Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

EVOLUTION OF SELF-ORGANISING BEHAVIOURS WITH NOVELTY SEARCH

Jorge Miguel Carvalho Gomes

DISSERTAÇÃO

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Interação e Conhecimento

2012

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

EVOLUTION OF SELF-ORGANISING BEHAVIOURS WITH NOVELTY SEARCH

Jorge Miguel Carvalho Gomes

DISSERTAÇÃO

Projecto orientado pelo Prof. Doutor Paulo Jorge Cunha Vaz Dias Urbano
e co-orientado pelo Prof. Doutor Anders Lyhne Christensen

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Interação e Conhecimento

2012

Acknowledgments

In the first place, I would like to thank my supervisors, Paulo Urbano and Anders Christensen, without them this work would not exist and they were great at guiding me through it. I am grateful to Paulo for accompanying me through my academic years, as a teacher and as supervisor in a few research projects, always with an incredible creativity, genius ideas, and an insatiable curiosity. And I am grateful to Anders Christensen for teaching me scientific rigour, and how to write clearly and expose my ideas.

I am also thankful to Joel Lehman for being available and enthusiastic to discuss novelty search issues with us, as well as Kenneth O. Stanley for answering our questions about NEAT and novelty search. They both did great works and made an excellent job at promoting them and making them available to the community.

I would like to thank everyone at LabMAg for the companionship and brainstorming. Special thanks to my long time workmates Fernando Silva and Davide Nunes for putting up with me, to Nuno Henriques for teaching me how to make the world a better place, and to Phil Lopes, Edgar Montez, João Silva and Christian Marques for being friends besides colleagues.

To my family, for providing for my education and for giving me the moral values needed to succeed in life. And to my close friends, for spicing up my life with fun and fellowship.

Finally, I wish to thank Viviana, for all the love, care, and for distracting me from my work when I needed.

To my parents and my brother.
To Viviana.

Resumo

A pesquisa de novidade (*novelty search*) é um nova e promissora técnica de evolução artificial, que desafia a abordagem tradicional focada na perseguição direta dos objetivos. O principal conceito por trás da pesquisa de novidade é a recompensa de soluções que sejam novas, em vez de soluções que se aproximem do objetivo pré-definido. Este carácter divergente da procura faz com que a pesquisa de novidade não esteja sujeita a alguns problemas comuns na evolução artificial, tal como a convergência prematura e a deceção da função objetivo, pois na pesquisa de novidade o objetivo não tem influência direta no processo evolutivo. A função objetivo diz-se decetiva quando ela conduz a população do algoritmo evolucionário para máximos locais, e como consequência não consegue atingir o objetivo desejado numa quantidade razoável de tempo.

No algoritmo de pesquisa de novidade, a função objetivo é substituída por uma métrica que quantifica a novidade das soluções, baseando-se em caracterizações de comportamento que são obtidas para cada uma. A função que obtém estas caracterizações deve ser definida pelo humano que conduz o processo, usando conhecimento sobre o domínio e tendo em consideração a tarefa que se está a tentar desempenhar. A novidade de cada individuo é medida relativamente à população corrente e a um arquivo de indivíduos que representa o espaço de comportamentos que já foi anteriormente explorado. Desta forma, soluções que se situem em locais do espaço de comportamentos que estejam pouco explorados são consideradas mais aptas para seleção, e vice-versa, guiando o processo evolutivo em direção à diversidade comportamental. Contraintuitivamente, embora a pesquisa de novidade ignore totalmente o objetivo, ela revelou em vários casos um maior sucesso que a abordagem tradicional baseada em objetivos, especialmente em tarefas onde a função objetivo sofre de algum grau de deceção.

Em trabalhos anteriores, a pesquisa de novidade foi aplicada com sucesso em tarefas de robótica não coletiva. Nesta tese, propomos a aplicação da pesquisa de novidade à evolução de controladores para robótica coletiva, uma área que tem sido dominada pelas técnicas de evolução tradicionais, focadas em objetivos. A motivação para a aplicação da pesquisa de novidade a esta área é o elevado nível de complexidade na robótica coletiva, resultante das relações entre os vários agentes do

grupo, e entre os agentes e o seu ambiente. À medida que um sistema se torna mais complexo, a função objetivo é mais suscetível de se tornar decetiva, e a pesquisa de novidade é particularmente eficaz a lidar com a deceção da função objetivo. Ultrapassar o problema da deceção neste domínio é mais um passo em direção à geração automática de controladores para grupos de robôs capazes de resolver tarefas com a complexidade do mundo real. O caráter da pesquisa de novidade orientado à diversidade comportamental é também interessante neste domínio, pois permite a geração de uma diversidade de soluções para o mesmo problema, possivelmente revelando formas originais de auto-organização.

Nas nosso trabalho, os controladores que são usados pelos grupos de robôs (todos os robôs do grupo usam o mesmo controlador) são redes neuronais recorrentes. O método escolhido para implementar o processo neuro-evolutivo foi o NEAT. A pesquisa de novidade é implementada sobre o NEAT, da forma como foi descrito acima. O NEAT é um método neuro-evolutivo que modifica tanto os pesos das ligações da rede, como a sua estrutura, podendo adicionar e remover nós e ligações. Começa com um conjunto de redes neuronais simples, completamente ligadas e sem nós intermédios, e vai gradualmente complexificando as redes neuronais, à medida que se verifique vantajoso, podendo levar à evolução de comportamentos gradualmente mais complexos.

Para conduzir o estudo descrito nesta tese, foi seguida uma abordagem experimental, através da realização de ensaios evolucionários com diferentes técnicas evolucionárias, parâmetros, e tarefas. Em cada ensaio foram recolhidas informações e métricas detalhadas de forma a facilitar a compreensão das dinâmicas evolucionárias. Para a execução dos ensaios evolucionários, foi desenvolvida uma nova aplicação, baseada num simulador de robótica existente e numa implementação do NEAT. A aplicação é altamente modular, permitindo a definição de novos ambientes, robôs, métodos evolucionários, entre outros, sem ter que modificar código fonte existente.

O primeiro passo do nosso trabalho consistiu em aplicar o algoritmo original de pesquisa de novidade à evolução de controladores para um grupo de robôs que deve executar uma tarefa de agregação. Nesta tarefa (amplamente estudada em trabalhos anteriores), os robôs são colocados em posições aleatórias dentro de uma arena fechada, e têm como objetivo formar um único agregado compacto, em qualquer ponto da arena. A tarefa é dificultada por uma arena de grandes dimensões e robôs com sensores de curto alcance. Foram realizadas experiências com a pesquisa de novidade usando três diferentes caracterizações de comportamento: uma altamente correlacionada com o objetivo, outra pouco correlacionada, e finalmente a combinação das duas. Foi também experimentada a evolução tradicional guiada por objetivos.

De seguida, é experimentada a aplicação da pesquisa de novidade a uma tarefa de gestão coletiva de energia, em que os robôs gastam energia ao longo do tempo e devem coordenar-se para permitir o acesso periódico à única estação de recarga, de modo a sobreviverem. São definidas duas variantes desta tarefa, uma em que os robôs gastam sempre a mesma quantidade de energia ao longo do tempo, e outra em que a quantidade de energia despendida depende da velocidade dos robôs. Na primeira variante, a função objetivo consegue guiar eficazmente a população em direção ao objectivo. Na segunda variante, a função objetivo é claramente decetiva, e conduz a população para máximos locais muito prematuros. Foram também experimentadas duas caracterizações comportamentais distintas na pesquisa de novidade: uma caracterização curta, altamente relacionada com o objetivo, e outra caracterização expandida, com algumas dimensões não relacionadas com o objetivo.

Os resultados destas experiências revelam que a pesquisa de novidade pode ser um método eficaz para evolução de controladores para robótica coletiva. A pesquisa de novidade mostrou ser eficaz em ultrapassar a deceção da função objetivo, evitando com sucesso os máximos locais. Foi particularmente bem sucedida na inicialização da evolução, evitando a convergência prematura e atingindo elevados valores de *fitness* cedo na evolução. Foram estabelecidas comparações detalhadas entre a pesquisa de novidade e o método evolutivo tradicional, baseado em objetivos. Em configurações onde a deceção da função objetivo não era um problema, a pesquisa de novidade obteve um desempenho semelhante à evolução guiada por objetivos, em termos dos valores de *fitness* das soluções evoluídas. Por outro lado, em configurações onde a função objetivo era decetiva, a pesquisa de novidade revelou-se claramente superior. Os resultados também mostram que a pesquisa de novidade consegue evoluir soluções com redes neuronais mais simples, em comparação com a evolução guiada por objetivos. Os nossos resultados representam uma contribuição relevante para o domínio da robótica coletiva evolucionaria, pois os trabalhos anteriores revelam dificuldades em evoluir grupos de robôs capazes de desempenhar tarefas ambiciosas. As experiências sugerem que a evolução de comportamentos coletivos é especialmente suscetível à deceção da função objetivo, e como tal a pesquisa de novidade revela-se como uma promissora alternativa para ultrapassar esta dificuldade, e conseguir a evolução de comportamentos coletivos mais ambiciosos.

Os resultados também revelaram que a pesquisa de novidade pode ser utilizada para descobrir uma ampla diversidade de formas de auto-organização. A procura de diversidade em robótica coletiva é um tópico relevante porque tipicamente existe um grande leque de possibilidades de comportamentos, resultante das possíveis interações entre os vários robôs do grupo, e entre os robôs e o ambiente. Procurar ativamente estas possibilidades pode levar a formas inesperadas de auto-organização e diferentes soluções para o mesmo problema. Por exemplo, nas experiências com a

tarefa de agregação, a pesquisa de novidade evoluiu um tipo de comportamentos de agregação que não é descrito no trabalho relacionado, mas que pode ser encontrado no mundo natural.

Estas experiências forneceram também alguma compreensão sobre como devem ser construidas as caracterizações comportamentais a usar na pesquisa de novidade. Mostrámos que combinar várias medidas pode ser uma forma de aumentar o desempenho da pesquisa de novidade. No entanto, deve-se evitar acrescentar à caracterização do comportamento dimensões que estejam pouco relacionadas com a tarefa que se está a tentar resolver. Neste caso, os resultados mostraram que a pesquisa de novidade pode começar a focar-se em zonas do espaço de comportamentos que não são relevantes para a solução da tarefa. Para visualizar e analisar espaços de comportamentos de elevada dimensionalidade, foram utilizados mapas de Kohonen auto-organizados. Esta técnica de visualização mostrou ser útil para uma melhor compreensão da dinâmica evolucionária na pesquisa de novidade.

Como referido acima, os resultados mostraram que a pesquisa de novidade pode ter dificuldade em encontrar boas soluções em espaços de comportamentos que tenham dimensões não relacionadas com o objetivo. Para ultrapassar este problema, estendemos o nosso estudo para variantes da pesquisa de novidade que combinam a diversidade comportamental com a função objetivo. Propomos um novo método para combinar a pesquisa de novidade com os objetivos, *chamado Progressive Minimal Criteria Novelty Search* (PMCNS). Este método restringe progressivamente o espaço de comportamentos, através da definição de um limiar de *fitness* que os indivíduos devem superar para serem selecionados para reprodução. Este limiar é dinâmico, começando sem impacto e aumentando progressivamente à medida que a população se vai aproximando do objetivo.

Para avaliar este novo método, foram realizadas experiências com as tarefas de agregação e gestão coletiva de energia, já apresentadas anteriormente. O PMCNS foi comparado com outro método bem sucedido, onde a avaliação de cada individuo consiste numa combinação linear dos seus valores de *fitness* e novidade. Os resultados mostram que o PMCNS é um método eficaz em direcionar a exploração do espaço de comportamentos para as zonas associadas a soluções de elevada qualidade, sem comprometer a diversidade que é descoberta pela pesquisa de novidade, e conseguindo na mesma ultrapassar a deceção da função objetivo. O desempenho do PMCNS foi superior a todos os outros métodos testados.

Palavras-chave: Robótica evolucionária, algoritmos genéticos, neuro-evolução, robótica coletiva, pesquisa de novidade, NEAT, auto-organização

Abstract

Novelty search is a recent artificial evolution technique that challenges the traditional evolutionary approach. The main idea behind novelty search is to reward the novelty of solutions instead of progress towards a fixed goal, in order to avoid premature convergence and deception. Deception occurs in artificial evolution when the objective-function leads the population to local maxima, failing to reach the desired objective. In novelty search, there is no pressure to evolve better solutions, only pressure to evolve solutions different from the ones seen so far, thus avoiding the potential deceptiveness of an objective-function. In previous works, novelty search has been applied with success to single robot system. In this thesis, we use novelty search together with NEAT to evolve neuro-controllers for homogeneous swarms of robots. The aim of this approach is to facilitate the achievement of more ambitious objectives through artificial evolution, and in the end contribute towards the evolution of robotic swarms capable of taking on complex, real-world tasks.

Our empirical study is conducted in simulation and uses two common swarm robotics tasks: aggregation, and sharing of an energy recharging station. Our results show that novelty search is capable of overcoming deception, and is notably effective in bootstrapping the evolution. In non-deceptive setups, novelty search achieved fitness scores similar to fitness-based evolution. Novelty search could evolve a broad diversity of solutions to the same problem, unveiling interesting forms of self-organization. Our study also encompasses variants of novelty search that combine novelty with objectives, in order to combine the exploratory character of novelty search with the exploitatory character of objective-based evolution. We propose Progressive Minimal Criteria Novelty Search (PMCNS), a novel method for combining novelty and objectives, where the exploration of the behaviour space is progressively restricted to zones of increasing fitness scores. We show that PMCNS can improve the fitness scores of the evolved solutions, without compromising the diversity of behaviours. Overall, our study shows that novelty search is a promising alternative for the evolution of controllers for robotic swarms.

Keywords: Evolutionary robotics, genetic algorithms, neuro-evolution, swarm robotics, novelty search, NEAT, self-organisation

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> In the measurement world, we set a goal and strive to achieve it. In the universe of possibility, we set the context and let life unfold.
>
> Benjamin Zander

The quote above raises an interesting question that often challenges common intuitions: should we really strive blindly to achieve previously set goals? Can't the best way to achieve a goal be to forget the goal for moments and just let things unfold, explore the possibilities, and see where it leads?

This absence of rigid goals can actually lead to great accomplishments. If we look onto the biological evolution, there are no pre-defined goals; however this open-endness has lead to incredibly complex and fit organisms. What drives the evolution forward is not the direct pursuit for goals, but rather the constant generation of diversity, which is the result of reproduction and mutation mechanisms that always create genetic diversity. This pressure towards diversity indirectly leads to organisms that develop the required traits for the survival of the species — without ever directly pursuing those traits or even knowing *a priori* what they are.

Novelty search (Lehman and Stanley, 2011a) is a unique evolutionary search method inspired by this character of biological evolution, which abandons the objectives and guides the evolution towards behavioural diversity instead. In traditional evolutionary computation (henceforth referred as *fitness-based evolution*), candidate solutions are scored by a fitness function that has been derived directly from the task or problem for which a solution is sought. Novelty search does not drive the evolutionary process toward a fixed goal. In novelty search, candidate solutions are scored based on how different they are from solutions seen so far, and the evolutionary process is therefore continuously driven towards novelty, following a divergent process.

The objective-oriented evolutionary approach seems intuitively more adequate to achieve goals, however, it is associated with a number of issues, where *decep-*

tion is one of the most prominent. Deception (Goldberg, 1987) is a challenging problem in evolutionary computation, and occurs when the evolutionary process converges prematurely to local optima, thus failing to reach the desired objective in a reasonable amount of time. Deception can be caused by an objective function that fails to reward the intermediate steps needed to achieve the final goal, or deceives the search by actively leading it in a wrong direction. Novelty search has the potential to overcome deception, since it is not influenced by the objective function, and has a divergent nature, thus avoiding premature convergence. Lehman and Stanley (2011a) have shown that, although novelty search does not pursue a goal directly, it may be able to find the goal faster and more consistently than traditional fitness-based evolution. Novelty search has also proven capable of finding a greater diversity of solutions to a problem than traditional fitness-based evolution (Lehman and Stanley, 2011b).

Novelty search has been successfully applied to many domains, including non-collective evolutionary robotics in tasks such as maze navigation (Lehman and Stanley, 2011a; Mouret, 2011), T-maze tasks that require lifetime learning (Risi et al., 2009), biped walking (Lehman and Stanley, 2011a), and the deceptive tartarus problem (Cuccu and Gomez, 2011). There are many motivations behind the use of evolutionary techniques for the design of a control system for a robot (Harvey et al., 1993). In a multirobot domain in particular, the dynamical interactions among robots and the environment make it difficult to hand-design a control system for the robots that yields the desired macroscopic swarm behaviours. Artificial evolution has been shown capable of exploiting these dynamic features and synthesise self-organised behaviours (Trianni et al., 2003).

However, evolutionary robotics has only been proven effective in simple tasks, under laboratory conditions (Sprong, 2011). The main challenge in evolutionary robotics is the difficulty in guiding the evolution towards solutions that are able to solve complex problems. As the complexity of a task or a system increases, artificial evolution is more likely to get affected by deception, which causes the evolutionary process to fail. Complex tasks also cause issues in bootstrapping the evolution (Gomez and Miikkulainen, 1996), because the qualities of the solutions in the initial population, which are randomly generated, may not be adequately distinguished by the objective function. As such, the evolution has no clues about which are the most promising individuals that should be propagated to the next generations. Some solutions to these problems have been proposed, such as incremental evolution (Gomez and Miikkulainen, 1996) and multi-objective evolutionary algorithms (Knowles et al., 2001). However, these methods are highly dependent on domain knowledge, that should be provided *a priori* by the experimenter. This dependence on domain knowledge is not desirable, because it can bias the evolutionary process

and ultimately is contrary to the ambition of having a method that automatically generates robotic controllers.

In this thesis, we use novelty search to evolve neural controllers for swarm robotic systems. Most previous works in evolutionary swarm robotic systems use objective-based evolutionary approaches. Our motivation for applying novelty search to swarm robotic systems is their high level of complexity, resulting from the intricate dynamics between many interacting units. This high level of complexity is prone to generate deceptive fitness landscapes (Whitley, 1991), and novelty search has been shown capable of overcoming deception (Lehman and Stanley, 2011a). Evolutionary techniques based on diversity maintenance mechanisms have also been proven effective in bootstrapping the evolutionary process (Mouret and Doncieux, 2009), which is another challenge in evolving complex systems and solving complex tasks. By overcoming these challenges with novelty search, we will work towards solutions for evolving multi-robot systems able to perform complex tasks, requiring less effort and intervention from the experimenter when compared to other techniques such as fitness shaping and incremental evolution.

The drive of novelty search towards behavioural diversity is another motivation for this work. Novelty search can generate a diversity of solutions in a single evolutionary run, as opposed to fitness-based evolution, in which a particular run often converges to a single solution. This diversity can provide a range of different solutions to the experimenter who is conducting the evolutionary process. This is especially relevant in the domain of swarm robotics, because the dynamical interactions between the robots and the environment may result in many behavioral possibilities (Trianni, 2006). Novelty search can explore these possibilities, potentially revealing new and unexpected forms of self-organisation.

As novelty search is guided by behavioural innovation alone, its performance can be greatly affected when searching through vast behaviour spaces (Lehman and Stanley, 2010a; Cuccu and Gomez, 2011), since it may spend most of its time exploring behaviours that are irrelevant for the goal task. There are a few methods that were proposed to overcome this limitation, such as Minimal Criteria Novelty Search (Lehman and Stanley, 2010a), among others that bring together behavioural diversity and fitness evaluation (Mouret and Doncieux, 2012). In this thesis, we investigate the application of some of these methods in the swarm robotics domain, in order to cope with behaviour spaces where novelty search fails to explore the high fitness behaviour zones.

The evolution of the network topology along with the weights has proved to be beneficial in many domains (Stanley, 2004). However, most previous works in evolutionary swarm robotics use neuroevolution methods that optimise only the weights of the neural network. In this thesis, we use NEAT (NeuroEvolution of

Augmenting Topologies) (Stanley and Miikkulainen, 2002) to evolve the neural controllers used by the robots in a swarm. NEAT is a method that evolves both the network topology and weights, allowing solutions to become gradually more complex as they become better (Stanley and Miikkulainen, 2002). The use of NEAT as the underlying neuroevolution method in novelty search is motivated by the complexifying nature of NEAT, which imposes some order in the exploration of the behaviour space, because simple controllers are explored before moving on to more complex ones.

1.1 Original Contributions

The focus of our research is the application of evolutionary techniques based on novelty search to synthesise controllers for swarms of robots. This thesis comprises the following contributions:

1. This work is the first to introduce novelty search to the domain of evolutionary swarm robotics. We compare novelty search with the more traditional objective-based evolution.

2. In our work, the evolutionary methods are implemented with NEAT, a popular neuroevolution method with incremental complexification of network topologies. This is the first work to apply NEAT to the evolution of controllers for robotic swarms.

3. Our empirical study is based on two popular collective robotics tasks: aggregation and sharing of an energy recharging station. The energy management task represents a relevant problem that is not much addressed by the evolutionary approach. We show that evolutionary robotics can synthesise good solutions for this task.

4. We show that novelty search is effective in overcoming deception in the swarm robotics domain, and can successfully bootstrap the evolution. We study how to devise novelty distance metrics for this domain.

5. We show that novelty search has the potential of unveiling a broad diversity of collective behaviours. Previous works on evolutionary swarm robotics did not focus on discovering behavioural diversity.

6. This thesis proposes *progressive minimal criteria novelty search* (PMCNS), an extension of *minimal criteria novelty search*, that combines novelty search with objective-based evolution by progressively restricting the behaviour search space. PMCNS is compared with other novelty search variants described in the literature.

1.2 Context

This thesis is the final work for the obtainment of the Masters Degree in Informatics Engineering by Faculty of Sciences of the University of Lisbon (FCUL). This thesis was oriented by the Professor Paulo Jorge Cunha Vaz Dias Urbano (FCUL) and by the Professor Anders Lyhne Christensen (ISCTE).

The work was developed in LabMAg – Laboratory of Agent Modelling, a research unit dedicated to computational models of agents, hosted in the Informatics Department of FCUL.

1.3 Thesis Structure and Related Publications

Below, we provide an overview of the thesis. This thesis describes an original research, some of which has been published in proceedings of international conferences by the author and the supervisors. The publications related to the results presented in this document are listed below.

In Chapter 2, we present and explain the core concepts approached in this thesis, and discuss the related work. We start by briefly presenting evolutionary robotics, and evolutionary swarm robotics in particular. We present neuroevolution and why it is fit for the evolution of robot controllers. We then present in greater detail the neuroevolution method NEAT (NeuroEvolution of Augmenting Topologies). We move on to the problem of deception in evolutionary computation, and how novelty search can overcome this problem. The novelty search algorithm is presented in detail and we discuss some of the applications found in the current literature. We finish by describing limitations of novelty search and how some of these limitations were addressed in related work. We briefly present other evolutionary methods inspired by novelty search that can be found in the literature.

In Chapter 3, we apply the original novelty search algorithm to the evolution of controllers for two swarm robotics tasks. First, we experiment with the aggregation task. We begin by presenting previous works that used this task, and then describe the experimental setup used in our study. We experiment with different novelty measures to assess their impact in the effectiveness of the evolution and in the exploration of the behaviour space, and establish comparisons with the traditional fitness-based evolution. We then move on to the energy management task, presenting the previous works related to this task and the experimental setup. We show how novelty search can overcome deception, by using two variants of the task with different levels of deceptiveness. We also study how novelty search is affected by large behaviour spaces, using two novelty measures with different levels of detail. We wrap up with a discussion encompassing the results obtained in the experiments with both tasks. Chapter 3 is partially based on the following published papers:

– Gomes, J., Urbano, P., Christensen, A.L.: Introducing novelty search to evolutionary swarm robotics. In: Proceedings of the 8th International Conference on Swarm Intelligence (ANTS 2012), pp. 85–96. Springer Verlag, Berlin, Germany (2012). *Invited for publication of an extended version the Swarm Intelligence journal.*

– Gomes, J., Urbano, P., Christensen, A.L.: Diverse Behaviors in Swarm Robotics with Novelty Search. In: Proceedings of the Thirteenth International Conference on the Simulation and Synthesis of Living Systems (ALIFE XIII), pp. 553-554. MIT Press (2012).

