics, it is often unclear exactly how parameterisation of a given environment might influence the emergence of particular behaviours. Often, the focus of reported studies is on algorithm performance, without serious consideration of how the choice of environment may influence results. This is particularly important for an open-ended distributed algorithm such as *mEDEA* in which survival of robots is crucial for evolution to occur. To counter this, Steyven *et al* [17] recently proposed a technique by which preliminary experimentation could be used to generate a surface-plot, highlighting regions of the parameter space in which the environment provides the right balance between facilitating survival and exerting sufficient pressure for new behaviours to emerge. This enables a researcher to select appropriate settings for experimentation. For example, for a given task, on the one hand, there will be regions in which the characteristics of the environment are such that robots find survival to be trivial (e.g. food supplies are unlimited and easy to find), and hence there is little pressure to evolve specialised behaviours. On the other hand, environmental characteristics which are harsh enough to cause individual robots to die prematurely and therefore prevent any effective evolution are also identified.

Using the algorithm described above, we conducted experiments in an environment parameterised by two variables: the *number* of energy tokens available, and the *value* of the energy token. In each environment tested, there are n positive tokens with value v, and n negative tokens with value -400. The delta-energy δ_E, i.e. difference between start and end energy is recorded for multiple points in the parameter space, resulting in the plot shown in figure 1. From this plot, we identify three points to conduct experiments along the *energy neutral line*, i.e the region in which the robot expends as much energy as it acquires. This represents a region in which selection-pressure from the environment to survive is neither too small or too large to mask the behaviours we are

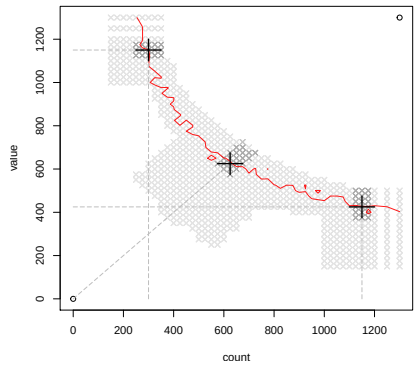


Figure 1: Overview of newly created surface landscape. The red line shows the Neutral Line, the line where the surface plot crosses a plane drawn at delta-energy (δ_E)=0.

*

Table 1: Environmental configurations: description refers to the prevalence of energy tokens within the environment.

Number of tokens	*Value per token*	*Description*
300	1150	Scarce
625	625	Balanced
1150	425	Abundant

interested in investigating. The points identified are specified in table 1.

4 INDIVIDUAL LEARNING

The neural network described above has a set of binary inputs (one for each sensor ray of the robot) that denote the presence (1) or absence (0) of a token (independent of its type). Therefore, in an environment in which there are multiple types of tokens, the only way for an individual to distinguish between them is to pick up the token and observe the change in energy. If the environment in which the robot operates is known *a priori*, then clearly, the neural network could be designed in order to include relevant information about each token type. However, if the environment is unknown, then the robot must learn to adapt to the different types and values of tokens it may encounter.

We use an adaptation mechanism which enables a robot to modify the value input to the RNN corresponding to a token sensor: instead of simply having a binary input, the robot uses a learned/evolved multiplier to adapt the token input to a continuous value between -1 and 1.

Each time a previously unseen type of token is encountered (detected by a sensor ray or through consumption[2]), a new multiplier is added to the multiplier set. As tokens are usually detected before they are consumed, no information regarding a new token's value is known: the robot therefore randomly initialises a value to associate with the type (x) of detected token. Following consumption, the resulting change of energy is detected by the robot and its learning mechanism can modify the corresponding multiplier value (m_x).

All multiplier values are adjusted every time a token is consumed according to equation 5:

$$m'_x = m_x + LS \times \left(LR - \frac{C_x}{C_{\text{total}}}\right) \times \left(\frac{V_x}{V_{\max} - V_{\min}}\right) \quad (5)$$

where m_x is the current value for the multiplier for type x; $C_x$is the number of tokens of type x collected; C_{total} is the total number of all tokens collected; V_x is the value of the token that has just been consumed and is therefore now known to the robot (being equivalent to the change in energy); $V_{\max}, V_{\min}$ define the minimum and maximum values of all tokens encountered so far. LR is a learning rate that controls the magnitude of the change, and LS is either -1 or $+1$ and simply inverts the direction of change; this is required to adjust the learning mechanism to the internal value notation of the neural network and can be adapted via evolution. The learning mechanism is shown in Algorithm 2.

```
if token_x is unknown then
    multipliers.add(token_x);
end
if token_x is consumed then
    tokenCounter_x.update(token_x);
    totalTokenCount.update();
    tokenValue_x.update(δ_E(t) − δ_E(t − 1));
    totalValueRange.update();
    for m_x in multipliers do
        m_x.update();                          // eq. 5
    end
end
```

Algorithm 2: Pseudo code of the steps carried out to update all multipliers every time a token is encountered.

Three factors influence the learning mechanism: the initial value assigned to a token V_x, the learning rate LR and the associated sign LS. These factors can be randomly assigned, fixed to some specific value, or can themselves be subject to evolution. Allowing the learning sign to co-evolve enables the learning mechanism to self-adapt to the internal value convention of the neural network. Finally, enabling the robot to evolve an appropriate starting value for each type of token based on its experience may speed up learning in some circumstances. Even though token values change over seasons, inheriting a good starting value may be beneficial, likely dependent on the rate of change of the environment.

Table 2 defines four variants of the learning algorithm that we investigate in conjunction with the three environments described in section 3.2. Note that in no case is any Lamarkian evolution used, i.e. although the multiplier starting values

[2]The sensor rays of the robot are not evenly distributed around the robot body. This can lead to the situation in which a robot drives over the token before any of the sensor rays detected it.

Table 2: Learning scenarios investigated showing heritability of information

	Initial Value of Multiplier	LR	LS
Baseline	1 (all tokens)	none	n/a
IL	random	fixed	evolved
EVO	evolved	none	n/a
EVO+IL	evolved	evolved	evolved

Table 3: Simulation and Experimental Parameters for all experiments

Simulation parameters	
Arena size	1024 px × 1024 px
Max. robot lifetime	2500 iterations
Token re-spawn time	500 iterations
Sensor range	196 pixel
Max. communication range $r_{\max}$	128 pixel
Experimental parameters	
Number of independent runs	30
Number of robots	100
Max. iterations	100,000
Start energy	500

are adapted over the course of a lifetime, they are *never* written back to the genome and are therefore not inherited.

4.1 Experiments

An experiment is defined by a tuple <*environment, seasonal change rate, algorithm*>. Three environments (see section 3.2) and three different rates of seasonal change are investigated: 0 (no change, i.e. static environment), every 5000 iterations, and every 15000 iterations. Note that the maximum lifetime of a genome before it is replaced is 2500 iterations, so every robot should go through at least one evolutionary generation during the shorter (5000 iterations) season and at least 5 times in the 15000 season. In practice, as robots tend to die before their maximum lifetime, more evolutionary cycles are likely to occur.

Four algorithms are investigated as detailed in table 2. Note that in the baseline experiments, all tokens have a fixed multiplier of 1 and therefore the robots cannot distinguish between tokens of different types. Thus, in total 36 (=3x3x4) experiments are conducted. In each experiment, we record the *totalTokenRatio* at the end of the season. This value is the ratio of the number of collected token with positive value divided by the sum of all collected token within that season. A ratio of 0.5 shows that an equal amount of positive and negative token was collected, below 0.5 more negative and above more positive token, respectively.

Experimental and simulation parameters are given in table 3. Parameters associated with the learning mechanism are given in table 4. The values for LR_{initial} and $LR_{\max}$ where selected following limited empirical exploration.

Table 4: Learning parameter with their initial values and ranges in which they can change during runtime of the experiment.

Parameter	**Init. Value**	**Value Range**
Learning rate, LR	1.02	$[LR_{\min}, LR_{\max}]$
Min. LR, $LR_{\min}$	1	fixed
Max. LR, $LR_{\max}$	1.5	fixed
Multiplier of type x, m_x	random	$[-1, 1]$
Learning sign, LS	random	$[-1, 1]$

The positive value of an energy token is determined by the environment. In seasons when a token is negative, the value is fixed -400 which is 80% of a robot's initial energy.

Following 30 runs of each experiment, statistical analysis was conducted based on the method in [16] using a significance level of 5%. The distributions of two results were checked using a Shapiro-Wilk test. If both followed a Gaussian distribution then Levene's test for homogeneity of variances was perfomed. For equal variances the p-value was determined using an ANOVA test, otherwise using a Welch test. A Kruskal-Wallis rank sum test was perfomed to determine the p-value if one of the results followed a non-Gaussian distribution.

5 RESULTS

This section provides summarised results: detailed experimental data is available as supplementary material. Table 5 shows the median totalTokenRatio for each of three individual learning mechanisms (EVO, EVO+IL, IL) in each of the 3 environments and for each value of seasonal change. The values are compared to the result from the baseline experiment each case, and statistical significance is indicated in the table.