In Chapter 4, we study evolutionary techniques that combine novelty search with fitness-based evolution. We propose *Progressive Minimal Criteria Novelty Search*, an evolutionary technique based on novelty search that intends to overcome the limitations of *Minimal Criteria Novelty Search*. To evaluate the proposed technique, we revisit the tasks used in Chapter 3. Using the energy management and aggregation tasks, we compare PMCNS with an evolutionary technique that uses a linear blend of novelty and fitness scores, and with the results obtained in the previous chapter using novelty search and fitness-based evolution. We also study the parameters of both evolutionary techniques, especially the balance between novelty search and fitness-based evolution. We finish with a discussion compiling all the experimental results presented in the chapter. Chapter 4 is partially based on the following paper:

– Gomes, J., Urbano, P., Christensen, A.L.: Progressive Minimal Criteria Novelty Search. In: Proceedings of the 13th edition of the Ibero-American Conference on Artificial Intelligence (IBERAMIA 2012), Springer, Berlin, Germany (2012), *in press.*

In Chapter 5, we confront the work presented in this thesis with the initial objectives layed out in the preliminary report, summarise the major findings described in this thesis, discuss the challenges that were found, and point out some future work directions.

Appendix A describes the software application developed in the context of this thesis for carrying on the experimental studies.

Chapter 2

Related Work

In this chapter, we address the core concepts approached in this thesis and review the related work. We start by introducing the field of evolutionary robotics, and we discuss some of the most notable works, then we move on to swarm robotics and how evolutionary approaches have been applied to such systems. We then approach in greater detail how robot controllers can be evolved via neuroevolution and will explain how NEAT algorithm works. We move on to the problem of deception in evolutionary computation, a frequent problem in evolutionary robotics, and how it can be overcome. We finally explain in detail how novelty search has the potential to overcome the deception problem and present similar approaches already developed.

2.1 Evolutionary Robotics

Evolutionary robotics is a field of research that applies artificial evolution to the generation of control systems for autonomous robots. During evolution, robots attempt to perform a given task in a given environment. The controllers of the better performing robots are selected, altered, and propagated, to perform the task again in an iterative process that mimics some aspects of natural evolution (Nelson et al., 2009).

The concept of evolutionary robotics, based on neural networks controllers and evolutionary algorithms, is not a recent idea. Beer and Gallagher (Beer and Gallagher, 1992) introduced the idea of evolving agent controllers through genetic algorithms (Holland, 1975). Evolutionary algorithms, such as genetic algorithms, are optimisation methods that use operators of reproduction, mutation, and selection to artificially evolve solutions for a given problem. Candidate solutions of the problem play the role of individuals in a population, and a fitness function determines which individuals are best suited for solving the problem. The evolution of the population then takes place through the repeated application of the genetic operators.

According to (Beer and Gallagher, 1992), the evolutionary approach is more

adequate than traditional symbolic AI in the task of developing adaptive behaviours, since it promotes the shaping of the agents to their environment, and does not depend on the ability of the designer to consider all the possible contingencies. Using the evolutionary approach, the intelligent behaviours emerge from the interaction between an agent's internal control mechanisms and its external environment, rather than from an agent's ability to reason explicitly with symbolic representations of its situation. In (Beer and Gallagher, 1992), the controllers are artificial neural networks, and are evolved with genetic algorithms to perform two tasks: i) Chemotaxis, which consists of following chemical signals in order to reach the source of the signal, and ii) legged locomotion, a much more difficult problem since the agent has to simultaneously solve the problems of support and progression. The evolutionary process was successful in evolving controllers for both tasks, which deemed this approach very promising.

The foundations of evolutionary robotics were later established in (Nolfi and Floreano, 2000), where the basic concepts and methodologies are described, and a set of empirical experiments of different complexity is presented. Many problems have been solved with evolutionary robotics, but the complexity of the solved problems has not yet reached the point where they can be useful in real world complex problems. The most common tasks solved by evolutionary robotics are described in a recent survey (Sprong, 2011).

The evolutionary process typically requires a fitness function to measure how well each controller performs in respect to the objective. This is a key component in evolution, as it establishes the goal the robot should achieve, and can also provide some clues on how to achieve it by incorporating *a priori* knowledge. A survey of fitness functions in the field of evolutionary robotics can be found in (Nelson et al., 2009).

The fitness evaluation of each controller usually consists of running a simulation with the robots in their environment, using that controller, and measuring their performance according to the defined fitness function. Although some attempts have been made to evolve the controllers entirely in the physical robots (Floreano and Mondada, 1994, 1996), this process is typically too slow and cumbersome, so it usually is made in simulated environments. In (Jakobi et al., 1995) is described the gap between the real and simulated worlds, the common pitfalls when evolving controllers in simulated environments and how to overcome them. This work demonstrates that it is possible to develop successful robot controllers in simulation that generate almost identical behaviours in reality.

2.2 Evolutionary Swarm Robotics

Multi-robot systems are inspired by the observation of social activities, which are based on concepts like division of labour, cooperation and communication. If such collective organisation can benefit societies, robotic groups could also benefit from those same concepts, gaining numerous advantages, such as: ability to solve more difficult tasks; robustness to the failure of individuals; versatility; parallelism of operation (Jones and Mataric, 2005).

One research area of collective robotics is swarm robotics (Şahin, 2005), which our work will be based upon. In swarm robotics, large homogeneous groups of robots coordinate themselves to accomplish complex tasks. In (Dorigo and Şahin, 2004), four criteria are given to measure the degree to which a robotic system can be considered a swarm robotic system:

i. The study should be relevant for the coordination and control of a large number of robots, including all approaches that aim for scalability.

ii. The study should involve relatively few groups of homogeneous robots, each group comprising a large number of individuals.

iii. The study should consider tasks that cannot be efficiently solved by a single robot, due to individual limitations.

iv. The study should involve robots that have local and limited sensing and communication abilities.

In (Trianni, 2006), the author identifies the main challenges in the design of swarm robotic systems and why the evolutionary approach is particularly useful for this purpose. According to the authors, manually designing individuals of a swarm is a very challenging task, because one has to decompose the global behaviour into behavioural rules for the individual robots, taking into account the interactions among the system components. This requires discovering the relevant interactions between the individual robots and between them and the environment, which will ultimately lead to the emergence of global coordinated behaviour.

Evolutionary robotics represents an effective solution to this design problem because it eliminates the necessity of decomposing the swarm behaviour. Instead, the system is evaluated as a whole and then relies in the evolutionary process to synthesise the controller that will be used locally by the individuals. Several works have been developed using evolutionary robotics in swarm systems, in order to evolve fairly simple collective behaviours, such as coordinated motion (Baldassarre et al., 2007), foraging (Liu et al., 2007), aggregation (Trianni et al., 2003), and hole avoidance (Trianni et al., 2006). An extensive survey over the modelling of swarm

robotics and the problems that have been solved so far can be found in (Bayindir and Şahin, 2007).

2.3 Neuroevolution

In order for the robots to be autonomous, they need some kind of controller that can process the sensorial data and control the actuators based on that information. One of the most used structures to model controllers in evolutionary robotics are Artificial Neural Networks (Haykin, 1994). An ANN is an information processing paradigm that is inspired by the way biological brains process information. It is composed of a large number of highly interconnected processing elements (neurons) working as whole to achieve complex mappings between inputs and outputs.

In (Floreano and Mondada, 1994) are identified some reasons why artificial neural networks are particularly adequate for the control system of artificial autonomous agents:

i. They are flexible. Their ability to learn enables dynamic adaptation of robot behaviour to changes in the environment, even when the networks have a fixed structure.

ii. They deal with the micro-structure of the robot, meaning that they can exploit at its best the sensory-motor features of the robot, and can learn to use only a subset of the available sensors and actuators.

iii. They have a good tolerance to noise, making them good candidates for mediating between physical sensors and actuators with intrinsic noise.

iv. With recurrent and lateral connections, they can cope with temporal structure and complex mappings required in many real-world tasks.

Neural networks are also well-suited for artificial evolution because small changes in the network typically correspond to small changes in its behaviour, allowing the evolutionary algorithms to progress gradually towards the solution. Harvey et al. (Harvey et al., 1993) also advocates the use of neural networks for evolutionary robotics and compares them with other types of evolvable controllers, such as high level programs, evolved via genetic programming (Koza, 1992), or polynomial transfer functions.

Neuroevolution is a field of research that focuses on methods for evolving artificial neural networks with evolutionary algorithms. Some methods start from a fixed network topology and evolve only the network connection weights, while others evolve the network topology along with the weights, generating new nodes and

connections alongside with the adjustment of the weights of the existing connections. Evolving topologies brings a few more challenges, such as a crossover method that should work between different topologies and the necessity of protecting new network structures in order for them to have space to optimise their weights. However, the evolution of the topology along with the weights has shown a number of advantages (Stanley and Miikkulainen, 2002). First, the experimenter does not have to decide the topology of the network, which typically requires a trial and error process. Second, evolving the topology can actually increase by several times the efficiency of neuroevolution, when compared to other fixed-topology methods.

Many systems have been developed that evolve both neural network topologies and weights, differing in the underlying evolutionary algorithm that is used, in the encoding of the neural networks and on the aspects of the network that are evolved. In this work, we will focus in one algorithm, NeuroEvolution of Augmenting Topologies (NEAT) (Stanley and Miikkulainen, 2002). NEAT is based on genetic algorithms and modifies both the network structure and weights through a complexification mechanism. NEAT will be further detailed in the next section.

2.4 NEAT

NEAT (Stanley and Miikkulainen, 2002) combines the usual search for the appropriate network weights with complexification of the network structure, allowing the behaviour of evolved neural networks to become increasingly sophisticated over generations. The NEAT method consists of solutions to three fundamental challenges in evolving neural network topology, which will be briefly explained below.

i. What kind of genetic representation would allow meaningful crossover between networks with different topologies? The proposed solution is to use historical markings to line up genes with the same origin.

ii. How can a topological innovation, that may require a few generations to optimise the connection weights, be protected so that it does not disappear from the population prematurely? The proposed solution is to separate innovations into different species.

iii. How can topologies be minimised throughout evolution so the most efficient solutions will be discovered? The proposed solution is to start from a minimal structure and add nodes and connections incrementally.

2.4.1 Genetic Encoding

Evolving structure requires a flexible genetic encoding. In order to allow structures to complexify, their representations must be dynamic and expandable. Each genome

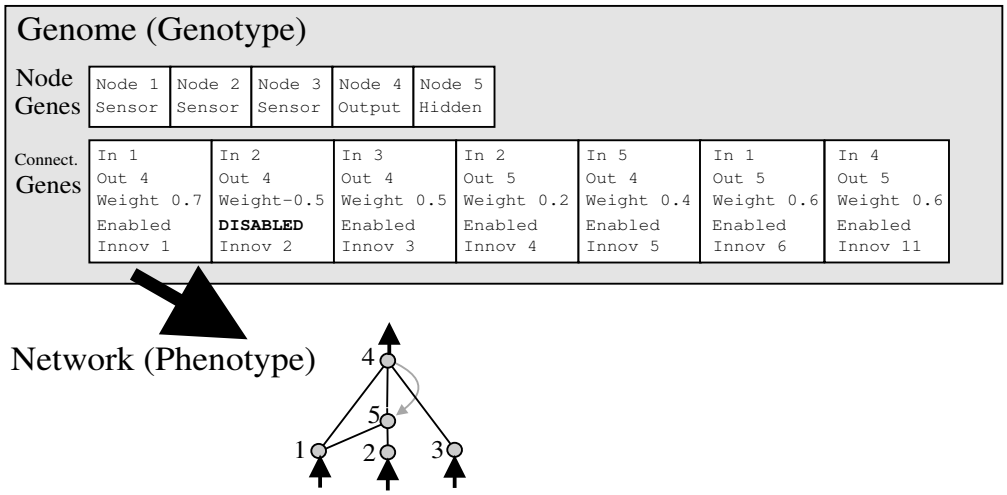

Figure 2.1: A genotype to phenotype mapping example in NEAT. [Image from (Stanley and Miikkulainen, 2002)]

in NEAT includes a list of connection genes, each one specifying the in-node, the out-node, the weight of the connection, and an innovation number, which allows finding corresponding genes during crossover. The structure of the genome and its mapping to the phenotype is showed in Figure 2.1.

Mutation in NEAT can change both connection weights and network structures. Connection weights mutate as in any neuroevolution system, with each connection either perturbed or not. Structural mutations, which form the basis of complexification, occur in two ways, either by adding a new connection between to previously unconnected nodes or by adding a new node that splits an existing connection and preserves its weight (Figure 2.2). Adding nodes in this way preserves the previous functionality but at the same time introduces a new non-linearity that provides the opportunity to elaborate those functionalities in the next generations.

Through mutation, the genomes in NEAT will gradually get larger. Genomes of varying sizes will appear, sometimes with different connections at the same positions in the genome representation. The crossover operator must be able to cope with these challenges in order to perform crossovers between different topologies, which will be explained next.

2.4.2 Tracking Genes Through Historical Markings

The historical origin of each gene can be used to determine exactly which genes match up between any individuals in the population. Two genes with the same historical origin represent the same structure, since they were both derived from the same ancestral gene at some point in the past. Thus, in order to properly align and recombine any two disparate topologies in the population, the system only needs to

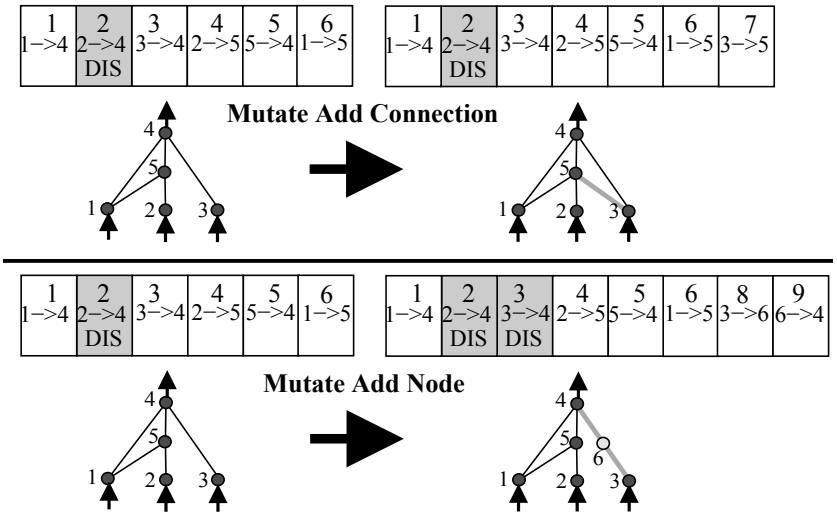

Figure 2.2: An example of a *Add Connection* mutation (top) and a *Add Node* mutation (bottom). [Image from (Stanley and Miikkulainen, 2002)]

keep track of the historical origin of each gene.

Whenever a new gene appears through structural mutation, a global innovation number is incremented and assigned to that gene. The innovation numbers, thus, represent a chronology of every gene in the population. Whenever two genomes cross over, the offspring will inherit the same innovation numbers on each gene. Thus, the historical origin of every gene is known throughout evolution.

When crossing over, the genes with the same innovation numbers are lined up (Figure 2.3). The offspring is then formed in one of two ways: In uniform crossover, matching genes are randomly chosen for the offspring genome. In blended crossover, the connection weights of matching genes are averaged. The disjoint and excess genes are inherited from the more fit parent.

Genomes of different organisations and sizes remain compatible throughout evolution, and the variable-length genome problem is essentially avoided. This methodology allows NEAT to complexify structure while different networks still remain compatible for crossover.

However, it turns out that it is difficult for a population of varying topologies to support new innovations that add structure to existing networks, because smaller structures optimise faster than larger structures, and adding nodes and connections usually initially decreases the fitness of the network, giving recently augmented structures little chance of surviving. The solution is to protect innovation by speciating the population, as explained next.

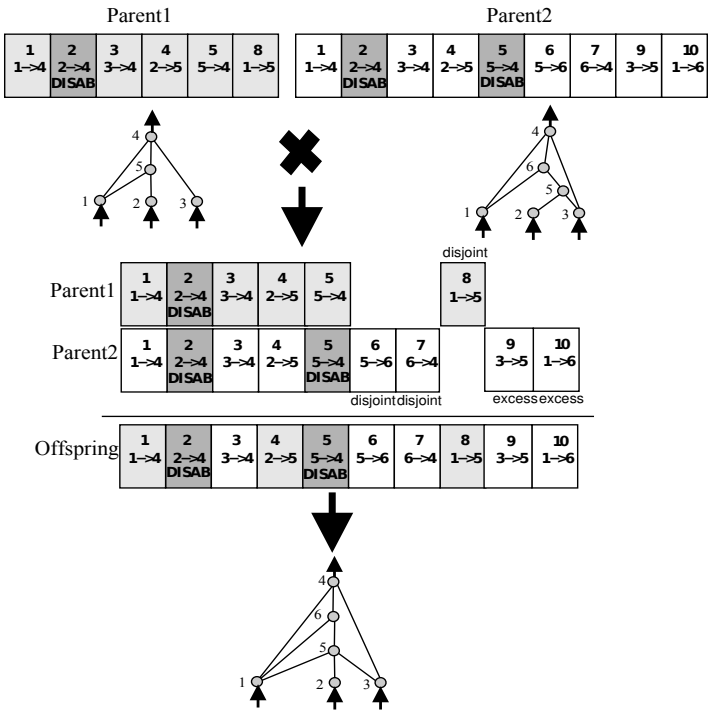

Figure 2.3: The gene alignment process, based on the innovation numbers, necessary for the crossover. [Image from (Stanley and Miikkulainen, 2002)]

2.4.3 Protecting Innovation Through Speciation

NEAT speciates the population so that individuals compete primarily within their own niches instead of with the population at large. This way, topological innovations are protected and have time to optimise their structure before they have to compete with other niches in the population.

Historical markings make it possible for the system to divide the population into species based on how similar they are topologically. The distance between two network encodings can be measured as a linear combination of the number of excess and disjoint genes, as well as the average weight differences of matching genes. When a new genome is created, it is compared to a random genome of each existing species, and if the distance between them is below a certain threshold, it is placed in that species. If the genome is not compatible with any existing species, a new one is created.

As the reproduction mechanism, NEAT uses explicit fitness sharing (Goldberg and Richardson, 1987), where organisms in the same species must share the fitness of their niche. The adjusted fitness of each organism is calculated by dividing its individual fitness score by the number of organisms in the respective species. Every species is then assigned a potentially different number of offspring in proportion to the sum of adjusted fitnesses of its member organisms. Species reproduce by

first eliminating the lowest performing members from the population. The entire population is then replaced by the offspring of the remaining individuals in each species.

2.4.4 Incremental Complexification

Other systems that evolve network topologies and weights begin evolution with a population of random topologies. In contrast, NEAT begins with a uniform population of simple networks, with no hidden nodes and fully connected, differing only in their initial random weights. Speciation protects new innovations, allowing diverse topologies to gradually accumulate over evolution. Thus, NEAT can start minimally, and grow the necessary structure over generations. New structures are introduced incrementally as structural mutations occur, and only those structures survive that are found to be useful through fitness evaluations. In this way, NEAT searches through a minimal number of weight dimensions, significantly reducing the number of generations necessary to find a solution, and ensuring that networks become no more complex than necessary.

2.5 Deception in Evolutionary Computation

> We are kept from our goal not by obstacles but by a clear path to a lesser goal.
>
> Robert Brault

Evolutionary algorithms typically measure the progress of the population according to a pre-defined fixed objective, rewarding the individuals that seem closer to that objective (Holland, 1975). While this may seem straightforward and intuitive, in fact there are some problems associated with this approach. Objective functions often suffer from the problem of local optima, dead ends in the search space that stop the population from getting better and reaching the global optima. This global optima might have been achieved if a different path in the exploration of the search space was taken (Goldberg, 1987). The problem is that the objective function does not necessarily reward the stepping stones in the search space that ultimately lead to the objective, and can actually deceive the evolution by pointing the wrong way. Objective functions with this property are called deceptive objective functions. This makes the design of good objective functions challenging, since sometimes pursuing what appears to be a reasonable objective results in an ineffective objective function.

Another issue that can arise if the objective function does not reward the intermediate steps towards the solution is the bootstrap problem (Mouret and Doncieux, 2009; Gomez and Miikkulainen, 1996). This problems occurs when the task is too

demanding to exert significant selective pressure on the population during the early stages of evolution, as all of the individuals perform poorly. As a consequence, the genetic algorithm gets trapped very early in an uninteresting region of the solution space.

One way of overcoming these issues in order to solve complex problems with the evolutionary process is described in (Gomez and Miikkulainen, 1996). In this work, the complex behaviours are learned incrementally through the gradual complexification of the task. Instead of evaluating a population on the same task throughout the course of evolution, a set of goals of increasing difficulty is defined. The population is then evaluated using those goals in sequence throughout the evolution, in an effort to ultimately achieve the final goal. A drawback of this approach is that it requires a significant amount of *a priori* knowledge about the task and a careful oversight and analysis of the evolution, in order to try to understand where the evolution gets stuck and how to devise a new sub-goal that will allow the population to overcome that difficulty. The underlying problem of local optima is also not fixed, because if these optima are pervasive, search will still likely get stuck in a dead end.

Other common approaches for mitigating deception are based on some kind of diversity maintenance technique, many of them inspired by speciation or niching in natural evolution, where the competition occurs within the same niche instead of compassing the entire population. Fitness Sharing (Goldberg and Richardson, 1987), used by NEAT, was one of the first works concerning the mitigation of deception. This method enforces competition between similar solutions so that there is pressure to find solutions in distant parts of the search space. Other similar approaches include enforcing competition among individuals with similar fitness scores (Hu et al., 2005; Hutter and Legg, 2006) and among individuals of different genetic ages (Hornby, 2006; Castelli et al., 2011). Multi-objectivisation (Deb, 2001) might also be used to avoid deception, because if a local optimum is reached in one objective, there is still hope to evolve in respect to the other objectives (Knowles et al., 2001). In (Mouret and Doncieux, 2008), incremental evolution of a robot controller is implemented as a multi-objective algorithm, where each sub-task is modelled as a different objective. This eliminates some biases of the traditional incremental approaches, like the necessity of pre-determining the order in which the tasks should be learned. However, some issues are still identified, like the limitation of multi-objective algorithms of only handling a few objectives (Mouret and Doncieux, 2008) .

While these methods for mitigating deception might aid evolution to avoid getting stuck in local optima, they leave the underlying problem untreated, namely the issue that the objective function itself might be actively misdirecting the search towards dead ends. With this issue in mind, a new radical approach for seeking ob-

jectives was recently proposed, Novelty Search (Lehman and Stanley, 2008, 2011a), in which our work will be built upon. This approach puts the objectives completely aside, and consists of seeking behavioural diversity instead of actively pursuing the objectives. Novelty Search will be further explained in the next section.

2.6 Novelty Search

Evolutionary algorithms are often applied as black-box optimisation algorithms designed to converge to a globally optimal fitness. In contrast, natural evolution diverges, creating and maintaining a wide variety of solutions to the problems of life. Novelty search (Lehman and Stanley, 2011a) is a divergent evolutionary technique, inspired by natural evolution's drive towards novelty, which strives to create novel behaviours instead of progress towards a fixed objective.

In novelty search (Lehman and Stanley, 2011a), individuals in an evolving population are selected based exclusively on how different their behaviour is when compared to the other behaviours discovered so far. Through the exploration of the behaviour space, the objective will eventually be achieved, even though it is not being actively pursued. Implementing novelty search requires little change to any evolutionary algorithm aside from replacing the fitness function with a domain dependent novelty metric. This metric quantifies how different an individual is from the other individuals with respect to their behaviour. Like the fitness function, the novelty metric must be adequate to the domain, expressing what behaviour characteristics should be measured and therefore conditioning what behaviours will be explored. The use of a novelty measure creates a constant pressure to evolve individuals with novel behaviour features, instead of maximising a fitness objective.

The novelty of a newly generated individual is computed with respect to the behaviours of an archive of past individuals and to the current population, giving a comprehensive sample of where the search has been and where it currently is. The novelty archive is representative of the previously explored behaviours. However, it does not contain all of those behaviours, in order to minimise the impact in the algorithm's computational complexity. The archive is initially empty, and behaviours are added to it if they are significantly different from the ones already there, i.e., if their novelty is above some threshold. The purpose of the archive is to allow the penalisation of future individuals that exhibit previously explored behaviours.

The novelty metric characterises how far away the new individual is from the rest of the population and its predecessors in behaviour space, determining the sparseness at any point in that space. A simple measure of sparseness at a point is the average distance to the k-nearest neighbours of that point, where k is a fixed

parameter empirically determined. Intuitively, if the average distance to a given point's nearest neighbours is large then it is in a sparse area; it is in a dense region if the average distance is small. The sparseness ρ at point x is given by:

$$\rho(x) = \frac{1}{k} \sum_{i=1}^{k} dist(x, \mu_i) \ , \tag{2.1}$$

where μ_i the ith-nearest neighbour of x with respect to the distance metric $dist$, which is a domain-dependent measure of behavioural difference between two individuals in the search space. Candidates from more sparse regions of the behaviour space thus receive higher novelty scores, guiding the search towards what is new, with no other explicit objective.