The EVO method (which evolves multiplier values but has no adaption during a lifetime) outperforms the baseline method in all three static environments (season change = 0). Here, evolution is able to determine appropriate values for each multiplier type. However, in the dynamic environment, evolving the multiplier value is detrimental. In the first season, evolution can find appropriate multiplier values (particularly in a long season). However, as soon as the season changes, these become irrelevant; if these values have spread sufficiently through the population it may take considerable time for evolution to reverse this change, while in the meantime, the robot will continue to collect negative tokens.

The IL method (fixed learning rate and random initialisation of values) never outperforms the baseline method in the static environment, and is worse than the baseline in the dynamic environments. The magnitude of the effect is highest in the seasonal change=5000 environment for a balanced environment. It appears that the learning rate is not sufficient to adapt a randomly initialised multiplier to a suitable value while the randomness can actually bias the

Table 5: Showing median of end values by seasonal change and Experiment for *totalTokenRatio* over generation 199 to 200 (N:30). ↓, ↔, ↑ indicate whether the value is lower, not different or higher respectively compared to the baseline experiment. The number of arrows corresponds to the magnitude level of the effect size based on a Vargha and Delaney A test. (1 = small, 2 = medium, 3 = large)

Experiment		**Evo**			**IL**			**Evo + IL**		
	count	**300**	**625**	**1150**	**300**	**625**	**1150**	**300**	**625**	**1150**
Season	**value**	**1150**	**625**	**425**	**1150**	**625**	**425**	**1150**	**625**	**425**
0		↑↑↑ 0.5301	↑↑↑ 0.5411	↑↑↑ 0.5662	↔ 0.5034	↔ 0.5056	↓ 0.4997	↑↑↑ 0.5306	↑↑↑ 0.5388	↑↑↑ 0.5705
5k		↔ 0.4995	↓ 0.4982	↓ 0.4989	↓ 0.5006	↓↓ 0.4975	↔ 0.5023	↔ 0.5029	↑↑ 0.5134	↑↑↑ 0.5191
15k		↓↓ 0.496	↓↓ 0.495	↓ 0.4981	↓ 0.4973	↓ 0.4993	↓ 0.5011	↔ 0.4981	↑↑ 0.5136	↑↑↑ 0.5121

robot towards collecting a particular type. On average, this is worse than the baseline case in which the robot has equal preference for both types.

In contrast, with the exception of the two *dynamic and scarce* environments, the EVO+IL method which evolves the LR, LS and the multiplier values and also adjusts the latter during lifetime, a significant improvement is observed with respect to the baseline method. In the *scarce* environments, the robots have little information available to them to inform learning, as there are few tokens. When the environment is changing rapidly this is particularly detrimental. In the other environments, there are more tokens to learn from. When this is coupled with the ability to both evolve useful multiplier values *and* adapt them at a appropriate rate, the swarm learns to adapt to the changing environments and improves its behaviour in the static environment.

5.1 Influence of environmental parameters

Next, we examine the first question posed in section 1 in more depth: *under what environmental conditions is augmenting evolution with an individual learning mechanism beneficial?*

Table 6 provides a pairwise comparison of environments for *totalTokenRatio* obtained at the end of each experiment. In this table and subsequent ones, the symbols =, <,> indicate whether the median values for totalTokenRatio are not significantly different, significantly smaller or larger respectively. p-values below the significance level of 0.05 are written in bold.

Table 6 clearly indicates that for the methods that include an evolutionary component with the learning algorithm, then in the static environment, *abundant* > *balanced* > *scarce*. In contrast, when only a fixed individual learning mechanism is used with no adaptation of learning rate, then the reverse appears true; the token ratio is higher in the *balanced* and *scare* environments is higher than in the *abundant* environment, with no significant difference between *balanced* and *abundant*.

In the slow changing environment (15k), the general trend is that *abundant* > *balanced* > *scarce* for *all* three mechanisms. In the rapidly changing environment, a mixed picture emerges. For the EVO+IL mechanism, it is clear that *abundant* > *balanced* > *scarce*. For EVO, the *scarce* environment does *not* provide significantly different results to the other two, whereas for IL, both *scarce* and *balanced* prove harder than *abundant*, but *scarce* outperforms *balanced*.

5.2 Influence of Environmental Change

Table 7 illustrates how the rate of change of a given environment influences the interaction between environmental parameters and learning mechanisms. In 21/27 pairwise comparisons, statistically significant results are observed.

In the *scarce* environments, there is a general pattern that in terms of rate of change, *static* > $5k$ > $15k$ for *all* mechanisms. In the *balanced* environments, the same general pattern is observed, with the exception that for the IL and EVO+IL mechanisms, no statistical differences are noted between the 5k and 15k environments. In the *abundant* environments, we also note the same general pattern as above, except that for IL, the only significant result shows that 5k>15k significant, while in contrast, for EVO, 5k<15k.

5.3 Influence of learning mechanism

Table 8 provides a pairwise comparison of learning mechanisms within different environments. 22/27 comparisons are significant.

For the *scarce* environment, general pattern that EVO+IL outperforms the other two methods in 4/6 cases, with no statistical difference in the other two cases. In the balanced environment, EVO+IL also clearly dominates both EVO and IL. EVO dominates IL in the static and 5k experiments. Finally, in the *abundant* environment, we again observe the supremacy of EVO+IL, while IL dominates EVO in both of the dynamic environments.

5.4 Analysis

The previous section showed that the EVO+IL clearly outperforms IL and EVO in all parameterisations of the environment and for all rates of change. We examine its behaviour more closely by plotting the normalised difference between the number of positive tokens (p) and the number of negative tokens (n) collected per season over time (i.e. p-n). This is shown in figure 2 for the (scarce, balanced, abundant) environments for the two cases in which the values of the tokens change dynamically with seasons. The solid lines on the graph represent this value combined over both seasons, while the dashed and dotted lines represent the value in season 0 and season 1 respectively. All lines are smoothed over the

Table 6: p-values of pairwise comparison of environments for *totalTokenRatio* (row vs. column) over generation 199 to 200

Experiment			Evo		IL		Evo + IL	
	count		625	1150	625	1150	625	1150
Season		value	625	425	625	425	625	425
0	300	1150	< 1.24e-07	< 3.02e-27	= 5.78e-01	> 7.33e-05	< 3.27e-02	< 1.44e-40
	625	625		< 9.37e-16		> 7.87e-11		< 1.33e-32
5k	300	1150	= 4.55e-01	= 3.08e-01	> 7.89e-05	< 1.99e-02	< 3.87e-19	< 1.5e-32
	625	625		> 1.22e-03		< 1.81e-17		< 5.88e-04
15k	300	1150	< 4.78e-02	< 1.58e-04	= 6.51e-01	< 4.11e-04	< 1.94e-16	< 9.56e-30
	625	625		< 1.09e-08		< 1.44e-12		= 2.61e-01

Table 7: Showing p-values of pairwise comparison of seasonal change for *totalTokenRatio* (row vs. column) over generation 199 to 200

Environment		count:300 value:1150		count:625 value:625		count:1150 value:425	
Experiment	Season	5k	15k	5k	15k	5k	15k
Evo	0	> 9.09e-69	> 6.76e-55	> 1.2e-131	> 4.34e-99	> 6.91e-120	> 9.94e-88
	5k		> 1.89e-02		> 1.67e-05		< 6.04e-03
IL	0	> 2.54e-05	> 6.93e-09	> 4.05e-27	> 4.43e-20	= 1.03e-01	= 7.08e-01
	5k		= 7.61e-02		= 8.51e-02		> 8.33e-03
Evo + IL	0	> 2.82e-35	> 1.37e-31	> 1.76e-32	> 4.15e-32	> 3.41e-178	> 3.54e-118
	5k		> 1.7e-02		= 4.38e-01		= 7.19e-01

Table 8: Showing p-values of pairwise comparison of learning mechanism for *totalTokenRatio* (row vs. column) over generation 199 to 200

Environment		count:300 value:1150		count:625 value:625		count:1150 value:425	
Season	Experiment	IL	Evo + IL	IL	Evo + IL	IL	Evo + IL
0	Evo	> 6.04e-35	= 6.64e-01	> 8.51e-77	= 4.32e-01	> 4.44e-82	< 1.5e-03
	IL		< 3.6e-26		< 6.66e-59		< 6.7e-150
5k	Evo	= 8.45e-01	< 3.18e-05	> 4.42e-06	< 3.36e-55	< 4.22e-15	< 9.94e-122
	IL		< 1.27e-04		< 3.6e-70		< 2.9e-80
15k	Evo	= 7.84e-02	< 8.55e-04	< 7.4e-03	< 1.09e-41	< 1.28e-08	< 3.11e-77
	IL		= 5.27e-02		< 5.83e-42		< 2.23e-72

relevant points. The continuous improvement in this metric is clearly identified for EVO+IL, showing a generally robust response to the changes in token value (i.e. an upward trend). The *abundant* environment proves most straightforward to learn in: having a large *quantity* of information of low-value outweighs the situation in which a small quantity of high-value information is available. In contrast, in the baseline experiment in which *no* information is available as to token value, the (p-n) metric continuously cycles. In this case, the best that evolution can do is learn a token-avoidance behaviour, as there is no means of distinguishing between tokens.