The behaviour of each individual is typically characterised by a vector of numbers. The experimenter should design the behaviour characterisation so that each vector contains aspects of the behaviour of the individual that are considered relevant to the problem that is being solved. Once the behaviour characterisation is defined, the novelty distance metric $dist$ can be defined as the distance between the behaviour vectors. A commonly used distance is the Euclidian distance between the vectors.

The behaviour characterisation can be for example the situation of the agent at the end of the trial or some measure that is sampled along the trial. For instance, when originally introduced, novelty search was demonstrated on a maze navigation task (Lehman and Stanley, 2011a), where the behaviour characterisation was a vector containing the final position (x, y) of the robot in the maze. Choosing the aspects of the behaviour that should be put in the behaviour characterisation typically requires domain knowledge, and has direct implications on the diversity of behaviours that will be synthesised by evolution. Excessively detailed behaviour characterisations can open the search space too much, and might cause the evolution to focus on evolving behaviours that are irrelevant for solving the problem. On the other hand, a too simple behaviour characterisation might be insufficient for accurately estimating the novelty of each individual, and can prevent the evolution of some types of solutions.

It is important to note that the detail of the behaviour characterisation is not necessarily correlated with the length of the behaviour vector. In the maze navigation experiments (Lehman and Stanley, 2011a), the authors expanded the behaviour characterisation to include intermediate points along the path of an individual through the maze, instead of just the final position. The authors experimented with different sampling frequencies, resulting in behaviour characterisations of different lengths, and the results showed that the performance of the evolution was largely unaffected by the length of the behaviour characterisation. Although a longer characterisation increased the dimensionality of the behaviour space, only a small portion of this space was reachable since adjacent points in a given path were highly

correlated (i.e. the agent can only move so far in the interval between samples). It was demonstrated that larger behaviour descriptions do not necessarily imply a less effective search, despite having a larger behaviour space.

Once objective-based fitness is replaced with novelty, the underlying evolutionary algorithm operates as normal, selecting the most novel individuals to reproduce. Over generations, novelty search encourages the population to spread out across the space of possible behaviours, eventually encountering individuals that solve the given problem, even though progress towards the solution is not directly rewarded. In fact, there have been several successful applications of novelty search. The most notable applications include the evolution of adaptative neural networks (Soltoggio and Jones, 2009); genetic programming (Lehman and Stanley, 2010b); evolution strategies (Cuccu et al., 2011); robot body-brain co-evolution (Krcah, 2010); biped robot control (Lehman and Stanley, 2011a); and robot navigation in deceptive mazes (Lehman and Stanley, 2008; Mouret, 2011).

Although novelty search does not require a specific underlying neuroevolution method, as seen in (Cuccu and Gomez, 2011; Mouret, 2011), authors mention that it benefits from methods that gradually complexificate the neural network. This complexification imposes some order in the search, because simple networks are only able to model relatively simple input-output mappings, which typically translates in simple behaviours, thus temporarily reducing the search space. As the networks get more complex, more complex behaviours can arise, gradually opening the behaviour search space. This order also ensures that the evolved solutions will be as simple as possible, because the simpler behaviours in the search space will be depleted before advancing to more complex ones. Novelty Search was initially implemented and tested over NEAT (Stanley and Miikkulainen, 2002), which employs a complexification mechanism, as we have described in section 2.4.

2.7 The Problem of Vast Behaviour Spaces

One limitation of novelty search is that if the behavioural space is too big or even infinite, then the search might actually get lost in unfruitful regions, and not explore the regions that will ultimately lead to the objective (Lehman and Stanley, 2010a; Cuccu and Gomez, 2011). As such, several works have been built upon novelty search in an attempt to mitigate this problem, by reuniting novelty search with the traditional objective-oriented evolutionary approach. In (Mouret and Doncieux, 2012), it is presented a comprehensive empirical study where a large number of evolutionary techniques inspired on novelty search are compared.

To address the problem of vast behaviour spaces, the authors of novelty search proposed *minimal criteria novelty search* (MCNS) (Lehman and Stanley, 2010a).

MCNS is an extension of novelty search where individuals must meet some domain-dependent minimal criteria to be selected for reproduction, thus restricting the behaviour search space. Cuccu and Gomez (Cuccu and Gomez, 2011) proposed to base selection on a linear blend of novelty score and fitness score (henceforth referred to as *linear blend*). Mouret (Mouret, 2011) proposed novelty-based multi-objectivisation, which is a Pareto-based multi-objective evolutionary algorithm. The novelty objective is added to the task objective in a multi-objective optimisation. These approaches will be detailed next.

2.7.1 Minimal Criteria Novelty Search

Minimal criteria novelty search (Lehman and Stanley, 2010a) is an extension of NS that relies on a task-dependent minimal criteria. In MCNS, if an individual satisfies minimal criteria, it is assigned its normal novelty score, as computed in the original novelty search algorithm. If an individual does not satisfy the minimal criteria, it is assigned a score of zero and is only considered for reproduction if there are no other individuals in the population that meet the criteria. In (Lehman and Stanley, 2010a), the authors applied MCNS to two maze navigation tasks, both with very large behaviour spaces. The results show that MCNS can mitigate the problem of massive behaviour spaces, and in such cases, it could reach the goals more consistently than pure novelty search. It is argued that this extension can make novelty search more effective without losing its open-ended character.

However, MCNS suffers from two major drawbacks. First, the choice of minimal criteria in a particular domain requires careful consideration and domain knowledge, since it adds significant restrictions to the search space. Constraining the search space too much can be prejudicial to the effectiveness of the evolution. Second, if no individuals are found that meet the minimal criteria, search is effectively random. Therefore, it may be necessary to seed MCNS with a genome specifically evolved to meet the criteria, in case it is unlikely to generate individuals satisfying them in the initial population.

2.7.2 Linear Blend of Novelty and Fitness

Cuccu and Gomez (Cuccu and Gomez, 2011) proposed a linear blend of novelty and fitness score, as a form of sustaining diversity and improving the performance of standard objective search. Their approach constrains and directs the search in the behaviour space. Each individual i is evaluated to measure both fitness, $\mathit{fit}(i)$, and novelty, $\mathit{nov}(i)$, which after being normalised (Eq. 2.2) are combined according to Eq. 2.3.

$$\overline{fit}(i) = \frac{fit(i) - fit_{min}}{fit_{max} - fit_{min}}, \quad \overline{nov}(i) = \frac{nov(i) - nov_{min}}{nov_{max} - nov_{min}} \ , \tag{2.2}$$

$$score(i) = (1 - \rho) \cdot \overline{fit}(i) + \rho \cdot \overline{nov}(i) \ . \tag{2.3}$$

The parameter ρ controls the relative weight of fitness and novelty, and must be specified by the experimenter through trial and error. fit_{min} and nov_{min} are the lowest fitness and novelty scores in the current population, and fit_{max} and nov_{max} are the corresponding highest scores. In (Cuccu and Gomez, 2011), the linear blend method was applied to the deceptive Tartarus problem, with a large behaviour space, and performance was compared for different values of ρ. The best results were produced with values of ρ between 0.4 and 0.9, reaching the peak performance with $\rho = 0.8$, with the novelty score having much more importance in the blend than the fitness score. The fitness of the solutions achieved with this method was notably higher when compared to both pure novelty search and pure fitness-based evolution.

2.7.3 Multi-objectivisation

Another way of combining novelty and fitness is described in (Mouret, 2011), following a multi-objectivisation approach. In this work, the novelty measure and the fitness value are considered distinct objectives, and are combined via a Pareto-based Multi-objective Evolutionary Algorithm (MOEA) (Deb, 2001). The Pareto dominance relation ensures that novel but inefficient candidate solutions will be selected, but also that efficient but less novel ones will be considered equally valuable. The technique was applied to a deceptive maze navigation problem. Compared with pure novelty search, the multi-objectivisation obtained only slightly better results.

A similar approach is followed in (Lehman and Stanley, 2011b), where it is used to evolve robot morphologies. However, this work introduces local competition, meaning that each individual only competes with individuals of similar morphologies. The local competition combined with the Pareto multi-objectivisation, proves to be especially valuable in finding a diversity of robots with very distinct morphologies. This approach does not seek to find the singular most optimally fit solution, but rather a wide variety of solutions well-adapted to solve the problem at hand.

2.8 Generic Behaviour Measures

While previously introduced works show that behavioural diversity can contribute with substantial improvements to evolutionary robotics, the behavioural diversity

measures rely on problem-specific descriptions of behaviours. Designing these measures requires some expert knowledge about the task one is trying to solve. To counter this issue, a different approach to behavioural diversity was taken in (Doncieux and Mouret, 2010). This work introduces generic behavioural similarities that rely only on sensori-motor values, which are problem independent. The behaviour description is composed by the set of sensor and effector data sampled over time, and several similarity measures are proposed, such as Hamming distance; measure based on Fourier coefficients; state count; and trajectory similarity. It is shown that these measures can be as effective as domain-dependent behavioural descriptions, and some of these generic measures can even model the human perception of behavioural differences. However, the authors note that choosing which similarity measure to use remains an open question, and this choice has a great impact in the performance of the evolution. In (Gomez, 2009) is suggested that Normalized Compressed Distance (NCD) can be a good measure to assess behavioural difference, as it does not require sequences of the same length and explores only algorithmic regularities in the sequences.

The empirical study in (Mouret and Doncieux, 2012) also compares task-specific behaviour distance measures with generic similarity measures, using three different evolutionary robotics tasks. The results show that task-specific measures are in most cases better than generic measures, however, it is described one experiment where task-specific measures are significantly worse. This result highlights one of the main pitfalls of task-specific measures: when correctly chosen, they may be very efficient, but their design is not always obvious.

The work described in this thesis only deals with task-specific behaviour distances. While generic measures seem a promising approach, they are not much studied in the literature. Since the focus of our work is the application of novelty search to the domain of evolutionary swarm robotics, we chose to work only with task-specific behaviour distances, as they are described in the original and most popular novelty search algorithm.

In this chapter, we have reviewed the domain of evolutionary swarm robotics and its open challenges. We also presented novelty search, its advantages and downfalls, and the previous applications of this algorithm. In the next chapter, we will experiment with the application of novelty search to the domain of swarm robotics, to study the potential of this approach in overcoming some challenges of evolutionary robotics, like the problem of deception and bootstrapping. To the best of our knowledge, novelty search has not been applied to swarm robotics before.

Chapter 3

Novelty Search in Evolutionary Swarm Robotics

In this chapter, we will apply the original novelty search algorithm (Lehman and Stanley, 2011a) to the evolution of controllers for a swarm of robots. In our experiments, we use two different robotics tasks: an aggregation task, where robots start from random initial positions of a bounded arena, and must form a single aggregate in the end; and an energy management task, where the robots of the swarm must share a charging station in order to avoid the depletion of their batteries. We compare the performance of novelty search with fitness-based evolution, study the dynamics of novelty search in the exploration of the behaviour space, and address the challenge of devising behaviour distance metrics for the domain of swarm robotics.

3.1 Aggregation Experiments

In this section, we apply novelty search to the aggregation task and compare it with fitness-based evolution. Three experiments were performed using different novelty measures: one highly correlated with the fitness function, an alternative measure only weakly correlated, and finally a combination of the two. In each experiment, the performance of novelty search was compared to the performance of traditional fitness-based evolution. NEAT with random selection is used as a baseline for performance comparisons.

3.1.1 Aggregation Related Work

Several works describe the evolution of aggregation behaviours in swarms of robots, where neural networks with fixed topologies are evolved via evolutionary algorithms guided by fitness. Baldassarre et al. (2003) successfully evolved controllers for a swarm of robots to aggregate and move towards a light source in a clustered formation. Trianni et al. (2003) describe the evolution of a swarm of simple robots to

perform aggregation in a square arena. In those experiment, two different behaviours were evolved: a *static clustering* which forms compact and stable aggregates and a *dynamic clustering* which creates loose but moving aggregates. Bahgeçi and Şahin (2005) used a similar experimental setup as (Trianni et al., 2003), and studied how some parameters of the evolutionary method affect the performance and the scalability of behaviours in swarm robotic systems.

In these studies, the robots used directional sound sensors and sound signalling to identify other robots in the environment. Sound signalling enabled robots to follow sound gradients in order to aggregate. In fact, these works show that neural networks without any hidden neurons are sufficient to successfully solve the task. In our work, we make the aggregation task more challenging: we remove the sound gradient, decrease the range of the sensors, and increase the size of the arena. These modifications increase the difficulty of the task and may require quite different strategies for aggregation because it is harder for the robots to find each other (Soysal et al., 2007).

3.1.2 Experimental Setup

The experiments were conducted with a framework developed in the context of this thesis, described in Appendix A. The framework is based on the Simbad 3d Robot Simulator (Hugues and Bredeche, 2006) for the robotic simulations, and on NEAT4J[1] for the implementation of NEAT.

The environment is a 5 m by 5 m square arena bounded by walls. The robots are modelled based on the e-puck educational robot Mondada et al. (2009), but do not strictly follow its specification. Each simulated robot has 8 IR sensors evenly distributed around its chassis for the detection of obstacles (walls or other robots) within a range of 10 cm, and 8 IR sensors dedicated to the detection of other robots within 25 cm range. An additional sensor calculates the percentage of nearby robots, relative to the size of the swarm, within a radius of 25 cm. This implies that each robot has to know *a priori* the full swarm size. In this thesis, the authenticity of the robots is not our primary concern, since the focus is the comparison between various evolutionary methods, that are used over the same experimental setup. However, the obstacles sensors could be implemented with active IR sensors, the robots sensors with passive IR sensors, and the count sensor could be implemented with short-range communication (Correll and Martinoli, 2007). Figure 3.1 depicts the robot and sensor setup.

To evaluate each controller, 10 simulations are run with it, varying the number of robots and their starting positions and orientations. The swarm is homogeneous and the group size varies from 3 to 10, with each controller being run at least once with

[1]NeuroEvolution for Augmenting Topologies For Java – `http://neat4j.sourceforge.net`

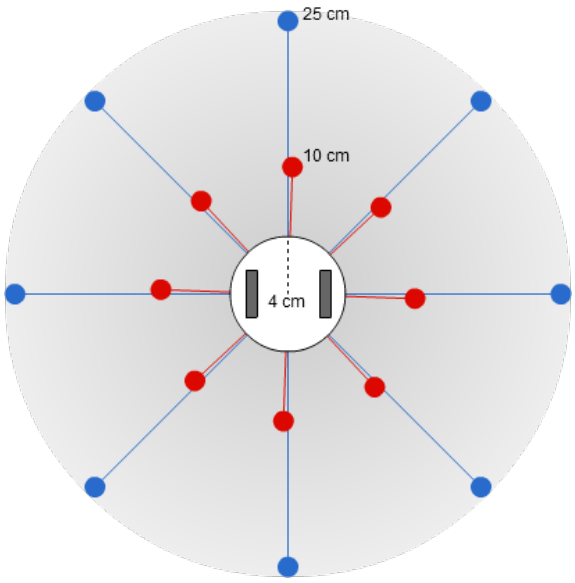

Figure 3.1: The model of the robot used in the aggregation experiments. The robot itself is depicted in white, its wheels in dark grey, the IR obstacle sensors in red, the passive IR sensors for detection of other robots in blue, and the grey circle represents the range of the robot counter sensor.

every group size. The starting positions and orientations are random but ensure a minimum distance of 30 cm between the robots. Each simulation lasts for 500 s of simulated time, corresponding to 5000 simulation steps (10 updates per second).

3.1.3 Configuration of the Evolutionary Algorithms

For fitness-based evolution, we used the default NEAT implementation provided by the NEAT4J library (see Section 2.4 for an explanation of NEAT). Random evolution was also implemented with NEAT, but in each generation random fitness scores are assigned to the individuals of the population. For novelty search, we extended the same NEAT implementation following the description and parameters in (Lehman and Stanley, 2011a), with a k value of 15 and a dynamic archive threshold (Lehman and Stanley, 2010a). This dynamic threshold ensures a constant and reasonable flow of individuals to the archive, at an average rate of 3 individuals per generation. If in one generation more than 3 behaviours are added to the archive, the archive threshold is raised by 10%, if no behaviour is added in one generation, the archive threshold is lowered by 10%.

The NEAT parameters were the same in both evolutionary methods: recurrent links are allowed, the crossover rate was 25%, the mutation rate 10%, the population size 200, and each evolution runs for 250 generations. The rest of the parameters were the default of the NEAT4J implementation. Table 3.1 summarises the NEAT parameters used in our experiments.

Table 3.1: The NEAT parameters used in the experiments. Only the parameters in italic were modified, the rest are the default in the implementation. See (Stanley and Miikkulainen, 2002) and `http://neat4j.sourceforge.net/documents/config.html` for a detailed explanation of the parameters.

Parameter	**Value**
Probability mutation	0.1
Probability crossover	0.25
Probability add link	0.05
Probability add node	0.03
Probability mutate bias	0.3
Population size	200
Recurrency allowed	true
Max weight perturbation	0.5
Max bias perturbation	0.1
Compatibility threshold	0.5
Compatibility change	0.05
Target species count	10
Max generations with no improvement	15
Survival threshold	0.2
Excess coefficient	1
Disjoint coefficient	1
Weight coefficient	0.4

The fitness function that evaluates the performance of the swarm in each simulation is based on the average distance to the centre of mass, also used in (Trianni et al., 2003). The centre of mass of the swarm is given by:

$$\mathbf{R} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{r}_i \ , \tag{3.1}$$

where N is the number of robots and $\mathbf{r}_i$ is the position of the robot i.

The average distance to the centre of mass is sampled throughout the simulation at regular intervals of 10 s. The samples are then combined in a single fitness value using a weighted average, with linearly more weight towards the end of the simulation. The purpose of the weighted average is to express that the robots should aggregate as soon as possible but that it is relatively more important that they *remain* aggregated at the end of the simulation. The fitness F of a simulation with T time steps and N robots is defined as:

$$F = 1 - \frac{1}{\sum \frac{t}{T}} \sum_{t=1}^{T} \frac{t}{T} \sum_{i=1}^{N} \frac{dist(\mathbf{R}_t, \mathbf{r}_{i_t})}{N} \ , \tag{3.2}$$

where $\mathbf{R}_t$ is the centre of mass at each instant, and $\mathbf{r}_{i_t}$ is the position of each robot. The distance values are normalized to [0,1].

The fitness scores obtained in each of the 10 simulations are combined in a single value using the harmonic mean. The harmonic mean of a list of numbers tends strongly towards the smaller elements of the list, mitigating the impact of large outliers and aggravating the impact of small ones. The choice of this mean is supported by the results reported in (Bahgeçi and Şahin, 2005), where it is argued that pessimistic combination functions (that give more importance to the lower values) are preferable.

As mentioned in Section 2.6, the novelty measure characterises the distance between one controller and the others in behaviour space. To calculate the distance in behaviour space, we use the Euclidian distance between vectors that represent the level of aggregation along time. These vectors are built by measuring behaviour features at regular intervals throughout the simulation (every 10 s). We devised three ways of measuring the group behaviour, which will be presented in the next sections. As 10 simulations are conducted to evaluate each controller, its behaviour vector is the average of the vectors obtained in each of the simulations. In order to compare novelty search with the fitness-based evolution, the controllers evolved by novelty search were also evaluated with the fitness function F. It is important to note that the fitness scores did not have any influence in the novelty search experiments, and the novelty scores did not influence fitness-based evolution.

3.1.4 The First Experiment

The first behaviour measure uses a metric similar to the fitness function; a vector is built with the average distance to centre of mass sampled throughout the simulation. Unlike the fitness function, the behaviour measure does not have any bias towards the end of the simulation. Considering a simulation with N robots and T temporal samples, the behaviour vector $\mathbf{b_{cm}}$ that characterises a controller is given by:

$$\mathbf{b_{cm}} = \frac{1}{N}\left[\sum_{i=1}^{N} dist(\mathbf{R}_1, \mathbf{r}_{i_1}), \cdots, \sum_{i=1}^{N} dist(\mathbf{R}_T, \mathbf{r}_{i_T})\right] \quad . \tag{3.3}$$

In our experiments, the sampling rate was 10 s and the simulation time 500 s, resulting in a behaviour vector of length 50.

The fitness scores of the highest scoring individuals evolved using novelty search and fitness driven evolution are depicted in figure 3.2. The statistic used for a fair comparison is the best fitness score found so far, from the start of the evolution until the current generation. Since novelty search does not explore the search space following the fitness gradient, using the statistic of the best fitness found in each generation (one of the most common measures) would not result in a fair comparison. The evolution was tested with more generations but there was no perceivable change in the fitness values after the 250th generation. Individuals with fitness value over

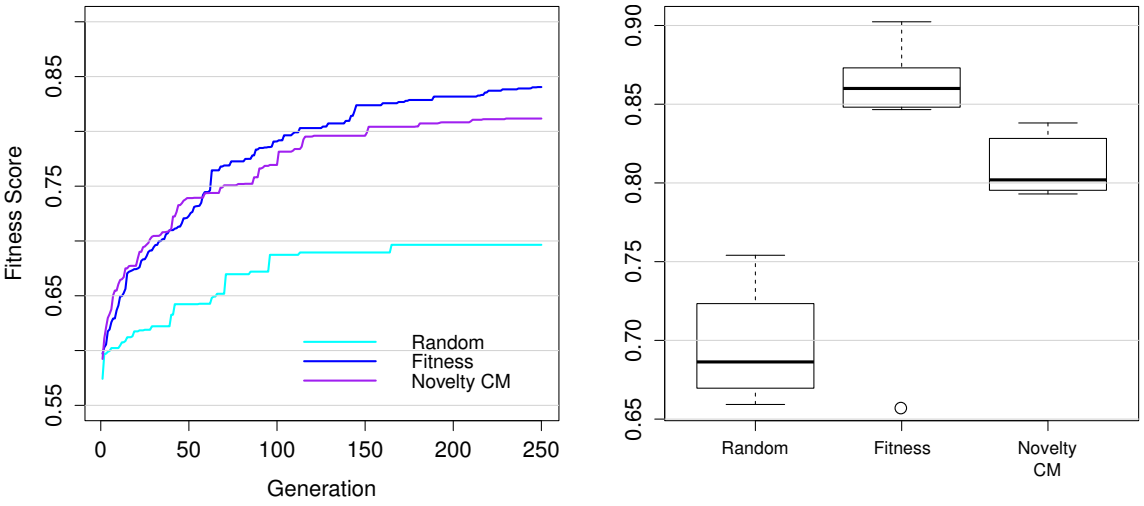

Figure 3.2: Fitness trajectories of novelty search with the centre of mass novelty measure, fitness-based evolution, and evolution with random selection. Left: Fitness value of the best individual found so far in each generation, with each evolutionary method. The values are averaged over 10 evolutionary runs for each method. Right: Boxplots with the best fitness score found in each evolutionary run, for each method.

0.8 are reasonable solutions to the task. Note that although the minimum fitness value is ≈ 0.05, the best fitness on an initial random population is on average ≈ 0.6. This happens not because the task is too easy, but because in this task, the robots doing nothing at all and staying in their initial positions (which is one of the most simple input-output mappings) can be more beneficial than many other complex behaviours.

There is not a big difference between the fitness of the controllers evolved with novelty search and fitness-based evolution. The difference between the fitness trajectories of both methods is not statistically significant (Student's t-test with p-value < 0.05). Both methods are significantly better than the random evolution (p-value < 0.01). The boxplots show that novelty search is fairly consistent in achieving high fitness scores. Fitness-based evolution also shows a good consistency, but there was one evolutionary run (outlier in the boxplot) where the best fitness score was very low, probably caused by premature convergence of the population.

If we analyse the behaviours of the best controllers evolved by both methods, significant differences are found, despite the similar fitness values. In the fitness-based evolution, the highest scoring controllers were always very similar, displaying only one distinctive behaviour: the robots explore the environment in large circles, and form static clusters when they encounter one another. If the cluster size is less than half of the swarm size, the robots abandon it after a while and start exploring again. The best behaviour found by the random evolution consisted of robots navigating in circles in the arena, and when two collide, they stay together

in that place. This typically results in bad fitness values because multiple clusters are formed, often far from each other.