6 CONCLUSION

We have investigated the performance of a number of adaptation mechanisms that augment evolution of a neural network controller. Adaptation mechanisms that included heritable and fixed components were analysed in three different environments in which both the number of learning opportunities and the impact of the learning opportunity varied.

We show that an adaptation mechanism in which all components evolve and are heritable (EVO+IL) copes well in static and dynamic environments, and is able to learn to distinguish between tokens of different value. In dynamic environments, the greatest effect is observed when the environment contains a large number of small learning opportunities. The fewer the learning opportunities, the less effective the mechanism becomes, despite the fact that the opportunities provide more energy and therefore more information to the learning mechanism.

In contrast, the EVO and IL mechanisms both prove to be detrimental in a changing environment when compared to the baseline scenario. No clear pattern emerges however in terms of the magnitude of the effect with respect to the number of learning opportunities present. The IL method

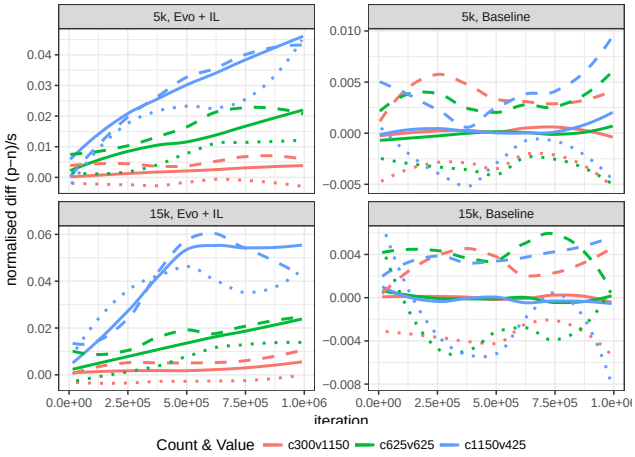


Figure 2: Normalised difference between positive and negative tokens collected. Solid line is value combined over all seasons, dashed = season 0, dotted = season 1

never outperforms the baseline experiments, whereas EVO is beneficial only in a static environment. In the latter case, performance is greatest in the environment with most tokens, and decreases as the number of tokens decreases.

The results clearly demonstrate the interaction between the learning mechanism and environmental parameters. This is of particular relevance for distributed algorithms such as mEDEA in which environmental pressure influences reproductive abilities. The huge variety of behaviour that were displayed in different environments highlight how fundamental it is to not just select parameters at random, but to perform a more thorough analysis. The emerging behaviour using a single set of algorithmic parameter varied from giving a massive advantage, to showing no difference, to even being counter productive. Future work will extend the analysis to other mechanisms for adding individual learning and/or adaptation, as well as considering social learning, recently demonstrated by [11, 12] to be effective in some scenarios.

REFERENCES

[1] Nicolas Bredeche and Jean-Marc Montanier. 2010. Environment-driven embodied evolution in a population of autonomous agents. In *Parallel Problem Solving from Nature, PPSN XI*, Robert Schaefer, Carlos Cotta, Joanna Kołodziej, and Günter Rudolph (Eds.), Vol. 6239. Springer Berlin Heidelberg, Krakov, Poland, 290–299.

[2] Nicolas Bredeche, Jean-Marc Montanier, Berend Weel, and Evert Haasdijk. 2013. Roborobo! a Fast Robot Simulator for Swarm and Collective Robotics. *CoRR* abs/1304.2 (apr 2013). arXiv:1304.2888

[3] Kai Ellefsen. 2013. Balancing the Costs and Benefits of Learning Ability. In *Advances in Artificial Life, ECAL 2013*, Pietro Liò, Orazio Miglino, Giuseppe Nicosia, Stefano Nolfi, and Mario Pavone (Eds.). MIT Press, Taomina, 292–299.

[4] Jorge Gomes, Miguel Duarte, Pedro Mariano, and Anders Lyhne Christensen. 2016. *Cooperative Coevolution of Control for a Real Multirobot System*. Springer International Publishing, Cham, 591–601.

[5] Evert Haasdijk. 2015. Combining Conflicting Environmental and Task Requirements in Evolutionary Robotics. In *2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems*. IEEE, 131–137.