Novelty search, on the other hand, found several distinct high-scoring controllers that could perform the aggregation task. Each evolutionary run of novelty search could evolve several different solutions, finding many (sometimes all) of the solutions described next and variants of them. Figure 3.3 also illustrates the following behaviours.

1. The robots go straight forward until they hit the wall, and then, depending on the impact angle, they stay there for a while or start moving along the wall until they find other robots. When they do, they stop and form an aggregate there. If no robot stays stopped near the wall long enough, this behaviour drastically fails, as all robots start moving alone along the wall.

2. Similar to (1), but when they meet each other they continue to follow the wall until they hit a corner, aggregating there.

3. Similar to the best behaviour evolved by fitness-based evolution, but without splitting the small clusters. This naturally results in a worse solution, since more than one small aggregates can be formed.

4. Similar to (3), but navigating in the environment only in straight trajectories instead of curves. Whenever a robot hits a wall, it rebounds with an angle of 90^{o}.

It is important to note that none of these behaviours (including the best solution evolved by fitness-based evolution) are perfect, failing frequently by forming more than one aggregate. The success rate seems to be highly dependent on the size of the swarm. The solutions tend to fail more frequently as the swarm size approaches either the maximum (10) or the minimum size (3).

The main difference between the behaviours was that novelty search evolved controllers that exploited the wall to achieve better solutions, while in the fitness-based evolution robots always avoided navigating near the walls. It is interesting to note that none of the related works in evolutionary robotics reports aggregation behaviours that exploit the walls of the arena. However, this behaviour represents a promising solution, since it can even be found in biological systems, such as in groups of cockroaches (Jeanson et al., 2005).

Our hypothesis is that learning to navigate along the walls requires going against the fitness gradient. If the robots go towards the walls, they will often end up in different ones, and staying there will result in a low fitness because the centre of mass will be in the centre of the arena, far from the robots. On the other hand,

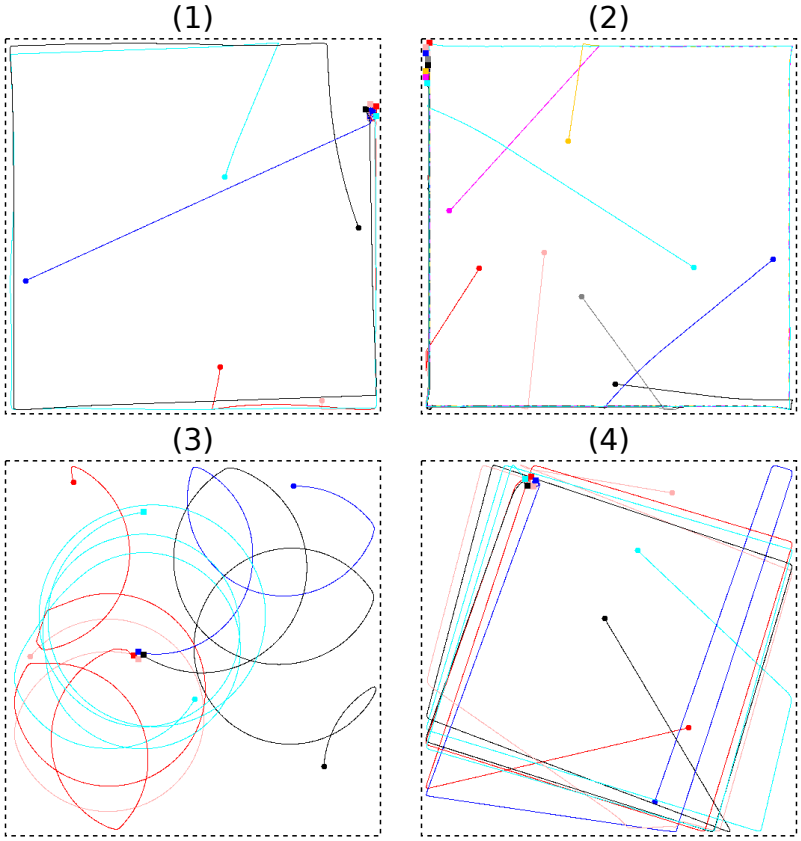

Figure 3.3: Some of the solutions found with novelty search with the centre of mass behaviour measure. Each line represents the trajectory of the robot throughout the simulation. The circle depicts the initial position of the robot and the square its final position.

avoiding the walls results in better fitness because they will be on average closer to the centre. If the fitness evolution misses the stepping stone of being close to the walls, it will hardly be able to reach behaviours that require the use of walls to achieve aggregation. A good analogy that explains this problem is given in (Lehman and Stanley, 2011a):

> "Consider fingers stuck within a Chinese finger trap. While the goal is to free one's fingers, performing the most direct action of pulling them apart yields no progress. Rather, the necessary precursor to solving the trap is to push one's fingers together, which seems to entrap them more severely. In this way, the trap is deceptive because one must seemingly move farther from the goal to ever have the hope of reaching it."

To confirm our hypothesis, we analysed the behaviour space explored in novelty search and in fitness evolution. To facilitate this analysis, all the individuals evolved in fitness evolution were also evaluated with the same behaviour measure that was used in novelty search. Since each behaviour description is a long vector, we applied

a dimensionality reduction method in order to visualise the behaviour space. We used a Kohonen self-organising map (Kohonen, 1990), a type of neural network trained using unsupervised learning to produce a two-dimensional discretisation of the input space of the training samples, preserving the topological relations.

The Kohonen map was trained with the behaviours found both in novelty search and in fitness evolution. However, since the number of evolved individuals was too large (200 individuals per generation, 250 generations, 10 evolutionary runs, 2 evolutionary methods, totalling 1 million individuals), it was computationally infeasible to build the Kohonen map with all the individuals. To overcome this limitation, it was taken a random sample of 100.000 individuals from each evolutionary method, and then the map was built with these 200.000 individuals. Each random sample of individuals was then mapped individually to the trained map. The resulting maps can be seen in Figure 3.4.

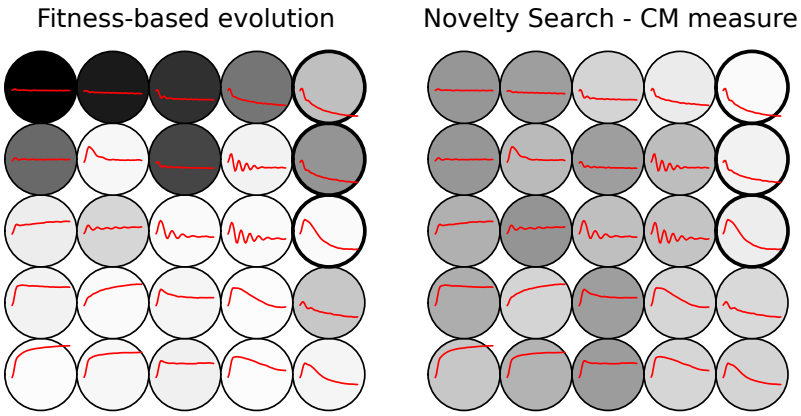

Figure 3.4: Kohonen maps representing the explored behaviour space in fitness-based evolution (left) and in novelty search with the centre of mass measure (right). Each circle is a neuron that is characterised by the vector depicted by the line inside (the average distance to the centre of mass over time). Each behaviour vector is mapped to the most similar neuron. The darker the background of a neuron is, the more behaviours were mapped to it. The behaviour patterns associated with higher fitness scores are highlighted with a bold circle.

As it can be seen in the maps, the fitness-based evolution avoids the zones where the average distance to the centre of mass rises beyond the initial value, preventing the evolution of good solutions that might require traits found only in those behaviour zones. The evolution is much more focused in behaviours that express a monotonic fall of the average distance to the centre of mass, which is consistent with the observable performances of the best controllers. This is an important result because it demonstrates that the fitness function is preventing the evolution of certain types of solutions. On the other hand, novelty search is not subject to this fitness pressure, and can therefore explore and discover a wide range

of solutions to the task.

In our version of the aggregation task, the convergence of fitness-based evolution to one type of solutions does not have harmful consequences to the evolution, since the fitness function is not deceptive and can lead the search towards other good solutions. However, it is clear that there is one class of solutions that is being hindered by the fitness function. If that class of solutions was essential for the solution of the problem, fitness-based evolution would have most likely failed. In Section 3.2, we will show an example of the harmful consequences of premature convergence.

3.1.5 The Alternative Novelty Measure

We devised a new behaviour description, based on the metric used in (Bahgeçi and Şahin, 2005), in order to determine how the novelty measure influences the evolved solutions. The new description consists of measuring the number of robot clusters along the simulation. Two robots belong to the same cluster if the distance between them is less than 30 cm. Applying this iteratively we can obtain the number of clusters. The number of samples was the same as in our previous experiments (50). The behaviour vector $\mathbf{b_{cl}}$ is described by:

$$\mathbf{b_{cl}} = \frac{1}{N}\left[clustersCount(1), \cdots, clustersCount(T)\right] \ . \tag{3.4}$$

With this new measure, the best fitness score found in each evolutionary run was on average lower than novelty search with the centre of mass behaviour measure, but an insufficient number of trials were performed to claim statistical significance beyond $p < 0.1$. The fitness trajectories are depicted in Figure 3.5. However, the boxplots show that novelty search with the centre of mass novelty measure achieves high fitness scores much more consistently than novelty search with the number of clusters measure. The explanation for this will be presented ahead. Analysing the evolved behaviours, it is possible to identify significant differences, when compared to the behaviour evolved with the centre of mass measure. The following distinct solutions were evolved (Figure 3.6):

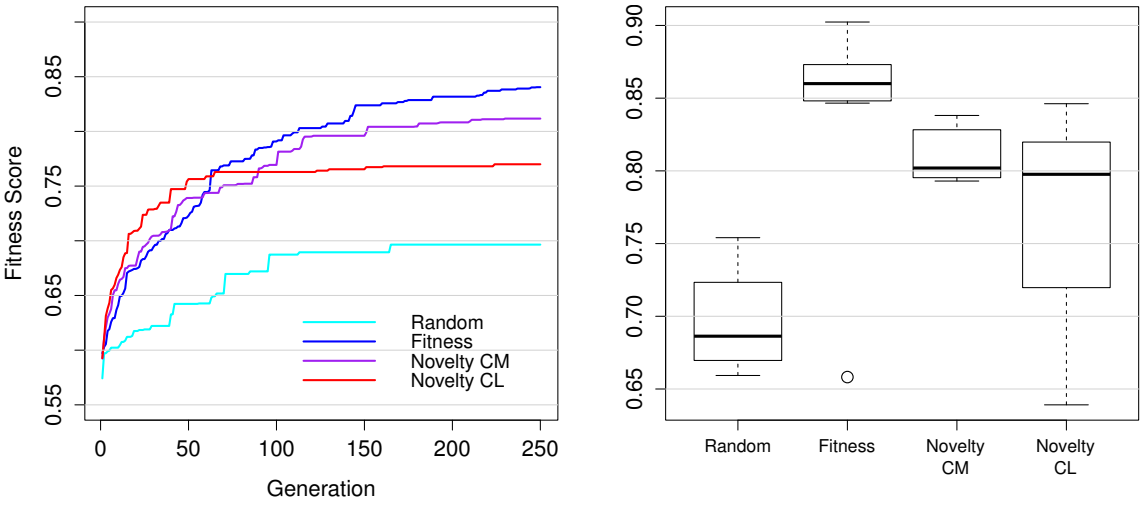

Figure 3.5: Fitness trajectory of novelty search with the number of clusters novelty measure. Left: Fitness trajectory of the best individual found so far in each generation, with each method. The values are averaged over 10 evolutionary runs for each experiment. Right: Boxplots with the best fitness score found in each evolutionary run, for each method.

1. Each robot goes towards walls, navigates along it, and when it finds another robot, they form a single file, keeping a fixed distance. The file keeps moving along the walls of the arena. This fails frequently, because the robots all move at the same speed, and as such they may never find one another. Also, the fitness score is not good when the swarm is big, since a long single file has a high average distance to the centre of mass.

2. The robots navigate in circles in the environment, forming a static cluster when they sense each other. This often results in more than one aggregate, and consequently, a low fitness score.

3. Similar to (2), but they randomly abandon their respective clusters, which partially mitigates the problem of forming many small static clusters. However, it can create another undesirable situation, where there is a single big cluster and one or two robots outside the cluster wandering in the arena.

4. The robots navigate in circles, and when two robots meet at some distance, one tries to follow the other. When robots collide, they form a cluster and remain aggregated.

Most behaviours were quite different from the ones found in the previous experiment. The reason the previous experiment did not find these behaviours (and vice-versa) is conflation (see Lehman and Stanley (2011a)). Conflation occurs when individuals with distinct observable behaviours have very similar behaviour

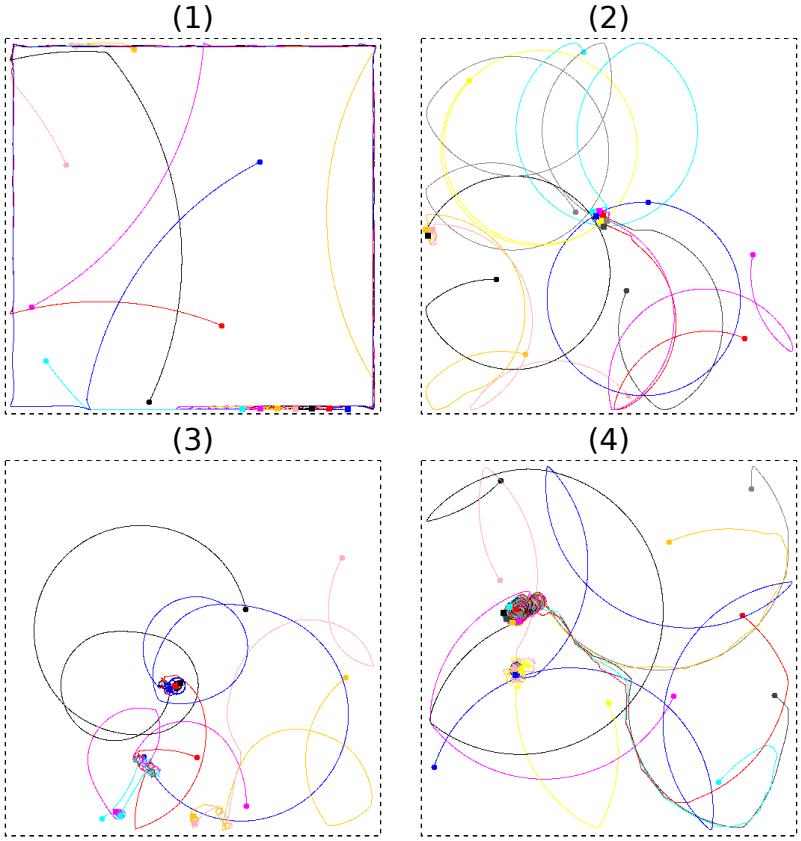

Figure 3.6: Some of the solutions found with novelty search with the number of clusters behaviour measure. Each line represents the trajectory of the robot throughout the simulation. The circle depicts the initial position of the robot and the square its final position.

descriptions. The consequence is that an individual with a distinct observable behaviour might not be considered novel by the novelty measure, thus eventually disappearing from the population. Conflation can represent both an advantage because it reduces the search space, and a disadvantage, when different successful solutions or important stepping stones are dismissed. In our experiments, what happens is that the centre of mass novelty measure is conflating some solutions that are not conflated in the clusters measure and vice-versa, thus evolving different solutions in both the experiments.

Two examples of behaviours that can be conflated are shown in Figure 3.7. When the centre of mass measure is used, for example, the clustering of the robots is irrelevant. The search will therefore avoid behaviours that have an already explored centre of mass progression but differ in the clustering of the robots, possibly by-passing interesting solutions. This effect can also be seen in the evolved behaviours: with the centre of mass measure, there were more solutions that exploited the use of the walls, because navigating near them has a great impact in that novelty measure; while with the number of clusters measure, the solutions focused on the interactions

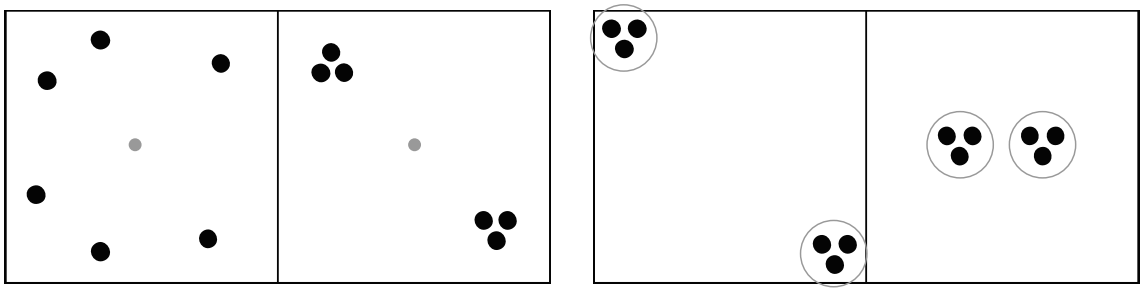

Figure 3.7: An illustration of conflation in the centre of mass measure (left) and in the number of clusters measure (right). In both cases, if the robots evolved from the left configuration to the right, that change would not be captured by the respective behaviour description, despite potentially being relevant.

between the agents and clusters, including following each other and leaving the cluster.

Conflation is also responsible for the lower fitness scores obtained with the number of clusters novelty measure. Individuals with the same behaviour characterisation can display different observable behaviours and can have very distinct fitness scores. The consequence is that solutions that are novel are not being rewarded as such, regardless of having a different fitness score of the other individuals in the population (or the archive) with a similar behaviour characterisation. Consider for instance the case in Figure 3.7 – right. If that was the final position of the robots, the first sample would have a good fitness score, while the second sample would not, despite having the same number of clusters.

The consequence is that some evolutionary runs might achieve high fitness scores, while others might not, depending on the order good solutions are evolved. When a good and intuitively novel solution appears, its novelty score still depends on the behaviour patterns previously explored. If the originality of that good solution is captured by the novelty measure, it will receive a high novelty score and it will be further explored. Otherwise, if the novelty measure fails to reflect the originality of the solution (i.e. the solution is conflated), it will receive a low novelty score, and a promising solution for the task might get discarded. The outcome of this undesirable conflation is reflected in the boxplot in Figure 3.5, where it is shown that novelty search with the number of clusters measure is fairly inconsistent in achieving high fitness scores. The centre of mass behaviour measure is less prone to suffer from this effect, since the behaviour characterisation is closely related to the fitness score of the individuals.

3.1.6 Combining novelty measures

In order to reduce conflation, we setup a new experiment with a richer behaviour description, by combining the novelty measures proposed in the two previous experiments. To combine the two behaviour descriptions presented before in Equations 3.3 and 3.4, we simply concatenate the two vectors. But as the novelty measure is based on the Euclidean distance between the vectors, caution must be displayed to ensure that both components have similar contributions to this distance. Namely, we want the vectors to have the same length and the items in the vectors to have the same range, which can be achieved by normalising each of the components. Note that both b_{cm} and b_{cl} had the same length (50) and that each element the vectors ranged from 0 to 1. The new behaviour description $\mathbf{b_{comb}}$ is thus defined as:

$$\mathbf{b_{comb}} = (\mathbf{b_{cm}}, \mathbf{b_{cl}}) \ . \tag{3.5}$$

The fitness performance of the search with this new measure was improved, evolving individuals with high fitness scores much sooner than in the other experiments, as seen in Figure 3.8. The fitness values in novelty search were significantly higher than fitness-based evolution until generation 100. After that, the difference is not statistically significant. Novelty search with the combined measure is also fairly consistent in achieving high fitness scores. This performance improvement is justified by the reduction of conflation. Remind that if novelty search evolves individuals that have a behaviour characterisation similar to one already present in the population or the archive, they will receive a low novelty score, regardless of being actually different and having a distinct fitness score. The low novelty scores can hinder the exploration of such solutions, negatively affecting the performance of the search. With a more detailed behaviour characterisation, the novelty score of the individuals could be measured with more accuracy, reducing conflation. This caused the evolution to reach good solutions more consistently, as there is less chance that those solutions are conflated.

It is also interesting to look at the explored behaviour space (Figure 3.9). We can see that there was a greater diversity of solutions, exploring many combinations of the progression of the number of clusters and the distance to the centre of mass. Novelty search explored the behaviour space much more uniformly, while fitness-based evolution spent much time in rather uninteresting behaviour zones (bottom left). On the other hand, novelty search did spend less time exploring the higher fitness behaviour zones (top left). Observing some of the best controllers in action, we notice that this combined measure seems to have evolved all the successful behaviours that were generated using the previous two measures independently.

To understand why novelty search with the combined measure was faster than fitness-based evolution in finding good individuals, we evaluated the network com-

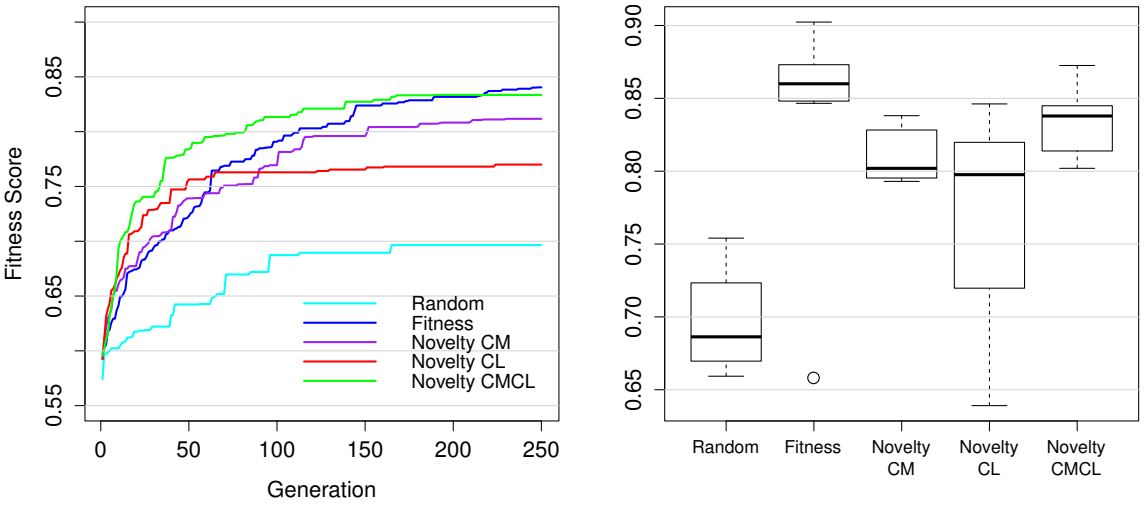

Figure 3.8: Fitness trajectory of novelty search with the combined novelty measure. Left: Fitness value of the best individual found so far in each generation. The values are averaged over 10 evolutionary runs for each experiment. Right: Boxplots with the best fitness score found in each evolutionary run, for each method.

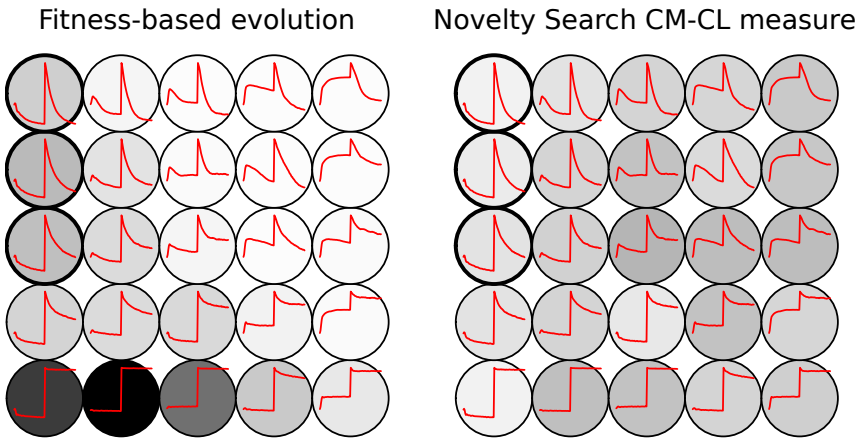

Figure 3.9: The explored behaviour space in novelty search with the combined novelty measure and in the fitness-based evolution. In each neuron, the left half is the number of clusters measure and the right half is the centre of mass. The darker the neuron background is, the more behaviours were mapped to it. Neurons with the best behaviours have a bold circle.

plexity of the solutions. We analysed the first individual to appear in each evolutionary run with a fitness score above some threshold. The results are show in Table 3.2.

We can see that on average, novelty search finds good individuals with less complex neural networks. For example, novelty search finds the first good individual (with fitness value over 0.8) on average with 22.5 neurons and 38.4 links, while the fitness-based evolution finds the first good individual on average with 26.6 neurons and 44 links. Taking in consideration that the initial networks (without hidden

Table 3.2: Network complexity of the first individual evolved with a fitness score above the threshold, in each evolutionary method (Fit – Fitness-based evolution; NS – Novelty Search). The values are averaged over 10 evolutionary runs for each method.

Threshold	Generation		Neurons		Links	
	Fit	NS	Fit	NS	Fit	NS
0,60	4,38	3,13	19,63	19,13	34,63	34,13
0,65	19,63	9,13	21,38	20,00	36,63	35,13
0,70	41,00	15,63	23,14	21,13	38,71	36,25
0,75	56,86	35,00	24,71	21,13	40,86	36,75
0,80	94,43	71,88	26,57	22,50	44,00	38,38
0,825	131,43	111,00	28,00	24,20	46,86	41,8

neurons) have 19 neurons and 34 links, these differences can be considered quite pronounced.

These differences in network complexity can offer an explanation to the steeper fitness trajectory of novelty search. Due to the incremental nature of NEAT, more complex networks take more generations to evolve. If fitness-based evolution starts to converge to more complex structures, it takes more time to evolve effective controllers. As novelty search does not converge, and explores the simple solutions before moving on to more complex ones, it is more capable of finding solutions with less network complexity. Looking at the early solutions found by novelty search, we discovered that in some cases they are the ones that the fitness-based evolution could not evolve at all (behaviours that used the wall). In other cases, they were apparently the same solutions that the fitness-based evolution would find in later generations with more complex networks.

3.2 Energy Management Experiments

In this section, we study the application of novelty search to a different swarm robotics task, in which multiple robots must share a single battery charging station in order to survive. The charging station only has room for one robot and the robots must therefore evolve effective coordination strategies.

3.2.1 Energy Management Related Work

The problem of autonomous robot charging and resource conflict management is widely studied in the literature. In (Cao et al., 1997), resource conflict is identified as one of the fundamental challenges in the design of cooperative behaviours in multi-robot systems. Resource conflict arises when a single indivisible resource (in our experiments, the charging station) is requested by multiple robots at the same

time.

The problem of sharing a charging station in particular is studied in (Muñoz Meléndez et al., 2002). In this work, a few strategies are proposed for a group of three robots to share a charging station using simple mechanisms, without communication. These strategies share a basic approach: each robot seeks the charging station when its energy level is below some threshold and leaves the charging station when the energy level reaches some other pre-defined threshold. While this approach works if the energy levels of the robots are not synchronised, it may fail if the robots start with the same energy levels. The paper also proposes some strategies to create this alternation. In (Michaud and Robichaud, 2002), the problem of sharing a charging station is also addressed. In this work, more complex strategies are proposed, with the robots being able to reason and predict about their energetic capabilities.

To the best of our knowledge, there is only one previous work that uses artificial evolution to approach this task. In (Bastos, 2011), genetic algorithms are used to evolve neural controllers for robots of a swarm that should perform an energy management task. Multiple charging stations are present in the environment, each one with capacity for two robots. The swarm is composed by 5 robots, each one equipped with sensors with a maximum range bigger than the size of arena. The maximum autonomy time of the robots is also just slightly inferior to the simulation time. This set of factors oversimplifies the problem, and the maximum fitness score can be achieved with only 10 generations of the evolutionary algorithm. Despite addressing a similar task, our experimental setup is much more demanding, and as such we can not establish comparisons with Bastos' work.

3.2.2 Experimental Setup

The experiments used a resource sharing task, where a swarm of 5 homogeneous robots must coordinate in order to allow each member periodical access to a single battery charging station. The charging station only has room for one robot. To survive, each robot will have to possess several competencies: navigate and avoid walls, find and position itself on the charging station to recharge, and effectively share the common resource with the other robots.

The environment is a 4 m by 4 m square arena bounded by walls. The charging station is placed in the centre of the arena. The robots are based on the physical characteristics of the e-puck educational robot (Mondada et al., 2009), but do not strictly follow its specification. Each simulated robot has 8 IR sensors evenly distributed around its chassis for the detection of obstacles (walls or other robots) up to a range of 10 cm, and 8 sensors dedicated to the detection of other robots up to a 25 cm range.

Each robot starts with full energy (1500 units) and lose energy over time. In

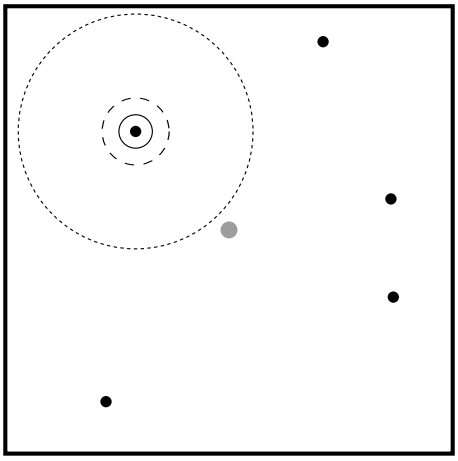

Figure 3.10: The energy management experimental setup. The grey circle in the middle is the charging station. The black filled circles are the robots (starting positions vary in each simulation). The solid circle around the top left robot represents the range of the obstacles sensor, the dashed circle represents the range of the robots sensor and the dotted circle represents the range of the charging station sensor.

order to charge, the robots must remain still (maintain the same position) inside the charging station, which has the same diameter as a robot. Each robot is additionally equipped with (1) a ring of 8 sensors for the detection of the charging station up to a range of 1 m; (2) a boolean sensor that indicates whether the robot is inside the charging station or not; (3) an internal sensor that reads the current energy level of the robot. If a robot runs out of energy, it stops working, and remains immobile until the end of the simulation. The experimental setup is depicted in Figure 3.10.

We test each controller 10 times in varying initial conditions. In each simulation, the initial position and orientation of each robot was randomised. The set of possible initial positions only includes those from where a robot cannot sense the charging station. Each simulation lasts for 400 s of simulated time at 10 updates per second, resulting in 4000 simulation steps.

We used two slightly different setups in our experiments. In setup A, the robots lose a fixed 10 units of energy per second. In setup B, the robots lose energy proportionally to the power used by their motors, at a rate between 5 and 10 units of energy per second. In both setups, the charging station charges a robot at a rate of 100 units of energy per second.

3.2.3 Configuration of the Evolutionary Algorithms

The evolutionary parameters (of NEAT and novelty search) are the same as the aggregation experiments, described in Section 3.1.3.

The fitness function F used to evaluate the controllers is a linear combination of

the number of robots alive at the end of the simulation and the average energy of the robots throughout the entire simulation:

$$F = 0.9 \cdot \frac{|a_T|}{N} + 0.1 \cdot \sum_{t=1}^{T} \sum_{i=1}^{N} \frac{e_{i_t}}{TNe_{max}} \quad , \tag{3.6}$$

where $|a_T|$ is the number of robots alive in the end of the simulation, T is the length of the simulation, N is the number of robots in the swarm, e_{i_t} is the energy of the robot i at instant t, and e_{max} is the maximum energy of a robot.

The average energy component of the fitness function, despite not being directly associated with the objective of survival, had to be included in order to bootstrap the evolution. Due to the difficulty of the setup, it is unlikely that the initial population contains solutions where at least one robot survives until the end. Analysing the initial populations, we calculated that the probability of randomly generating a solution where at least one robot survives until the end (in any of the trials) is 0.58% in setup A, and 0.89% in setup B. As such, it was necessary to add another component (with low weight) that could distinguish the equally poor solutions.

The behaviour characterisations, used to compute the behavioural difference in NS, followed a different principle from the aggregation experiment. While in the previous task there was a clear objective that the robots should achieve at the end (aggregate), in this task we want the robots to evolve a stable behaviour that endures through time, without a defined end. In the previous experiments, we measured behaviour features sampled through time, to describe the progression of the swarm towards the final aggregation state. In this experiment, we will only use single-value measures that describe the stable behaviour of the swarm.

The behaviour characterisation is closely related to the fitness function. It is composed by just two measures, normalised between 0 and 1: (1) the number of robots alive at the end of the simulation; and (2) the average energy of the alive robots throughout the simulation. It is defined by:

$$\mathbf{b_{simple}} = \left(\frac{|a_T|}{N}, \sum_{t=1}^{A} \sum_{i \in a_t} \frac{e_{i_t}}{A \cdot |a_t| \cdot e_{max}} \right) \quad , \tag{3.7}$$

where A is the number of time steps in which there was at least one robot alive and a_t is the set of alive robots at instant t.

3.2.4 Overcoming Deception with Novelty Search

To study how novelty search and fitness-based evolution are influenced by the deceptiveness of the problem, we evaluated and compared their performance in two different setups (described in Section 3.2.2). These setups, despite being intuitively similar, originate quite different fitness landscapes. The results can be seen in Figure 3.11.

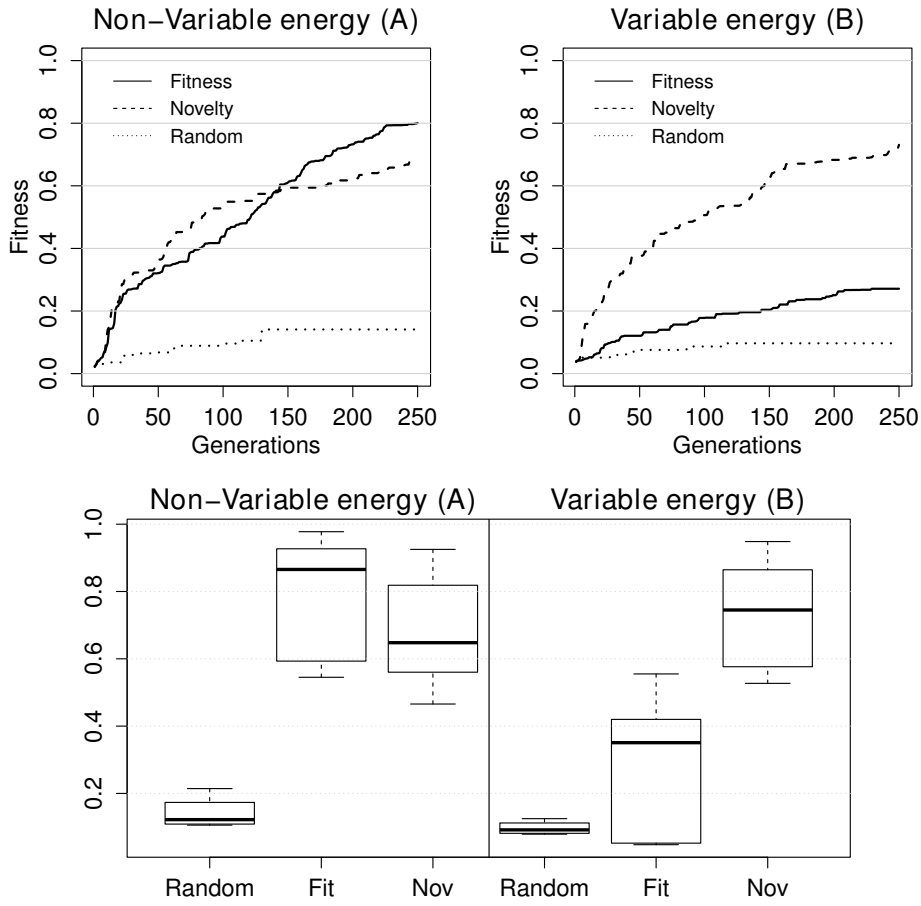

Figure 3.11: Top: Fitness score of the best individual found so far until each generation, with each evolutionary method. Values are averaged over 10 evolutionary runs with each method. Bottom: Boxplots with the best fitness score found in each evolutionary run.

Non-variable Energy Spending

In the setup A, fitness-based evolution can achieve on average higher fitness scores than novelty search. Novelty search, on the other hand, can bootstrap faster, achieving higher fitness scores until generation 100. The statistical significance of these differences could only be verified with p-value < 0.1 (Student's t-test). Both methods are significantly better than evolution with random selection (p-value < 0.01). The extremely poor performance of random evolution highlights the difficulty of the task. To acquire the ability of recharging, the robots must know how to seek the charging station (which requires the ability of avoiding the walls), and once they are close to it, they must position themselves inside the station and remain still. In order for more than one robot to survive, the robots should be capable of sharing the charging station, which includes determining when to go charge and when to leave the station.

The maximum fitness score in practice is about 0.97, which corresponds to all

robots surviving until the end of the experiment, while maintaining high levels of energy. The best solutions evolved by both evolutionary methods were on average far from this maximum score. This means that the best solutions were mediocre, with only about 3 robots surviving until the end of the experiment. The solutions evolved by novelty-search and fitness-based evolution are similar. It is often hard to make sense of the group dynamics in the evolved solutions, because sometimes robots take unexpected and unintelligible actions. However, it is possible to identify some patterns of behaviour (Figure 3.12):

1. The robots start navigating in circles until they sense the charging station, and when they do, they head towards it and charge. If another robot is already in the charging station, the first leaves and the new one takes its place. The robot that leaves moves around in the arena and then returns to the station.

2. The robots search for the charging station in straight trajectories and head towards it when they find it. If another robot is already charging, the first leaves and starts surrounding the charging station closely. It tries to charge again when its energy level is below some threshold.

3. The robots start navigating in circles to find the charging station. Once a robot is in the charging station, it remains there until its energy is above some threshold and then leaves. When a robot leaves, it moves away just a little from the station and remains keeps its position (with small movements) until its energy is almost depleted.

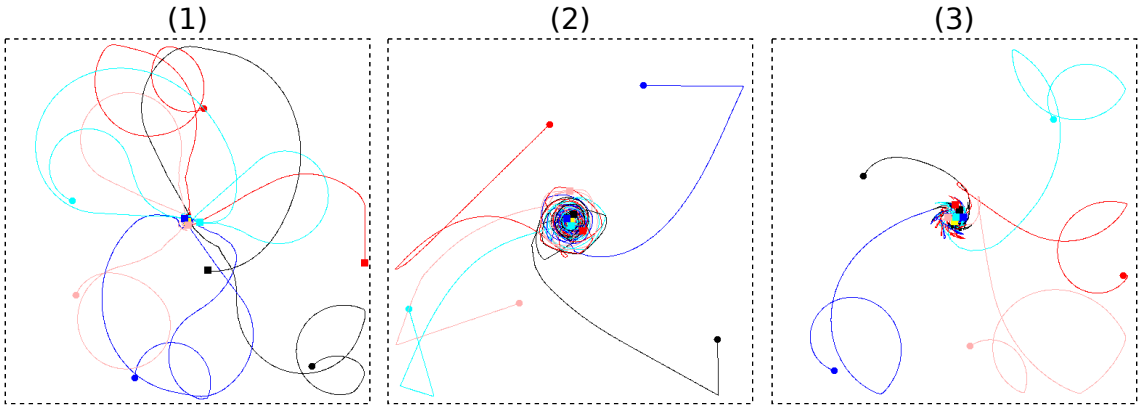

Figure 3.12: The patterns of behaviour corresponding to the best solutions found by novelty search and fitness-based evolution. Each line represents the trajectory of the robot throughout the simulation. The circle depicts the initial position of the robot and the square its final position.

These distinct behaviours all have the same shortcoming: sometimes the robots cannot find the charging station initially and die without ever recharging; when

leaving the charging station sometimes they get lost and cannot return to it; in some cases one robot dies too close to the charging station, preventing the others from charging, and the whole swarm dies. Both novelty search and fitness-based evolution could find solutions with the above described behaviour patterns. However, the trajectories in finding the charging station and leaving it may differ, as well as the energy thresholds for trying to charge and for leaving the station. Observing the solutions in action, there was not a perceivable difference in the quality of the solutions found by fitness-based evolution and novelty search, respectively.

Variable Energy Spending

In the setup with variable energy (setup B), the task is actually easier, since the robots can spend less energy than in setup A. Their energy consumption is a function of the wheels speed, with the maximum rate of consumption being equal to that in setup A. However, fitness-based evolution performs much worst in setup B. This happens because new local maxima are created in the fitness landscape, and the evolution gets stuck in them. Figure 3.13 depicts the individual fitness trajectories obtained with fitness-based evolution. We can see that half of the evolutionary runs fail to achieve fitness scores higher than the best fitness in the initial population, failing to bootstrap the evolution.

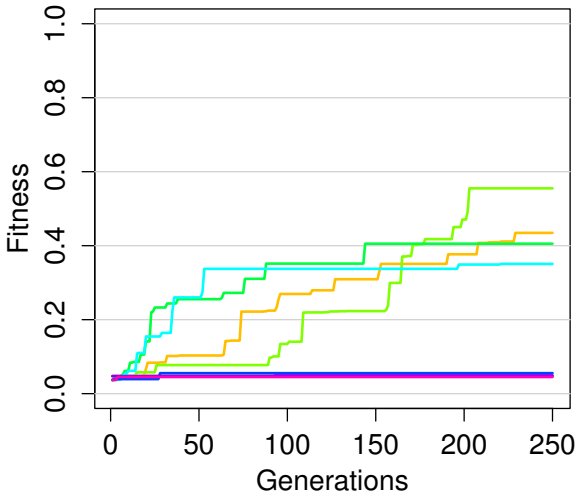

Figure 3.13: The fitness trajectories (best fitness found so far in each generation) of each evolutionary run of fitness-based evolution, using the setup B. Four out of eight evolutionary runs could not bootstrap the evolution (some lines overlap).

Observing in action the best controllers evolved by the evolutionary runs that prematurely converged, we can identify the following behaviours: (1) The robots move slowly in small circles, and if they eventually get close to the charging station, they go towards it and stay there until the end, causing the other robots do die; (2) the robots do not move at all. The dead end in the fitness landscape can be

easily identified: the evolution starts to converge to controllers that reduce the wheel speed in order to save energy. Saving energy will cause the robots to remain alive longer, thus increasing the fitness score of the solution just slightly. But reducing the wheel speed too much prevents the robots of charging, and results in very low fitness scores. Once the population starts converging to the local maxima of saving energy, it can hardly get away from it, since reducing the movement speed is adverse to finding the charging station.

With the setup A (non-variable energy), the average energy component of the fitness-function helped in bootstrapping the evolution, as the only possible way of increasing the average energy was for the robots to use the charging station. The population evolved in the direction of learning how to charge efficiently, which is clearly a good step towards the solution of the problem. With the setup B (variable energy), there was another (easier) way of increasing the average energy: the robots can reduce their movement speed. The bootstrapping component of the fitness function actually had an adverse effect, and lead the population to a very premature local maxima.

The performance of novelty search, on the other hand, was not affected by the variable energy, since the evolution is divergent and does not get stuck in local maxima. It confirms that novelty search can successfully overcome deception and bootstrap the evolution in situations where fitness-based evolution can not. However, analysing the best evolved solutions in action reveals that they are very similar to the ones previously presented in Figure 3.12. The solutions with higher fitness scores did not seem to take advantage of the variable energy. Intuitively, one of the best solutions for this problem would be for the robots to remain still after leaving the charging station, in order to conserve energy, and only move again when their energy levels reach a low level. Nevertheless, this pattern of behaviour was poorly explored by evolution. The explanation for this is conflation – as movement speed is not directly contemplated in the novelty search distance metric, there is not a direct pressure to explore that possibility.

To study the explored behaviour space, the individuals evolved in fitness-based evolution were also evaluated with the behaviour measure used in novelty search. We randomly sampled 100.000 individuals evolved by each evolutionary method, and then mapped them in the corresponding plot, according to the two components of the behaviour measure. The resulting plots can be seen in Figure 3.14.

The analysis of the explored behaviour space confirms that novelty search was not significantly affected by the inclusion of variable energy, as the space exploration is very similar in both setups. When compared to fitness-based evolution in the non-deceptive setup (A), novelty search explores a greater diversity of solutions where only one robot survives. However, since novelty search spends much time

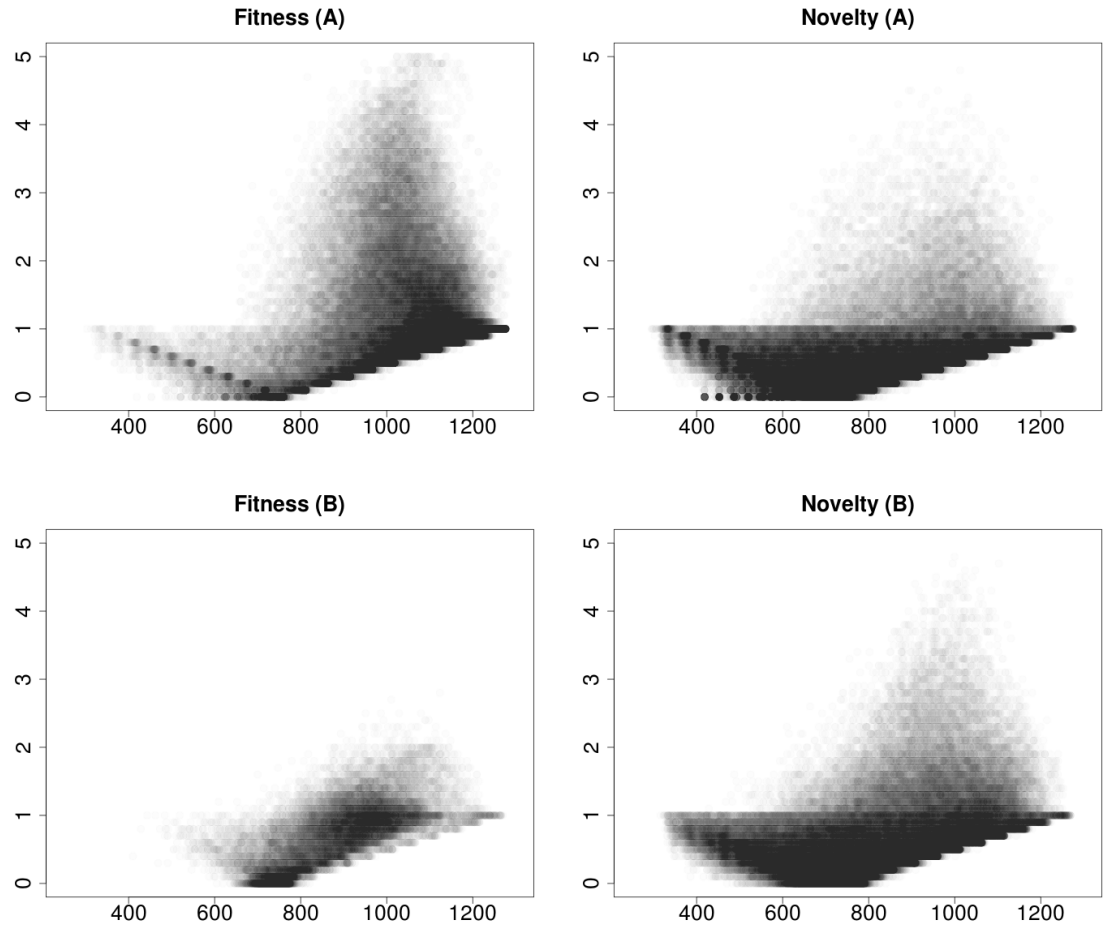

Figure 3.14: Behaviour space exploration with both setups and both evolutionary methods. The x-axis is the average energy level of the robots still alive, the y-axis is the number of robots alive at the end of the simulation. Each individual is mapped according to its behaviour. Darker zones mean that there were more individuals evolved with the behaviour of that zone.

exploring this mediocre behaviour zone, it fails to explore adequately the behaviour zones where more robots survive (associated with higher fitness scores). This can explain why novelty search has a faster bootstrap than fitness-based evolution, but fails to achieve higher fitness scores in end. In Chapter 4, we will address this problem in greater detail.

The plot of fitness-based evolution in setup B also confirms the prominent local maxima identified before: the robots do not move at all (alive = 0 and average energy $\approx$ 700)); and only one robot survives (alive = 1 and average energy $\approx$ 1000).

3.2.5 A More Rich Behaviour Characterisation

In the previous section, we saw that novelty search fails to explore some interesting behaviour zones, such as solutions that exploit the movement speed of the robots in order to save energy. To try to overcome this limitation, we devised a new behaviour measure that describes with greater detail the behaviour of the swarm. The new behaviour characterisation is an extension of the Equation 3.7, composed with two more measures, completely independent from the fitness function: (1) the average movement of the alive robots throughout the simulation; and (2) the average distance of the alive robots to the charging station. The movement of a robot in a given instant is determined by the average wheel speed at that instant. These two measures are also normalised between 0 and 1.

This new measure was experimented only with the setup B (variable energy spending). The comparison with the previous measure and fitness-based evolution is depicted in Figure 3.15.