[6] Evert Haasdijk, Agoston Endre Eiben, and Alan Frank Thomas Winfield. 2013. Individual, Social and Evolutionary Adaptation in Collective Systems. In *Handbook of Collective Robotics - Fundamentals and Challenges* (2013 ed.), Serge Kernbach (Ed.). Pan Stanford, Germany, Chapter 12, 411–469.

[7] Evert Haasdijk, P. A. Vogt, and Agoston Endre Eiben. 2008. Social learning in Population-based Adaptive Systems. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. IEEE, 1386–1392.

[8] Evert Haasdijk, Berend Weel, and Agoston Endre Eiben. 2013. Right on the MONEE. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, Christian Blum (Ed.). ACM New York, NY, USA, Amsterdam, The Netherlands, 207–214.

[9] Emma Hart, Andreas Steyven, and Ben Paechter. 2015. Improving Survivability in Environment-driven Distributed Evolutionary Algorithms through Explicit Relative Fitness and Fitness Proportionate Communication. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15*, Sara Silva (Ed.). ACM Press, New York, New York, USA, 169–176.

[10] Jacqueline Heinerman, Dexter Drupsteen, and Agoston Endre Eiben. 2015. Three-fold Adaptivity in Groups of Robots: The Effect of Social Learning. In *Proceedings of the 17th annual conference on Genetic and evolutionary computation*. ACM Press, New York, New York, USA, 177–183.

[11] Jacqueline Heinerman, Massimiliano Rango, and Agoston Endre Eiben. 2015. Evolution, Individual Learning, and Social Learning in a Swarm of Real Robots. In *2015 IEEE Symposium Series on Computational Intelligence*. IEEE, 1055–1062.

[12] Jacqueline Heinerman, Alessandro Zonta, Evert Haasdijk, and Agoston Endre Eiben. 2016. On-line Evolution of Foraging Behaviour in a Population of Real Robots. Springer, Cham, 198–212.

[13] Giles Mayley. 1996. Landscapes, Learning Costs, and Genetic Assimilation. *Evolutionary Computation* 4, 3 (sep 1996), 213–234.

[14] Stefano Nolfi and Dario Floreano. 1999. Learning and evolution. *Autonomous robots* 7, 1 (1999), 89–113.

[15] Nikita Noskov, Evert Haasdijk, Berend Weel, and Agoston Endre Eiben. 2013. MONEE: Using Parental Investment to Combine Open-Ended and Task-Driven Evolution. In *Applications of Evolutionary Computation*, A. I. Esparcia-Alcázar (Ed.), Vol. 7835. Springer, Berlin Heidelberg, 569–578.

[16] Carlos Segura, Carlos A. Coello Coello, Eduardo Segredo, and Arturo Hernandez Aguirre. 2016. A Novel Diversity-Based Replacement Strategy for Evolutionary Algorithms. *IEEE Transactions on Cybernetics* 46, 12 (dec 2016), 3233–3246.

[17] Andreas Steyven, Emma Hart, and Ben Paechter. 2016. Understanding Environmental Influence in an Open-Ended Evolutionary Algorithm. In *Parallel Problem Solving from Nature PPSN XIV*, Julia Handl et al. (Eds.). Vol. 9921 LNCS. Springer International Publishing AG, Chapter 86, 921–931.

[18] Andreas Steyven, Emma Hart, and Ben Paechter. 2015. The Cost of Communication: Environmental Pressure and Survivability in mEDEA. In *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference - GECCO Companion '15*, Sara Silva (Ed.). ACM Press, New York, New York, USA, 1239–1240.

[19] R.S. Sutton and A.G. Barto. 1998. Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks* 9, 5 (sep 1998), 1054–1054.

[20] Joanne H. Walker, Simon M. Garrett, and Myra S. Wilson. 2006. The balance between initial training and lifelong adaptation in evolving robot controllers. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society* 36, 2 (apr 2006), 423–32.

Appendix B

Adaptation Mechanism

B.1 Visualisation of Individual Learning Parameter

Each of the 12 plots in the following figures show the same algorithm deployed in a different experimental configuration. The experiments are made up of three different environments (rows), each experiencing four different rates of change (columns). The rows show different environments (count: 300, value:1150; count: 625, value: 625; count: 1150, value: 425), and the columns show different rates of change (no, every 5k, 15k and 30k iterations).

Data plotted in the individual scatter plots represents 10% of the population (randomly sampled) and is taken from a single randomly selected run of the experiment.

Each of the two sub-graphs is a scatter plot, showing the value of the positive and negative multiplier respectively. Note, the names "positive", for the upper sub-graph, and "negative", for the lower sub-graph, refer to the initial value of the corresponding token at the beginning of the experiment. Every genome has a data point on either of the two sub-graphs. The values end-of-lifetime (x position), distance travelled (y-value) and reproductive success (transparency; measured in number of offspring) are the same in each sub-graph. The value of the multiplier (colour; a continues scale from -1 = red to +1 = green) is unique to the corresponding sub-graph. The size of the dot shows the

adaptation success of the individual genome: positive tokens collected per lifetime in the upper sub-graph and the inverse ratio for negative tokens in the lower sub-graph. The red/green lines below the y-axes is made up of dots, each representing the learning sign *LS* of the individuals: red=negative and green=positive. Note that *LS* is negated in the lower sub-graph.

Fig. 1: Evo + IL

(a) No Seasonal Change (b) 5k Season Change (c) 15k Season Change (d) 30k Season Change

count: 300, value: 1150

count: 625, value: 625

count: 1150, value: 425

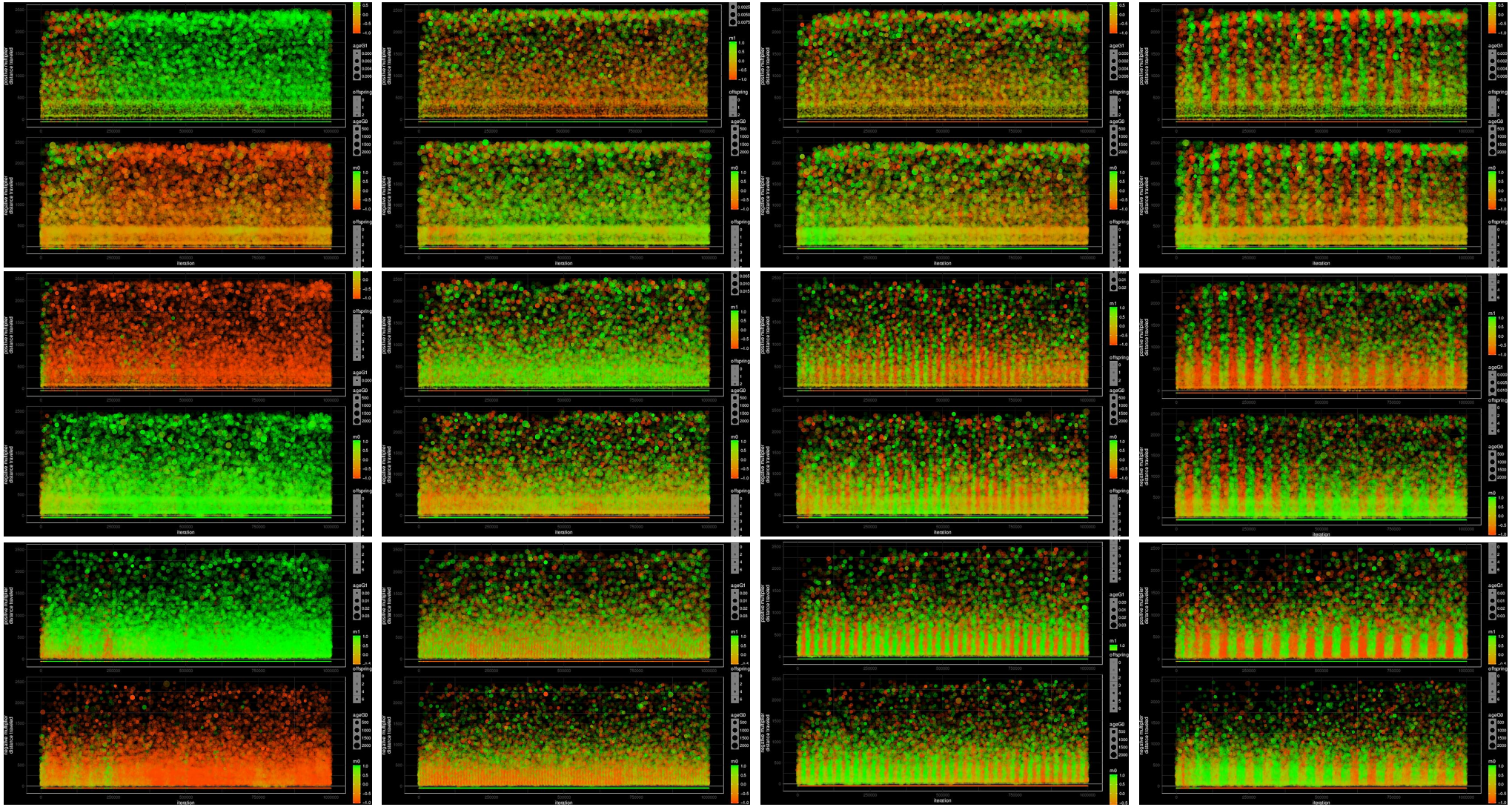


Fig. 2: Evo

(a) No Seasonal Change (b) 5k Season Change (c) 15k Season Change (d) 30k Season Change