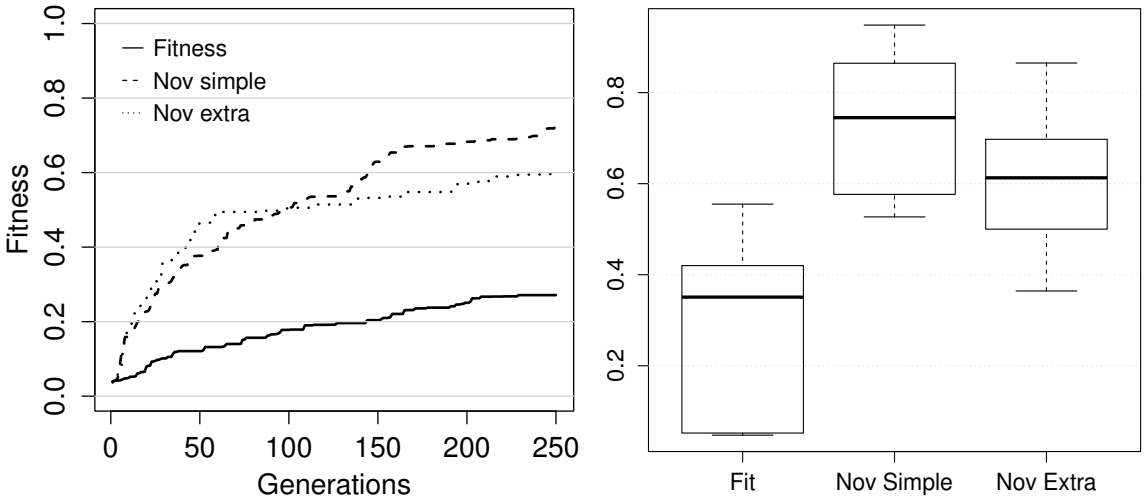

Figure 3.15: Left: Fitness score of the best individual found so far until each generation, using the setup B, and different novelty measures. Values are averaged over 10 evolutionary runs. Right: Boxplots with the best fitness score found in each evolutionary run.

The fitness performance of novelty search with the new behaviour measure was worse than novelty with the more simple measure. The more complex measure can actually be more effective in the early stages of the evolution, but then the evolution seems to get lost in the behaviour space and fails to find solutions with higher fitness scores. To confirm this hypothesis, we analysed the explored behaviour space. Since each behaviour is now 4-dimensional, a dimensionality reduction method had to be adopted for visualisation. The visualisation methodology was similar to the one described in Section 3.1.4. A Kohonen map was built with all the behaviours evolved

by novelty search, and then it was counted how many behaviours were mapped to each unit of the map. The resulting plot is in Figure 3.16.

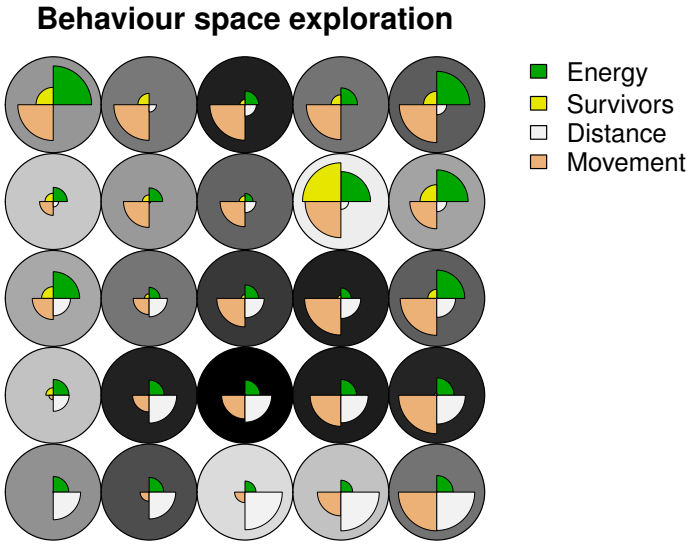

Figure 3.16: Kohonen map representing the explored behaviour space with novelty search with the expanded behaviour measure. Each circle represents a behaviour pattern, depicted by the 4 slices of different colour. Each slice represents one component of the behaviour measure – the bigger the slice, the bigger the value of that component. The darker the background of a circle is, the more individuals were evolved with the corresponding behaviour.

The analysis of the behaviour space reveals that novelty search successfully explored many different behaviours – a notable variety of combinations of average energy, movement and distance to the charging station. However, the number of surviving robots at the end was the least explored component of the behaviour measure. In almost all the behaviour patterns, the number of surviving robots stayed between 0 and 1. Only one pattern has about 3 surviving robots (the circle with the largest yellow slice), and this pattern was the least common in the evolution. This analysis confirms that novelty search did get *lost* exploring behavioural possibilities associated with low fitness scores.

The explanation for this phenomena is that the two new components that were added to the behaviour characterisation are mostly orthogonal to the component more related with the performance of the solutions (the number of surviving robots). As such, novelty search can find novelty by exploiting the three other components, without having to explore the component of the number of surviving robots. Furthermore, the three other components are easier to explore than the number of surviving robots, since they require less coordination among the robots. Consequently, novelty search ends up exploiting the three components less correlated with the performance of the solution, and does not exert sufficient selective pressure to adequately explore

the single component that is directly related with the performance of the solutions. In this case, reducing conflation was actually not beneficial for the performance of novelty search, as it significantly decreased the attention from exploring the most important dimensions.

With the previous, more simple, behaviour characterisation, the possibilities were more restricted. To find high novelty scores, novelty search had to focus more on exploring solutions with an increased number of surviving robots, as the reduced number of components resulted in less possibilities of obtaining high novelty scores. As a consequence, novelty search with the simple measure had significantly more success in finding solutions with high fitness scores, when compared to novelty search with this more complex measure. This issue is studied in the literature and was presented in Section 2.7. In the next chapter, we will study and experiment methods that try to overcome this limitation, by using the fitness function to provide selective pressure towards high fitness behaviour zones.

3.3 Discussion

The experiments described in this chapter used two different tasks for studying the use of novelty search in the evolution of controllers for a swarm of robots. Overall, novelty search was successfully used in this domain, offering a number of significant advantages, but also revealing a number of issues and challenges. Most of our results regarding novelty search are in accordance with the previous works in the literature. In this section, we will discuss the successes and limitations of novelty search in a broader perspective.

Effectiveness of Novelty Search

Our experiments reveal that novelty search can be as good as fitness-based evolution in non-deceptive setups, regarding the fitness score of the evolved solutions. Other works have shown that novelty search can perform better than fitness-based evolution in deceptive tasks, but fails to match the performance of fitness-based evolution as the task gets less deceptive (Lehman and Stanley, 2011a; Mouret, 2011). Our results show that the aggregation task and the energy task with non-variable energy spending are not notably deceptive, as fitness-based evolution can always find high-scoring solutions. Still, novelty search managed to perform almost as well as the fitness-based evolution.

The energy task with variable energy spending is highly deceptive, as fitness-based evolution frequently gets stuck in poor local maxima. In this setup, novelty search largely outperformed fitness-based evolution. It could effectively overcome deception, and did not get stuck in any local maxima. This result confirms that

novelty search is an effective strategy for overcoming deception in evolutionary swarm robotics. Overcoming deception allows for the evolution of solutions for more demanding tasks, pushing evolutionary swarm robotics towards real-world, complex problems.

In both tasks, and with all behaviour measures, novelty search exhibited a common pattern regarding the fitness score of the solutions that are discovered along the evolution. Novelty search can bootstrap the evolution notably well, better than fitness-based evolution, finding good solutions early in the evolution, with less complex neural networks. However, after this initial bootstrap, novelty search struggles to find solutions with progressively better fitness scores. Our results suggest that this happens because novelty search starts to diverge too much, and does not focus the search in behaviour zones associated with high fitness scores. Even towards the end of the evolution, novelty search continues to actively explore behaviours with very low fitness-scores, wasting time in behaviours that at that stage of evolution are mostly useless.

This pattern becomes even more clear in the energy management task, with the expanded behaviour measure. With this measure, the performance of the search was significantly worse, when compared to novelty search with the more simple measure. The more complex measure achieved a faster bootstrap, but then the search starts exploring dimensions of the behaviour space that are mostly irrelevant for the performance of the solutions, and the fitness scores started to improve very slowly. This suggests that the pressure exerted by the fitness function should not be completely ignored, as we did with novelty search in this chapter. At least in the later stages of evolution, the fitness pressure seems to be necessary in order to focus evolution in behaviour zones of high fitness scores. In the next chapter, we will address this issue, and study how the pressure from the fitness function can actually help novelty search in the exploration of the behaviour space.

Design of the Behaviour Characterisations

The biggest challenge in using novelty search in the domain of swarm robotics was the definition of the novelty measure. Our experiments suggest that conflation can be a serious issue when evolving collective behaviours with novelty search. While in single robot systems, conflation can be mitigated by describing the full behaviour of the robot, for example its position in space over time (Lehman and Stanley, 2011a), in swarm robotics that is not possible. Describing the behaviour of all the robots individually would open the search space too much. It would also introduce scalability issues, for example if the number of robots varies or if the swarm is very large. It is necessary to devise measures that evaluate the swarm as whole. Conflation is essential to cope with the greater diversity of collective behaviours,

but caution must be displayed in order not to conflate aspects of the swarm that are relevant to the solution.

The experiments with the aggregation task showed that by combining different novelty measures, we can reduce conflation and improve the performance of novelty search. This combination can simply be the concatenation of the behaviour vectors associated with each measure, which was effective in this task. This technique was also used in the energy management experiments, however, it had an adverse effect in the effectiveness of novelty search. By enriching the behaviour characterisation, conflation was reduced, but evolution lost the pressure to explore solutions with high performance, exploring instead other behavioural possibilities that were irrelevant for the achievement of the objective. The difference was that in the aggregation experiments the behaviour characterisation was enriched with another component that was also relevant to the aggregation objective, while in the energy experiments the components that were added were orthogonal to the objective.

This result suggests that it must be displayed caution when combining novelty measures. First, it is not advisable to blindly put everything together in a single behaviour measure, as the evolution might exploit behaviour zones that are not relevant for solving the task. The components of the behaviour characterisation should be somewhat related with the objective that one is trying to achieve, which happened in the aggregation experiments, but did not happen in the energy management experiments with the expanded behaviour characterisation. Second, if some components of the behaviour characterisation are known to be more relevant than others, this relevance should be expressed in the characterisation, for example by scaling some of the components. This allows for a greater exploration around the components that are more relevant. Our approach for combining behaviour measures was effective in one experiment, but overall it revealed to be naive, and more sophisticated ways of combining behaviour components should be devised.

In the experiments with the aggregation task, the combined behaviour measure (sampling of centre of mass and number of clusters) was a vector of length 100. In the energy management task, with the expanded behaviour measure, it was a vector of length 4. However, novelty search had more difficulty in exploring the behaviour space in the energy management task, than in the aggregation task. This result is a reminder that long behaviour characterisations do not necessarily translate in a behaviour space more difficult to explore, and do not degrade the performance of the evolution. In the aggregation task, the values of the behaviour characterisation vector are highly correlated between them, and correlated with the quality of the solutions. As such, the effectiveness of novelty search is not affected by the long behaviour characterisation. On the other hand, in the energy management task, the values of the behaviour characterisation are mostly independent from one another,

and some are independent from the quality of the solutions. Therefore, despite being just 4 values, there is a greater risk of novelty search start exploring zones of the behaviour space that are not relevant to the objective.

Behavioural Diversity

Regarding the solutions evolved in the aggregation task, the best solutions found by both novelty search and fitness-based evolution were satisfactory, but frequently displayed major flaws. It suggests that there is still margin for improvement. Although it is hard to compare our solutions with the ones in the related work, as we use quite different robot sensors in our setup, some similarities can be identified. In (Trianni et al., 2003) it is described an aggregation behaviour (*static clustering*) where the robots explore the arena in circles until they find one another, and when they do, they form a static cluster and wait for other robots to join the cluster. This behaviours is very similar to the best solution evolved by fitness-based evolution in our experiments, and to many behaviours evolved by novelty search.

In the energy management task, the best solutions were just mediocre, both in novelty search and fitness-based evolution, managing to keep alive on average only 3 of the 5 robots. The task seems quite demanding, and there is still room for improvement. We can not establish a comparison with related work, since to the best of our knowledge there are no works describing the evolution of solutions for an energy management task. However, there are works in swarm robots energy management from a non-evolutionary perspective. Interestingly, a number of solutions found by both fitness-based evolution and novelty search are remarkably similar to the basic strategy described in (Muñoz Meléndez et al., 2002) and presented in Section 3.2.1.

We showed that the diversity found by novelty search can produce many different solutions to the same task. This is most evident in the aggregation task, where there is one class of solutions explored by novelty search, that was not evolved at all by fitness-based evolution (the use of the walls to achieve aggregation). This is valuable because it can be used to provide a range of different solutions to the experimenter that is using the evolutionary process. It is especially relevant in the swarm robotics domain, because there are many behaviour possibilities and non-obvious relations between the agents that might be revealed. Interestingly, the related work does not describe aggregation behaviours that use the walls of the arena to form the aggregates, a behaviour that can be found in the biological world. This indicates that novelty search can indeed be used to unveil a diversity of forms of self-organisation.

The Kohonen maps proved to be useful in the visualisation of the behaviour search space. In situations where the behaviour space has more than 2 dimensions, a dimensionality reduction method is required in order to visualise the search space. Kohonen maps are adequate because they discretise the behaviour space around

"behaviour patterns", maintaining the topological relations between these patterns. This allows the understanding of the behaviour zones that were explored in the evolution, and the zones where the fitness-based evolution gets stuck. We verified that controllers mapped to different neurons typically have different observable behaviours. This suggests that analysing the differences in the behaviour vectors might be a way of automatically identifying distinct solutions.

Another advantage of novelty search verified in our experiments was that it could found solutions with simpler neural networks than the ones found by fitness-based evolution, for similar fitness scores. This is coherent with the results reported in (Lehman and Stanley, 2010a). Our results suggest that while fitness-based evolution tends to converge towards more complex networks, novelty search performs a more ordered search, exploring the simple solutions before moving on to more complex ones.

Chapter 4

Combining Novelty and Fitness

In the previous chapter, we saw that in some cases novelty search struggles to guide the evolutionary process towards behaviour zones associated with high fitness scores. This problem has also been encountered in other studies, and the common approach to overcome this issue is to combine novelty search with fitness-based evolution (see Section 2.7). Combining these two evolutionary techniques is promising because while novelty search is an exploration technique, fitness-based evolution can be viewed as an exploitation technique. The objective is to simultaneously take advantage of novelty search's capability of exploring the space of possible solutions, and the capability of fitness-based evolution of progressively improving solutions.

In this chapter, we propose *Progressive Minimal Criteria Novelty Search* (PMCNS), a new method for bringing together novelty search and fitness-based evolution. We compare it with another successful method for combining fitness and novelty, henceforth referred to as *linear blend* (Cuccu and Gomez, 2011), where each individual is assigned an evaluation score that is a linear combination of its novelty score and its fitness score (see Section 2.7.2). Both the energy management task and the aggregation task are used in the experiments described in this chapter. We compare the results described in the previous chapter with the results obtained with linear blend and PMCNS, by using the same experimental setups.

4.1 Progressive Minimal Criteria Novelty Search

One method of combining novelty and fitness is Minimal Criteria Novelty Search (MCNS) (Lehman and Stanley, 2010a). With MCNS, the behaviour space is restricted by defining domain dependent minimal criteria that the evolved individuals must meet. However, some limitations were identified, such as the necessity of fine tuning the minimal criteria, and the difficulty of bootstrapping the evolution if none of the individuals in the initial population meet the minimal criteria (see Section 2.7.1). To overcome these limitations, we propose an extension of MCNS,

named Progressive Minimal Criteria Novelty Search (PMCNS).

The objective of PMCNS is to take advantage of the behaviour space restriction provided by MCNS, without having to pre-define domain dependent minimal criteria, thus avoiding the need to seed an initial population that satisfy the criteria. In our algorithm, the minimal criterion is a dynamic fitness threshold – individuals with a fitness score greater than the threshold meet the criterion. Although in novelty search the fitness score does not influence the evolution, typically a fitness function must be provided anyway, in order to be able to identify the best controllers found by novelty search. In this way, our algorithm does not require the definition of task-specific minimal criteria or any other additional measures. It is important to note that this fitness function is used only to evaluate the quality of the solutions in respect to the objective, not to guide the evolution. As such, it does not need to be shaped to overcome local maxima or bootstrap the evolution, artifices that are frequently needed in fitness-based evolution.

As pre-defining a fixed fitness threshold would raise the same issues as in MCNS (choosing the adequate threshold and bootstrapping the search), we progressively increase the minimal criterion (fitness threshold) during the evolutionary process. The idea behind the increasing fitness criterion is to progressively restrict the search space, to avoid spending too much effort on search through regions of the behaviour space with the least fit behaviours.

The minimal criterion starts at the theoretical minimum of the fitness score (typically zero), so all controllers initially meet the criterion. In each generation, the new criterion is found by determining the value of the P-th percentile of the fitness scores in the current population, i.e., the fitness score below which P percent of the individuals fall. The P-th percentile ($0 \leq P < 100$) of N ordered values is obtained by first calculating the ordinal rank n:

$$n = \frac{P}{100} \times N + \frac{1}{2} \ , \tag{4.1}$$

rounding the result to the nearest integer, and then taking the value v_n that corresponds to the rank n. Only increases in the minimal criterion are allowed, and in order to smoothen the increase, the minimal criterion from the previous generation is used to compute the criterion for the current generation:

$$mc_g = mc_{g-1} + \max(0, (v_n - mc_{g-1}) \cdot S) \ , \tag{4.2}$$

where mc is the minimal criterion, and S is the smoothing parameter. The score of each individual in the population is then calculated according to:

$$\text{score}(i) = \begin{cases} nov_i & \text{if } fit_i \geq mc_g \\ 0 & \text{otherwise} \end{cases} \ , \tag{4.3}$$

where nov_i and fit_i is the novelty score and the fitness score of the individual i, respectively. The parameter P controls the exigency of the minimal criterion (0 – all individuals meet the criterion, 1 – only the individual with the highest fitness meets the criterion). The smoothening parameter S controls the speed of the adaptation of the minimal criterion (0 – no changes at all, 1 – the value from the previous generation is not considered).

The operation of the novelty archive is not modified, and works as in the original novelty search algorithm. Even individuals that do not meet the minimal criterion are still added to the repository if their behaviour is sufficiently novel.

4.2 Energy Management Experiments

In this section, we revisit the energy management task, by using evolutionary methods that combine novelty search with the fitness function. We experiment with the linear blend method described in the literature, and the PMCNS method proposed in this chapter. We compare these methods with the previous results obtained with novelty search and fitness-based evolution, and study how PMCNS compares to linear blend, and how it can overcome the limitations of MCNS.

The experimental setup was exactly the same as the one used in the previous energy management experiments, described in Section 3.2.2. The fitness function and novelty distance metrics are also the same of the previous experiments, described in Section 3.2.3.

4.2.1 Impact of Behaviour Space Dimensionality

To test the capability of fitness-novelty linear blend and PMCNS in dealing with behaviour spaces of different dimensionality, we experimented these methods with the two behaviour measures presented in Section 3.2.2:

- Simple measure: Average energy of the alive robots and the number of robots alive at the end of the simulation
- Expanded measure: Simple measure plus the average movement of each robot and the average distance to the charging station.

The setup B (variable energy spending) was used in these experiments. The novelty-fitness linear blend was run with $\rho = 0.75$ (75% novelty score and 25% fitness score). PMCNS was run with $P = 0.5$ (percentile parameter) and $S = 0.25$ (smoothening parameter). These parameters will be discussed below in Section 4.2.3. The results can be seen in Figure 4.1.

The results show that both linear blend and PMCNS are more effective than pure novelty search in both behaviour spaces. The performance differences are

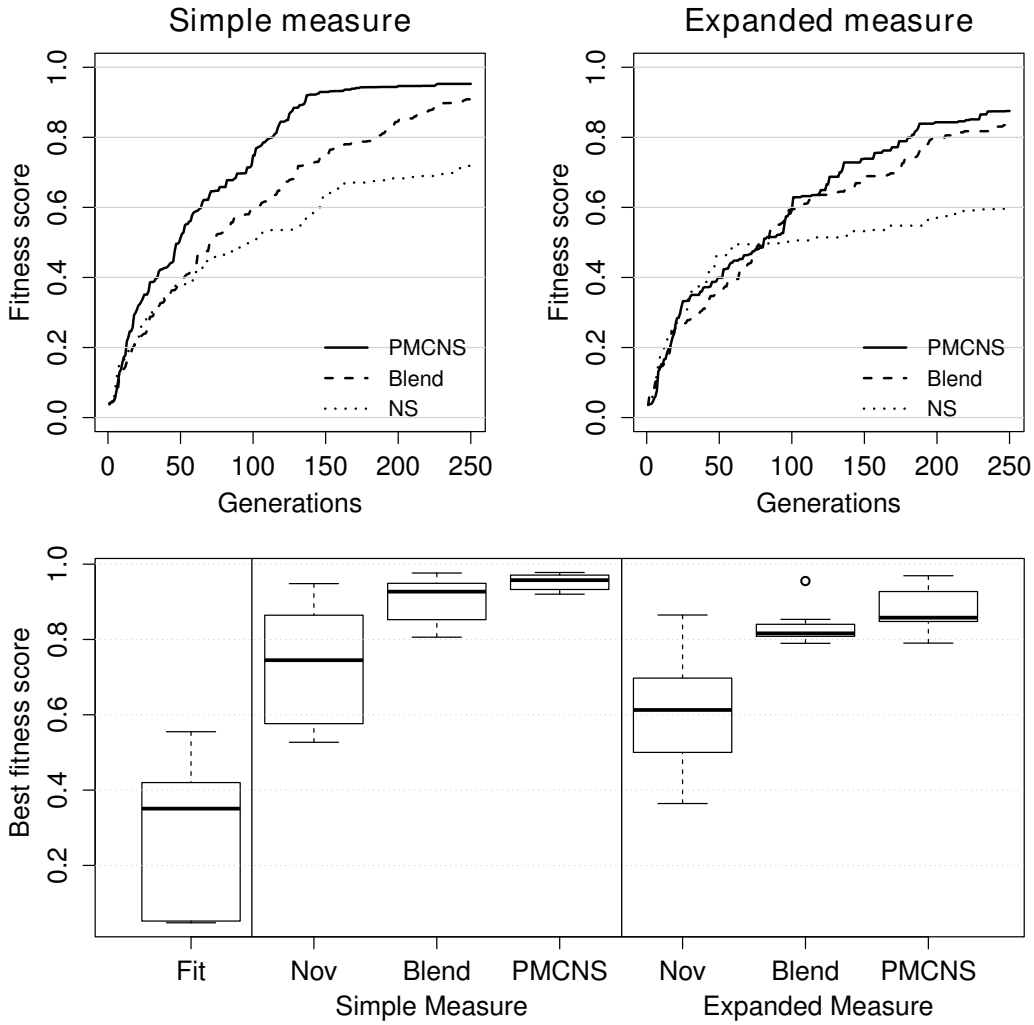

Figure 4.1: Top: Fitness trajectory obtained with PMCNS novelty search, linear blend, and pure novelty search, with each novelty distance measure. Bottom: Boxplots with the best fitness score found in each evolutionary run. The Simple measure is composed of the average energy level and number of surviving robots. The expanded measure is the simple measure plus the average distance to the charging station and average movement of the robots.

more significant with the expanded measure, that creates behaviour space with more dimensions. With this measure, novelty search is significantly affected. It can successfully bootstrap the evolution but fails to reach high fitness scores in the end, as the evolution starts focusing the exploration on behaviour dimensions that are not relevant for the performance of the solution. On the other hand, the results suggest that Linear blend and PMCNS can overcome this issue, and guide the evolution towards progressively better solutions. With the simple measure, PMCNS shows a significantly better performance (Student's t-test, with p-value < 0.05) than linear blend. With the expanded measure, there are no significant differences between the performance of both methods.