count: 300, value: 1150

count: 625, value: 625

count: 1150, value: 425

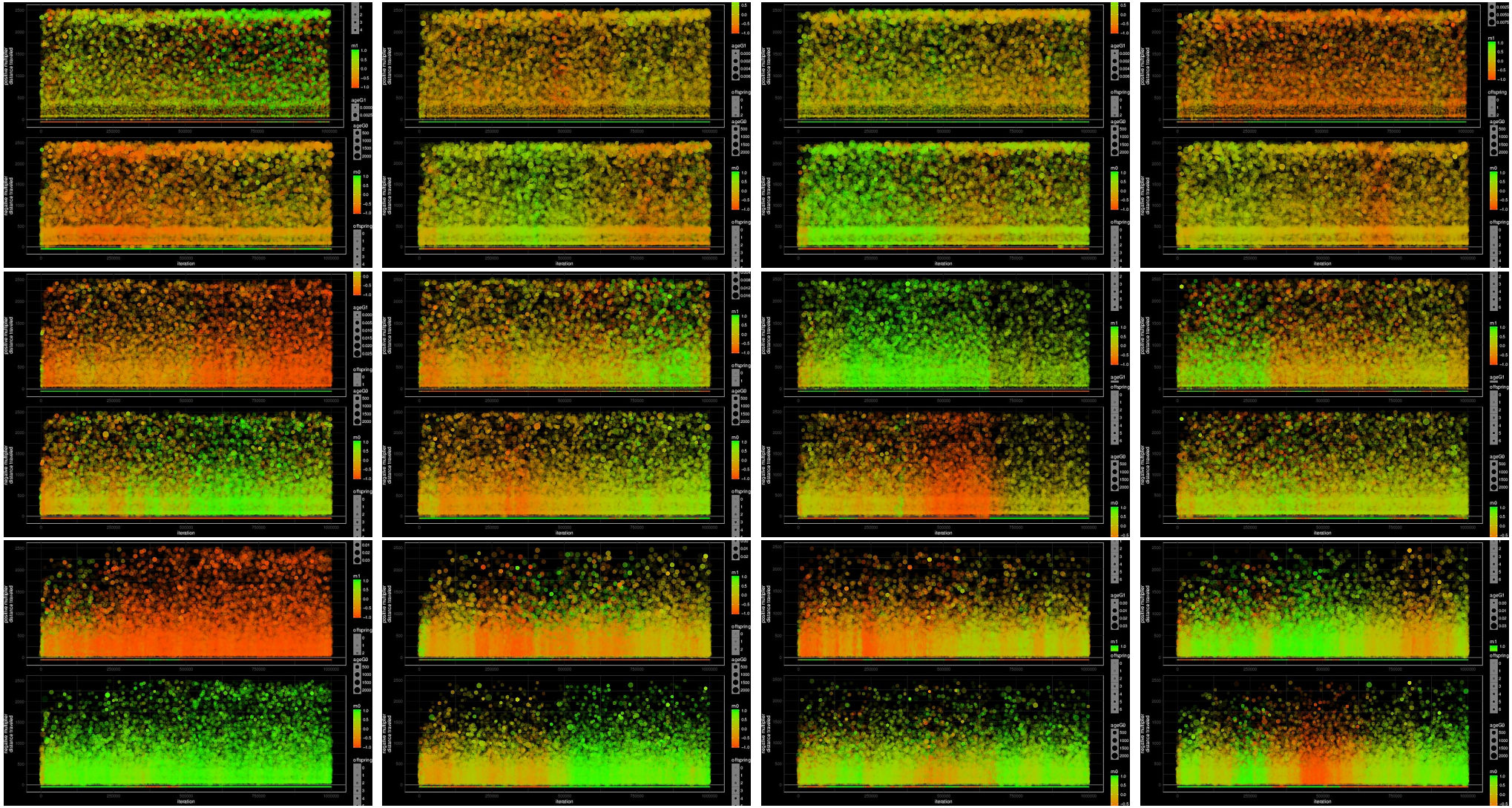


Fig. 3: IL

(a) No Seasonal Change (b) 5k Season Change (c) 15k Season Change (d) 30k Season Change

count: 300, value: 1150

count: 625, value: 625

count: 1150, value: 425