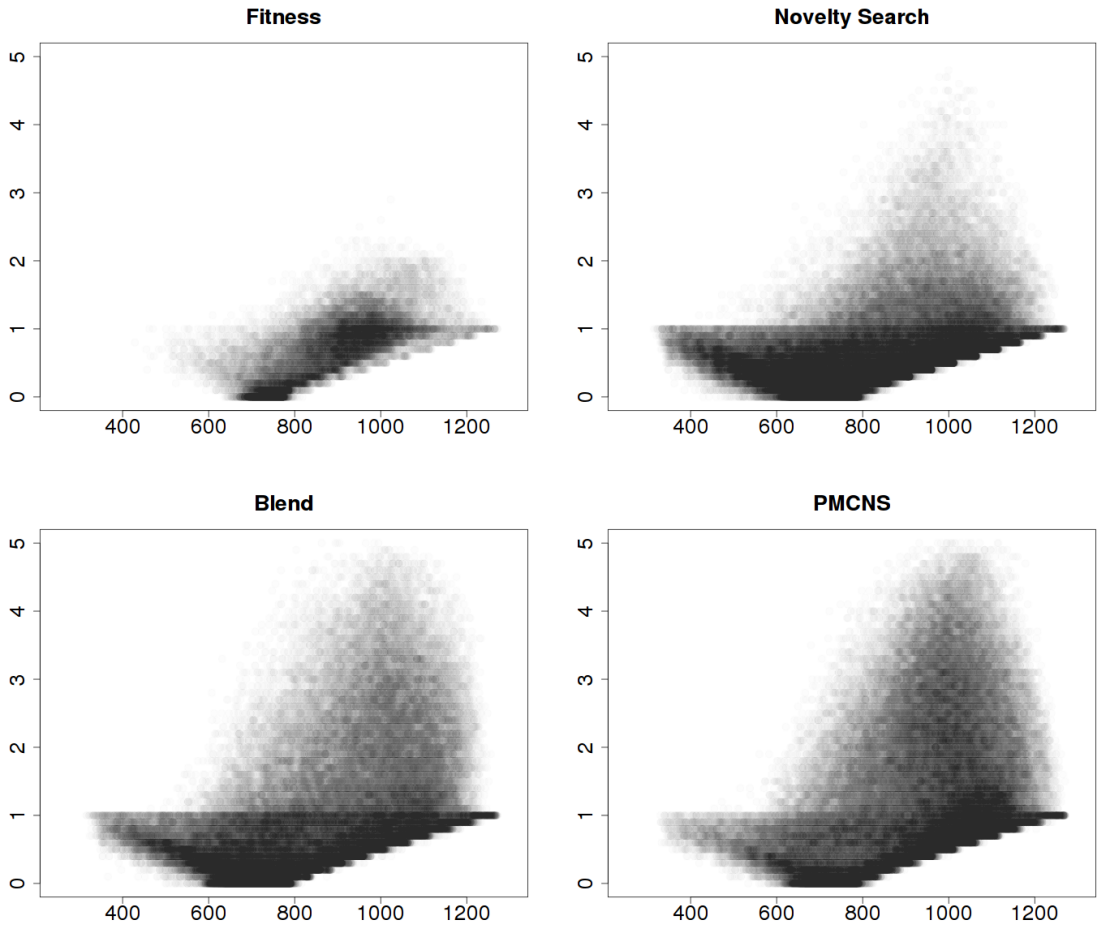

Figure 4.2: Behaviour space exploration with each evolutionary method, using the simple novelty distance measure. The x-axis is the average energy level of the robots alive, the y-axis is the number of robots alive at the end of the simulation. Each individual is mapped according to its behaviour. Darker zones mean that there were more individuals evolved with the behaviour of that zone.

An analysis of the explored behaviour space can provide a better insight into the strengths and weaknesses of each evolutionary method. To analyse the behaviour space, we used the same visualisation methods as in Section 3.2. Figure 4.2 and Figure 4.3 depict the behaviour space exploration with the simple novelty measure and the expanded measure, respectively.

The analysis of the behaviour space exploration confirms that PMCNS and linear blend have a greater focus on the behaviour zones associated with higher fitness scores, compared to pure novelty search. Also, it should be noted that the coverage of the behaviour space was not negatively affected in PMCNS and linear blend. These methods could still unveil the same behaviour diversity as novelty search alone, what changed was the effort they put in exploring each behaviour zone – the evolution focused less on the low fitness behaviours (with a low number of surviving

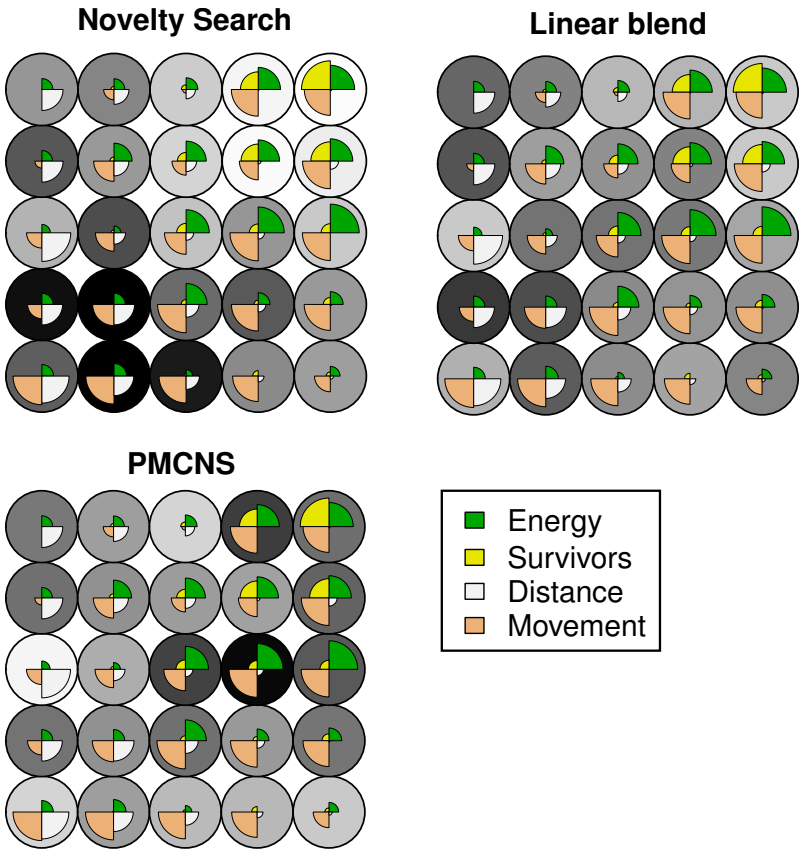

Figure 4.3: Kohonen maps representing the explored behaviour space with each evolutionary method, using the expanded behaviour measure. Each circle represents a behaviour pattern, depicted by the 4 slices of different colour. Each slice represents one component of the behaviour measure – the bigger the slice, the bigger the value of that component. The darker the background of a circle is, the more individuals were evolved with the corresponding behaviour.

robots) and more on the high fitness behaviours (high number of surviving robots).

The behaviour space exploration is not dramatically distinct between PMCNS and linear blend. However, PMCNS clearly has a greater focus in the behaviours with higher fitness values. It finds behavioural diversity where it is most relevant – in the zones of successful behaviours. It is interesting to note that although PMCNS might be viewed as technique to restrict the search space, it was actually able to find a broader behavioural diversity than novelty search alone. The explanation is that in this task, it is easier to find novel behaviours with low fitness scores than novel behaviours with high fitness scores, since there are 4 behaviour dimensions, and only one (number of surviving robots) is directly related to the quality of the solutions. Furthermore, this dimension is the most difficult to explore, as it requires more coordination in the robotic swarm. Consequently, novelty search exploits the other behaviour dimensions, and gets focused in low fitness behaviour zones. The growing minimal criterion creates an additional pressure to explore the behaviour dimensions

correlated with the fitness score, actually helping novelty search to explore new behaviour zones.

Both PMCNS and linear blend could find a broad diversity of successful behaviours, as it can be seen in Figure 4.2. For the same number of alive robots at the end there are behaviours with average energy ranging from 800 to 1150. Observing in action some of these successful behaviours confirms this diversity:

1. The robots go towards the charging station and stay there. When another one arrives, the first moves away from the station and returns some time later.

2. Similar to (1), but when they move away, they never go farther than the station sensor range (1 m).

3. The robots go towards the station, circle around it at a very close distance, and when their energy is below 1000 units, they go to the charging station and only leave when they are full.

4. Similar to (3), but they go to the charging station with energy bellow 400 units and only charge until 1000. Other combinations of thresholds for charging and leaving the station were also found.

The observable behaviour patterns of the solutions found by PMCNS and linear blend are not notably different from the patterns found by novelty search (see Section 3.2.4). However, the solutions evolved by PMCNS and linear blend are more fine tuned, with less frequent flaws, thus resulting in higher fitness scores. For instance, the robots are better at avoiding the walls, better at finding the charging station and returning to it, and can manage the conflicts in the charging station more efficiently. While with novelty search the best solutions only manage to keep 3 robots alive, with PMCNS and linear blend the best solutions can keep 4-5 robots alive. We consider that PMCNS and linear blend can effectively solve the task at hand.

4.2.2 Impact of Deception

To study how PMCNS and linear blend are influenced by the deceptiveness exhibited by the fitness function, we evaluated and compared their performance in the two setups presented in Section 3.2.2. As explained in the previous chapter, in the setup A the robots spend a fixed amount of energy, while in the setup B each robot spends energy proportionally to the speed of its wheels. Despite being intuitively similar, we have seen that setup B originates deception, while the setup A does not. In setup B, the fitness-based evolution usually converges to a very poor local maximum where the robots do not move in order to save energy.

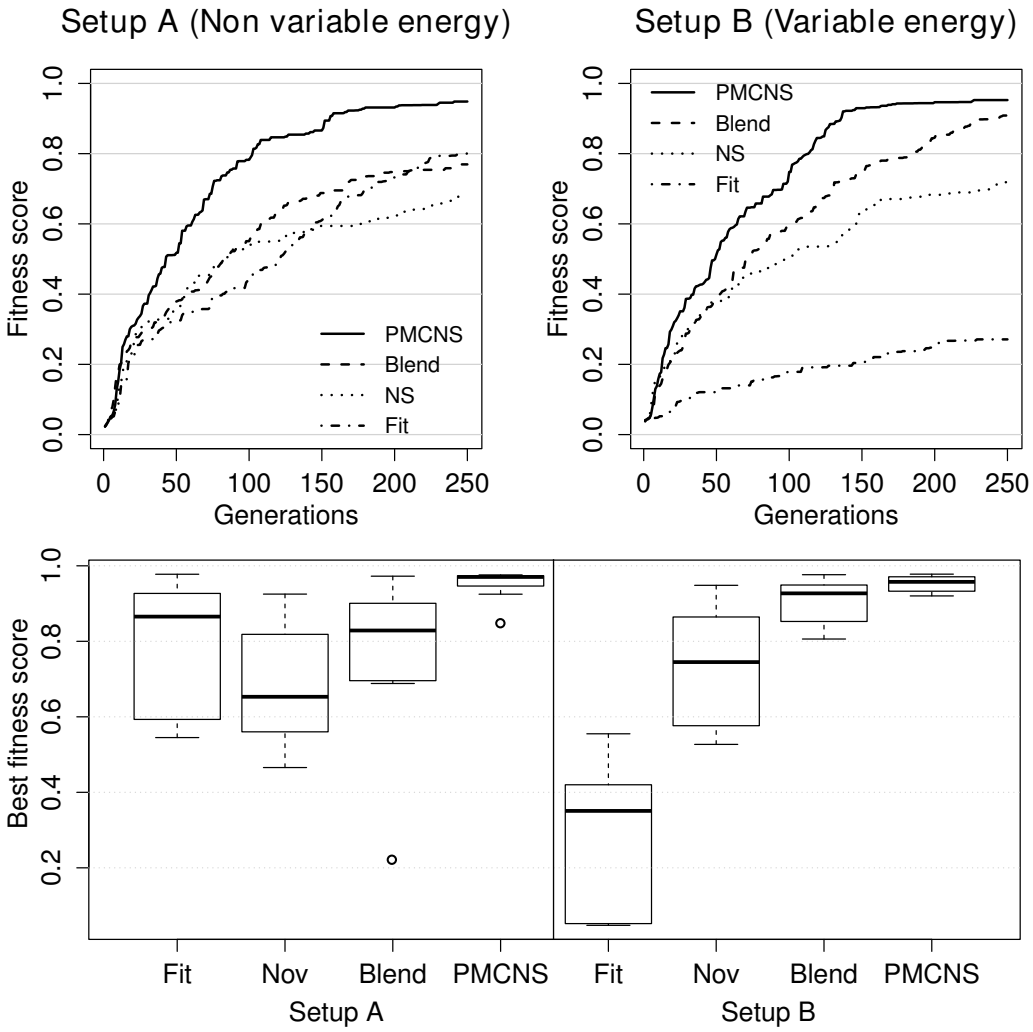

Figure 4.4: Top: Comparison of the fitness trajectories obtained with the two energy management experimental setups: with and without variable energy spending. The lines depict the fitness trajectory obtained with each evolutionary method, using the simple novelty distance measure. Bottom: Boxplots of the best fitness score found in each evolutionary run.

In this comparison, linear blend was run with $\rho = 0.75$, and PMCNS with $P = 0.5$ and $S = 0.25$. Only the simple novelty measure was used (average energy and number of surviving robots at the end). The results are presented in Figure 4.4.

In both setups, PMCNS significantly outperforms fitness-based evolution and NS (p-value < 0.01, Student's t-test). PMCNS is also significantly better than linear blend in both setups (p-value < 0.05). The boxplots show that PMCNS is more consistent than the other methods in finding good solutions. PMCNS does not seem to be affected by the deceptiveness of the fitness function, since its performance is very similar in both setups. The performance of the linear blend method is also satisfactory, being better than novelty search in setup B and as good as fitness-based evolution in setup A. Linear blend was also not negatively affected by the

deceptiveness in setup B.

The original MCNS algorithm was also tested by defining a fixed fitness threshold as the minimal criterion. Values of 0.03, 0.07, 0.10 and 0.20 were tested for the fitness threshold. Evolution was only able to bootstrap with a fitness threshold of 0.03. In this case, the fitness trajectory was slightly worse than pure novelty search. With greater fitness thresholds, the evolution could not find individuals with a fitness score that surpassed the threshold, and thus MCNS effectively acted as a random evolution, achieving on average a best fitness of 0.065.

4.2.3 Algorithm Parameters

The algorithm parameters in PMCNS and linear blend control the balance between behavioural novelty and the pressure from the fitness function. We tested several values of the parameters of PMCNS and linear blend, in order to assess the impact of the novelty-fitness balance in the performance of these evolutionary methods. All the experiments were run with the setup B (variable energy spending), and with the simple novelty distance measure. The parameters used in PMCNS and linear blend in the experiments described in the previous sections resulted from this analysis.

Linear Blend Novelty-Fitness Balance

The ρ parameter in the linear novelty-fitness blend establishes the ratio between the weight of the novelty score and the fitness score. Only two ρ values were tested (0.50 and 0.75). Fitness-based evolution corresponds to $\rho = 0$, and pure novelty search corresponds to $\rho = 1$. The results are depicted in Figure 4.5. The best performance was achieved with $\rho = 0.75$ (significantly better than all other setups with p-value < 0.01), which means that the novelty score has much more importance than the fitness score in the evaluation of each individual. This is in accordance with the results reported in (Cuccu and Gomez, 2011), where the best performance was achieved with $\rho = 0.8$.

The inferior performance verified with $\rho = 0.5$ can be explained by the deceptive nature of the fitness function, which leads the evolution to local maxima. As the weight given to the fitness score is higher, it is more likely that linear blend will be affected by the deceptiveness of the fitness function, and the reduced weight of the novelty score is not enough to divert the search away from local maxima.

PMCNS Minimal Criterion Strictness

The P (Percentile) parameter is the most important in the PMCNS algorithm. It defines the exigency of the minimal criteria, and consequently, the percentage of population individuals that receive a non-zero score. Three variations of P were

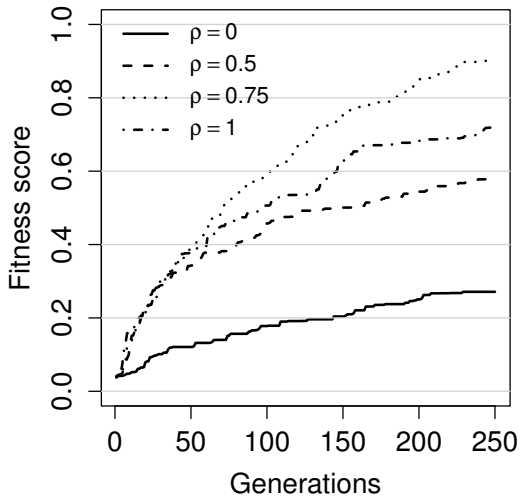

Figure 4.5: Fitness trajectories obtained with multiple values of ρ in linear blend. The higher the ρ value, the more weight novelty score has in the evaluation of the individuals.

tested: 25%, 50% and 75%. The experiments were run with a fixed smoothening parameter $S = 0.50$. Figure 4.6 shows how the P parameter affects the fitness trajectory, the progression of the minimal criteria, and the number of individuals that meet the minimal criteria.

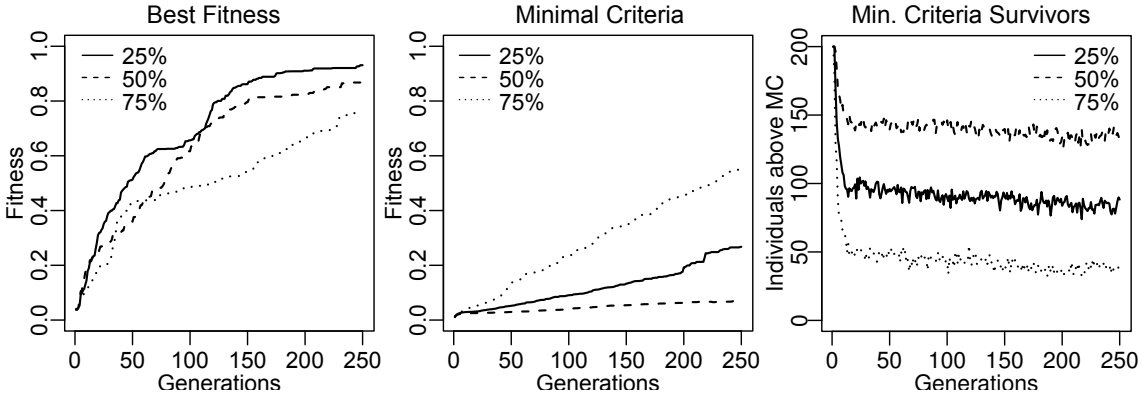

Figure 4.6: Left: how the P parameter of PMCNS affects the fitness trajectory. Middle: the progression of the minimal criterion value over the generations. Right: the average number of individuals above the minimal criterion in each generation.

We can see that a too strict minimal criteria ($P = 75\%$) is prejudicial to the evolution. Lower values are preferred, where just a small percentage of the population does not meet the minimal criteria. Analysing the behaviour space explored with each variant, we can see that the 50% variant have a greater focus in the high fitness behaviour zones than the 25%, but the 25% covers a slightly larger behaviour space. The 75% variant explored a very narrow zone of the behavioural space, and actually explored the high-fitness zones less because it got stuck in some

low-fitness zones, due to its high level of elitism.

PMCNS Minimal Criterion Smoothening

The smoothening parameter affects the rate of change of the minimal criterion threshold. A high S value will result in a fast adaptation of the minimal criterion threshold to the fitness profile of the current population. A low S value will delay this adaptation, indirectly reducing the fitness pressure, since the minimal criterion threshold has more resistance to increases, which allows low scoring individuals to remain longer in the population.

We tested three different values for S, maintaining a fixed percentile parameter of $P = 0.5$. The results (see Figure 4.7) show that although the smoothening parameter does not affect the performance of the evolution greatly, the impact can be significant. The fitness trajectory with the highest value (0.75 – fast threshold adaptation) was significantly worse (p-value < 0.05, Student's t-test) than the trajectory obtained with the other lower values. The lowest value (0.25 – slow threshold adaptation) was on average superior to the medium value (0.50), but it is not enough to claim statistical significance beyond p-value ≈ 0.06. These results suggest that lower smoothening values are preferred, however, small modifications in the smoothening value do not seem to have significant impact in the performance of the algorithm.

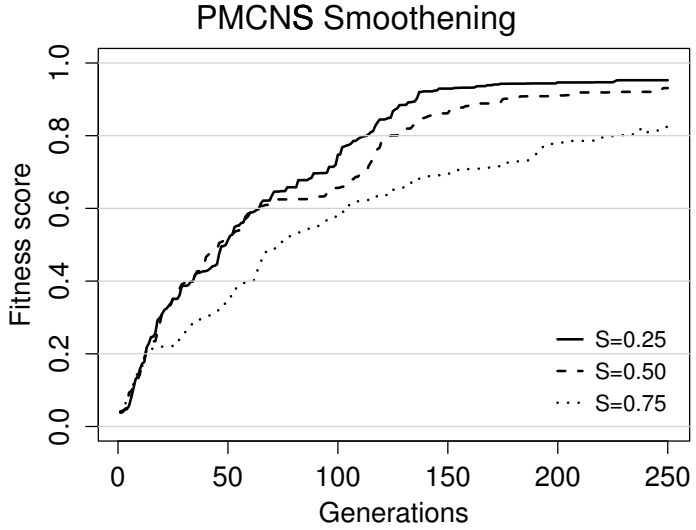

Figure 4.7: Fitness trajectories obtained with PMCNS, using multiple values of S and $P = 0.50$. The S parameter controls the speed of adaptation of the minimal criteria threshold. The higher the S value, the faster the adaptation is.

4.3 Aggregation Experiments

In the experiments with the aggregation task in the previous chapter, both novelty search and fitness-based evolution achieved good results, however, the evolved behaviours still displayed some flaws and there was margin for improvement. To strengthen our empirical study on the combination of novelty search and fitness, we revisit the aggregation task, using PMCNS and linear blend to evolve controllers for this task. With these experiments, we intend to study if PMCNS and linear blend can improve over novelty search and fitness-based evolution, even when novelty search achieves good results and the fitness function is not deceptive.

The experimental setup is exactly the same as the previous one (Section 3.1.3). Only the combined novelty measure is used, because it was the measure that delivered the best results in the previous aggregation experiments (see Section 3.1.6). This measure is composed by the average distance to the centre of mass and by the number of the robot clusters, both sampled throughout the simulation time.

4.3.1 Performance Comparison

We compared PMCNS and linear blend with the previous results obtained with pure novelty search fitness-based evolution. The results are depicted in Figure 4.8. PMCNS was run with the parameters $P = 0.50$ (50th percentile) and $S = 0.25$ (slow threshold adaptation). Linear blend was run with $\rho = 0.50$ (novelty score and fitness score with the same weight). These parameters are discussed below in Section 4.3.2.

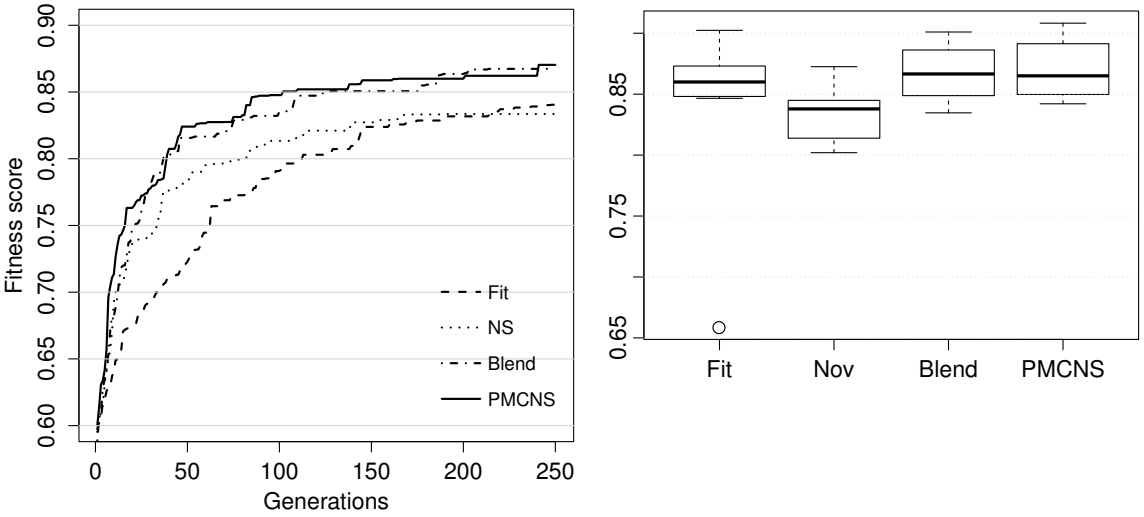

Figure 4.8: Left: Fitness trajectories obtained with each evolutionary method, in the aggregation experimental setup using the combined novelty measure. Right: Boxplots of the best fitness score found in each evolutionary run.

There are no significant differences between the fitness trajectories of PMCNS and linear blend. However, these methods improved over pure novelty search, achieving in the end significantly higher fitness scores (p-value < 0.01, Student's t-test). PMCNS and linear blend are on average superior to fitness-based evolution, but this superiority is only statistically significant until generation 150 (considering significance with p-value < 0.05).

Despite the slight differences in the fitness scores, the best solutions evolved by PMCNS and linear blend are essentially the same as the ones evolved by pure novelty search. As mentioned before, even the best solutions evolved by novelty search often displayed flaws, resulting in more than one aggregate of robots. In PMCNS and linear blend, the solutions evolved by novelty search were fine-tuned by the selection pressure from the fitness-function, resulting in solutions that are more frequently successful in achieving aggregation, with higher fitness scores. However, none of the evolutionary methods evolved a solution that *always* achieved a single aggregate in the end.

An analysis of the explored behaviour space provides a valuable insight into the evolutionary dynamics of each method (see Figure 4.9). There were behaviour zones (bottom left) explored in novelty search that were not explored by PMCNS and linear blend, however, these behaviour zones are associated with very low fitness scores. The diversity of possible solutions to the problem was not compromised (right column), and it was actually boosted. Both PMCNS and linear blend have a much greater focus in the high fitness behaviour zones (right column in the grid), when compared to pure novelty search.

As noted in the aggregation experiments in the previous chapter, fitness-based evolution spends much time exploring a rather uninteresting behaviour zone, associated with low fitness scores (top left corner). It suggests that fitness-based evolution gets stuck early in local maxima, which also explains why it has a slow bootstrap. This phenomena was also verified in linear blend (although with less magnitude), which is understandable since the fitness score acounts for 50% of the score assigned to each individual of the population. This reveals one weakness of the linear blend method: if the fitness function leads the evolution towards local maxima, linear blend can also get stuck in such local maxima. PMCNS, on the other hand, does not seem to be affected by this phenomena, since the fitness function does not directly contribute for the evaluation of the individuals of the population.

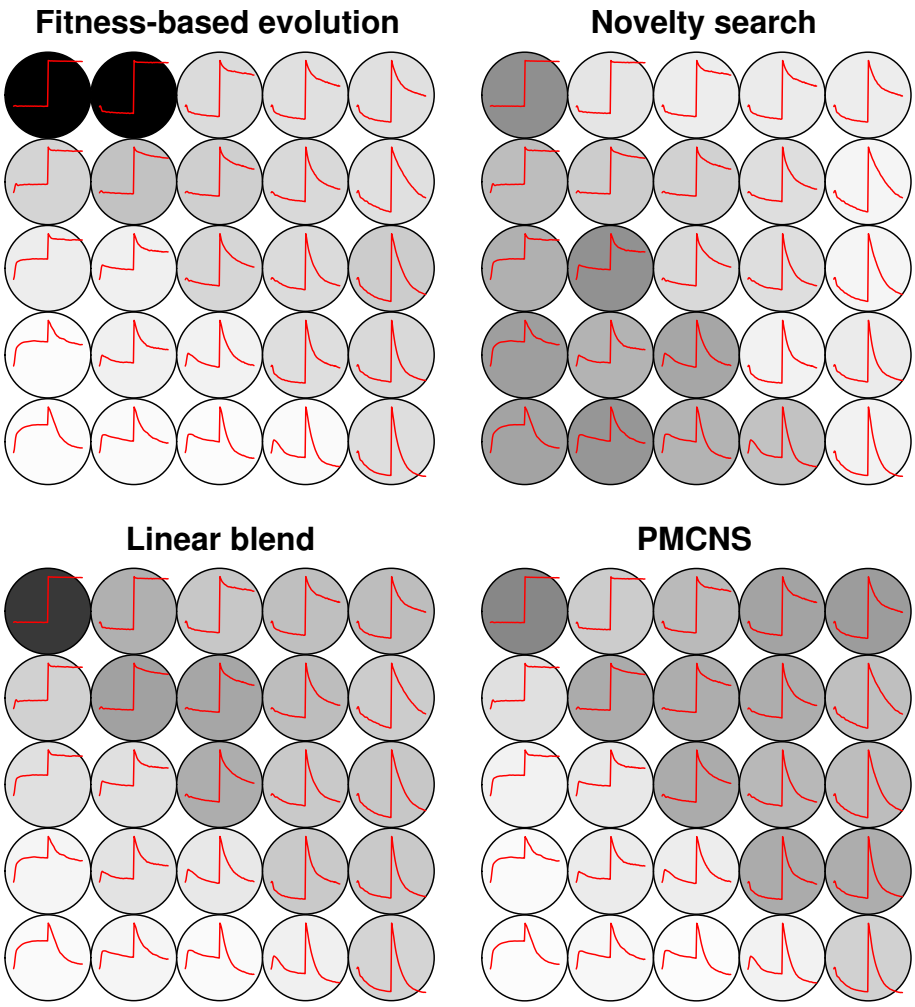

Figure 4.9: Kohonen maps representing the explored behaviour space with each evolutionary method. Each circle is a neuron that is characterised by the vector depicted by the line inside. The left half represents the average distance to the centre of mass over time, and the right half the average number of clusters over time. Each behaviour vector is mapped to the most similar neuron. The darker the background of a neuron is, the more behaviours were mapped to it. The solutions with higher fitness scores are mapped to the last column of the grid.

4.3.2 Novelty-Fitness Balance

To tune the parameters for linear blend and PMCNS methods, we tested multiple values of ρ in linear blend, and P in PMCNS. These parameters control the balance between the weight of the novelty score and the pressure from the fitness function. The results are depicted in Figure 4.10. In linear blend, $\rho = 0$ corresponds to fitness-based evolution, and $\rho = 1$ corresponds to pure novelty search.

Concerning linear blend, there are no significant differences between the fitness trajectory obtained with $\rho = 0.75$ and $\rho = 0.50$ (p-value < 0.05, Student's t-test). In PMCNS, there are no significant differences between the fitness trajectories obtained

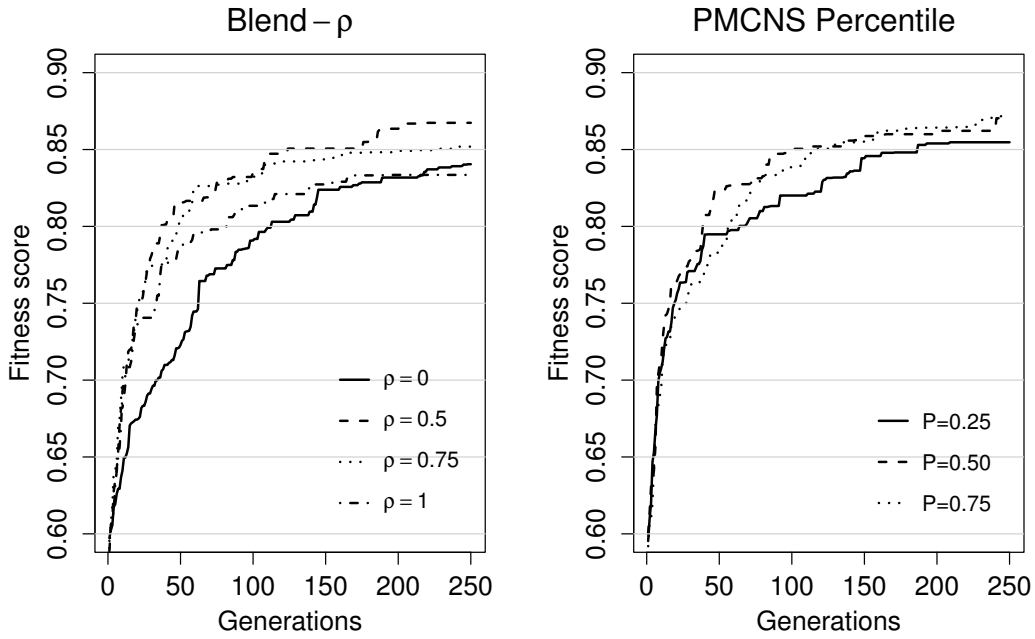

Figure 4.10: Left: The fitness trajectories obtained with linear blend using different ρ values (higher ρ gives more weight to the novelty score). Right: Fitness trajectories obtained with PMCNS, using different P values (The higher the P, the more exigent the minimal criteria is).

with $P = 0.75$ and $P = 0.50$, but with $P = 0.25$ the trajectory is significantly inferior to the other parameter values.

These results contrast to the ones obtained in the energy management experiments, where different parameter values had significant impact in the performance of the algorithm. In the energy management experiments, novelty search was clearly superior to fitness-based evolution, which prematurely converged to local maxima, achieving poor results. On the other hand, in the aggregation experiments, fitness-based evolution and novelty search are both able to achieve good results. This explains why the P and ρ parameters have a greater impact in the energy management experiments: the balance between fitness and novelty must be carefully adjusted so that evolution does not get stuck in local maxima that are created by the fitness function.

4.4 Discussion

We presented a new method, *progressive minimal criteria novelty search*, for combining fitness and novelty in evolutionary search. We extended *minimal criteria novelty search* by using a dynamic fitness threshold as the minimal criterion, pushing exploration of behaviour space towards zones of higher fitness. We experimented with the two tasks already presented in the previous chapter, aggregation and energy

management. The new algorithm was compared with MCNS with a fixed fitness threshold as minimal criteria, novelty search alone, fitness-based evolution, and a linear blend of novelty and fitness scores.

Effectiveness of Combining Novelty and Fitness

We showed that the combination of novelty and fitness is a promising approach for the evolution of controllers for swarm robotics. In all the experimental setups, the fitness trajectory obtained with PMCNS and linear blend were at least as good as novelty search and fitness-based evolution alone. In the energy management experiments, the improvement offered by the combination of fitness and novelty was pronounced, with the solutions being notably better. In the deceptive setup, the combination of novelty and fitness did not suffer from premature converge, as fitness-based evolution did. In the aggregation experiments, the improvement offered by PMCNS and linear blend was also statistically significant, but less noticeable in the quality of the evolved solutions.

Regarding the solutions evolved by PMCNS and linear blend, they were based on the same behaviour patterns found by novelty search alone. However, they were more fine-tuned and displayed less flaws, thus resulting in significantly higher fitness scores. In the energy management experiments, the behaviour space coverage was greater and more uniform with PMCNS and linear blend, when compared to novelty search alone. In the aggregation experiments, PMCNS and linear blend covered similar behaviour zones when compared to novelty search, however, in the first two, there was clearly a greater focus in the behaviour zones associated with higher fitness scores. This result suggests that the fitness function can actually help novelty search to explore the behaviour space, by creating an additional pressure to explore zones associated with higher fitness, which typically are more difficult to reach in complex tasks.

Progressive Minimal Criteria Novelty Search

PMCNS could effectively overcome the drawbacks of MCNS, while achieving a better performance. The fitness score was successfully used as minimal criterion. It was clearly advantageous to use a progressive minimal criterion, compared to a fixed minimal criterion in MCNS. The bootstrap problem was overcome, as the minimal criterion starts from the minimum fitness score, and only increases if the fitness profile of the current population also does. The smoothening of the growth revealed to be an important part of this process, with higher smoothening values delivering the best results (25% – the new threshold is 25% of the P-percentile value of the current population plus 75% of the last threshold).

The performance of PMCNS was superior to the linear blend of novelty and

fitness. In the energy management task, the fitness trajectory of PMCNS was significantly better than linear blend in both setups. In the aggregation task, it was as good as linear blend. PMCNS also explored more the behaviour zones associated with higher fitness scores. This is relevant because it suggests that PMCNS creates a pressure to evolve a diversity of successful individuals. As opposed to the linear blend, where the fitness function is always influencing the score of the individuals, PMCNS only imposes a minimal criterion for selection, and so the fitness function does not have any influence on the score of the individuals, which is based only on the novelty measure.

Algorithm Parameters

Regarding the balance between the weight of the fitness function and the weight of the novelty score in linear blend, our results suggest giving more weight to the novelty score (around 75%) delivers the best results, which is coherent with previous works with this method (Cuccu and Gomez, 2011). However, in the aggregation experiments, a lower weight of novelty score (50%) delivered the same performance as the higher weight (75%). This can be explained by the fact that the fitness function is more effective in guiding the evolution in the aggregation task than in the energy management task, and as such, the linear blend method can withstand a higher weight to the fitness function in the aggregation task.

With respect to the percentile parameter P in PMCNS, that determines the fitness threshold, our experimental results show that 50% (the median of the current population) offers the best performance in most situations. This means that, on average, half of the population receives its normal novelty score and the other half receives a null score. In the energy management task, a lower P value (25% – $^3/_4$ of the population receive normal novelty score) also delivered good results, and in the aggregation task, a higher P value (75% – $^1/_4$ of the population receive novelty score) also had good performance. This suggests a similar effect to the one found in linear blend, described above. When the fitness function has better capability of guiding evolution, a more exigent minimal criterion can be used. On the other hand, when the fitness function leads the evolution towards local maxima, a more permissive minimal criterion is preferable.

Chapter 5

Conclusion

The initial objective of this research was to explore if and how novelty search can contribute to the evolution of controllers for swarm robots. We consider that this objective was reached, as this thesis identifies and defends a number of advantages in using novelty search to evolve robotic swarms, instead of the traditional objective-based evolution. In the preliminary report of this thesis, a number of questions and challenges were identified. All those questions were addressed in this final document:

- How will novelty search perform in non-deceptive setups? We showed, in the experiments with both tasks, that the performance of novelty search in non-deceptive setups is similar to objective-based evolution. In deceptive setups, novelty search clearly outperforms it.

- How to analyse and visualise the behaviour space explored in novelty search? We showed that Kohonen self-organising maps provide an effective way of analysing behaviour spaces with high dimensionality.

- How to overcome conflation in order to increase the diversity of the evolved solutions? We showed that the combination of different behaviour measures can reduce conflation, and increase the diversity of solutions.

- How will novelty search perform with a more complex task? We experimented with a more challenging task, robots sharing a single energy charging station in order to survive. Novelty search performance was not significantly worse than objective-based evolution, but both failed to achieve excellent solutions to the task in the given number of generations. We showed that variations of novelty search that combine novelty with fitness might be more appropriate in these situations.

Beyond the initial objectives and challenges, we also proposed a new method (Progressive Minimal Criteria Novelty Search) for combining novelty search with

the fitness function. This method is based on the progressive restriction of the behaviour space, and was compared with other variants of novelty search.

5.1 Summary of the Contributions

The main contribution of our research was the study of novelty search based evolutionary techniques applied to the evolution of controllers for swarms of robots. To conduct this study, we followed an experimental approach, making evolutionary runs with different evolutionary techniques, parameters, and tasks. Detailed data was gathered in each evolutionary run to allow for an analysis of the evolutionary dynamics. The experimental framework itself was the first contribution of this thesis. A new experimental framework was developed based on a pre-existing 3d robotics simulator (Simbad), and on an implementation of NEAT (NEAT4J). The framework is highly modular, allowing the definition of new environments, robots, evolutionary techniques, and so forth, without having to modify existing code.

Our experimental study was based on two distinct swarm robotic tasks: i) an aggregation task, where the robots start randomly distributed in a closed environment and should form a single aggregate; and ii) an energy management task, where the robots lose energy over time and should coordinate themselves in order to allow periodical access to the single charging station in the environment. While the first task is widely studied in the context of evolutionary robotics, the energy management task is described in the literature, but only one work addresses it from an evolutionary perspective, using an oversimplified experimental setup. The evolution of robot controllers to perform the energy management task with resource conflict was an original contribution from this thesis. Some of the solutions evolved to the energy management task are similar to solutions proposed in the related work (non-evolutionary).

A secondary contribution of this thesis was the use of NEAT in evolutionary swarm robotics. NEAT is a neuroevolution method that evolves both the topology of the network and the weights of the connections. Previous works in this domain only describe the use of neuroevolution methods that start from a fixed network topology and evolve only the connection weights. Although this was not the focus of this thesis, and we did not establish experimental comparisons with other neuroevolution methods, we showed that NEAT can be effective in evolving controllers for swarms of robots. The complexification mechanism in NEAT brings interesting possibilities in this domain, as the evolutionary process produces gradually more complex forms of self-organisation throughout the evolution.

Our experimental results showed that novelty search can be an effective way to evolve controllers for robotic swarms. Novelty search proved to be effective

in overcoming deception in the swarm robotics domain, avoiding local maxima where fitness-based evolution got stuck. We established detailed comparisons of novelty search with the classical fitness-based evolution. In non-deceptive setups, where fitness-based evolution always succeeded, novelty search displayed a similar performance to fitness-based evolution, in terms of the fitness score of the evolved solutions. In deceptive setups, novelty search was clearly superior. Novelty search was particularly successful in bootstrapping the evolution, avoiding premature convergence and reaching higher fitness scores sooner in the evolution. We also showed that novelty search can evolve solutions with less complex neural networks. Our results represent a valuable contribution to the field of evolutionary swarm robotics, because the current literature describes significant difficulties in evolving robotic swarms for complex tasks. Our experiments suggest that the evolution of swarm behaviours is generally prone to deception, and thus novelty search presents as a promising alternative to overcome this difficulty.

Another contribution brought forth by the experiments in this thesis was the use of novelty search as a way to unveil a broad diversity of collective behaviours. Previous evolutionary swarm robotics studies did not focused on the discovery of behavioural diversity. However, this is an interesting topic because in collective robotics there is typically a broad range of behavioural possibilities, brought by the possible interactions among the robots, and between the robots and their environment. Revealing some of these possibilities can lead to unexpected forms of self-organisation and different solutions for the same problem, which can be valuable for the experimenter. For instance, novelty search could evolve a class of solutions for aggregation that is not described in the related work, but can be found in the biological world.

In this thesis, we provided insight on how to devise behavioural measures for robotic swarms, in order to overcome conflation and explore the behavioural possibilities. We showed that combining different novelty measures can be an effective way of improving the performance of novelty search. However, one should avoid increasing the dimension of the behaviour space with measures that are weakly related with the task that one is trying to solve. In this case, our results showed that novelty search can start to explore behaviour zones that are not relevant for solving the task, and fail to adequately explore the important zones. We also presented a technique for visualising high dimension behaviour spaces, with Kohonen self-organising maps. This visualisation proved useful to analyse and compare the exploration of the behaviour space, and to understand the evolutionary dynamics in novelty search.

As stated above, our results showed that novelty search can struggle to find high fitness solutions when the behaviour space has dimensions that are weakly

related with the objective. To overcome this issue, we also studied novelty search variants that bring together behavioural diversity and the fitness function. We proposed a new algorithm for combining novelty search with fitness evaluation, named Progressive Minimal Criteria Novelty Search (PMCNS). This new algorithm progressively restricts the search space by establishing a fitness threshold that the individuals must meet in order to be selected for reproduction. This threshold is dynamic, starts with a null impact and increases progressively as the fitness profile of the population also does.

Our experimental results showed that PMCNS is effective in guiding the behavioural exploration towards behaviour zones of higher fitness, without compromising novelty search capabilities of overcoming deception and finding diversity of solutions. We compared PMCNS with linear blend of novelty and fitness scores, another method described in the literature for combining novelty search with objectives. The performance of PMCNS was superior to the other tested evolutionary methods, achieving consistently good solutions at the end. Regarding the balance between novelty search and the pressure from the fitness function, our results suggest that when the fitness function alone has better capability of guiding evolution, a higher pressure from the fitness function is tolerable. Otherwise, novelty search should be the predominant force in guiding evolution.

5.2 Future Work

We showed that novelty search has the potential to find a great diversity of solutions for a given problem. However, this diversity of solutions is hard to unveil by hand, since only looking at the solutions with higher fitness scores can be restrictive: there can be solutions with lower fitness scores but with more interesting behaviour patterns. An interesting line of work would be to develop methods for automatically identifying solutions with diverse forms of self-organization. This could work as a system for suggesting to the human experimenter possible solutions for a given problem.

Our experiments showed that combining novelty measures can be an effective strategy for reducing conflation. We used a naive approach for this combination: a simple concatenation of the behaviour characterisations. This approach has some limitations, as the behaviour characterisations need to be previously normalised and must have the same length. It could be studied more elaborate strategies for combining novelty measures. One possibility would be to devise an algorithm that automatically normalises the length and range of the behaviour characterisations, so that all of them have similar contributions for the novelty measure. Another possibility would be to use a multi-objective evolutionary algorithm, with each

novelty measure as a separate objective.

Our results suggested that in novelty search some dimensions of the behaviour space are more easily explored than others, which might be prejudicial for the behaviour diversity. To overcome this issue, we could give less weight to the easily-explored dimensions, so that there is an additional pressure to explore the other dimensions. These weights could eventually be automated during evolution, based on how much each dimension is explored during the search.

We faced some challenges when hand-designing the behaviour characterisations, due to the wide variety of behaviour possibilities in swarm robotics. Conflation is hard to avoid, and a very detailed behaviour characterisation can lead to a less effective evolution. There are variants of novelty search that work with generic behaviour measures (see Section 2.8), where the behaviour characterisation is for example the mapping between the sensors of the robot and the actuators. It would be interesting to study how generic behaviour measures could be adapted to work in the collective robotics domain, and assess if they can be a viable alternative to domain dependent novelty measures.

In PMCNS, the functioning of the novelty archive was not modified, individuals can still be added to the archive even if they do not meet the minimal criteria. It would be interesting to experiment another variant where such individuals are not added to the archive. This could reduce the size of the archive considerably, and increase the computational performance of the algorithm.

A natural next step of our research would be to apply novelty search and PMCNS to more complex and demanding swarm robotics tasks. It would be valuable to provide insight on what are the limits of novelty search, in regard to the complexity of the goal that one wants to achieve.

5.3 Conclusion

This thesis showed that novelty search is a promising alternative for the artificial evolution of controllers for swarm robots. This was empirically demonstrated with two different swarm robotics tasks, and a number of different experimental setups for each task. Novelty search could overcome deception, excelled at bootstrapping the evolution, evolved solutions with less complex neural networks, and unveiled a wide variety of solutions for a given task. Combining novelty search with objective-based evolution can further increase the potential of this approach, by focusing novelty search in the most interesting behaviour zones. Our contribution represents another step towards the design of robotic swarms capable of performing complex, real world tasks.

Appendix A

Simulation and Evolution Workbench

In order to conduct the experiments necessary in our empirical study, an adequate experimental framework was needed. We implemented a new framework (named *EvoSimbad*) based on pre-existing libraries. Although the framework was naturally tailored to fit the requirements of our study, the implementation allows the configuration of a wide range of experiments. This framework can thus be used outside the scope of this thesis, in other evolutionary robotics studies.

Experiments in evolutionary robotics typically require two major software components that are mostly independent:

- A *robotics simulator*, where it must be possible to define the environment topology and rules, as well as the robots along with its capabilities of sensing and acting upon that environment.
- The *evolutionary process*, responsible for the running the evolutionary algorithms. It will use the robotics simulator in order to evaluate the evolved solutions.

These two components are connected through i) the neural controller, which is evolved by the evolutionary algorithm and used by the robots in simulation, and ii) the evaluation function, responsible for measuring the fitness and the behaviour of the robots in simulation and returning it to the evolutionary algorithm. In the context of this thesis, a software application was developed that comprises these two components, taking advantage of other existing tools, namely Simbad (Hugues and Bredeche, 2006) for robotics simulation, and NEAT4J for neural network evolution.

A.1 Application Features

The developed tool offers the following possibilities and features.

i. Easily extensible design, allowing the definition of new components without having to modify any previously existing code.

ii. Supports the definition of simulated environments with possibly different topologies and obstacles of different shapes.

iii. Supports the definition of robots with a great variety of customizable sensors and actuators.

iv. The environments can vary in each simulation, such as different obstacle positions or variable initial positions of the robots.

v. Simulator with realistic physics, including collisions effects and realistic movement.

vi. Supports the definition and usage of distinct evolutionary algorithms, including novelty search.

vii. Supports the definition and customisation of fitness functions.

viii. Production of detailed logs about the evolution, along with the persistence of individuals (robots controllers) from the population.

ix. Visualisation of the robots behaviour in their environment, using any previously evolved controller.

x. Each experiment setup is intelligible and flexible, and allows the customisation of experiments without having to modify the source code of the application.

xi. Allows the distribution of the processing power, to further accelerate the attainment of experimental results.

A.2 Tools and Libraries

The application was developed in Java 7[1] programming language. A few open source libraries were used, taking advantage of existing technologies and thus speeding the development of the application.

[1] JDK 7 – `http://jdk7.java.net`

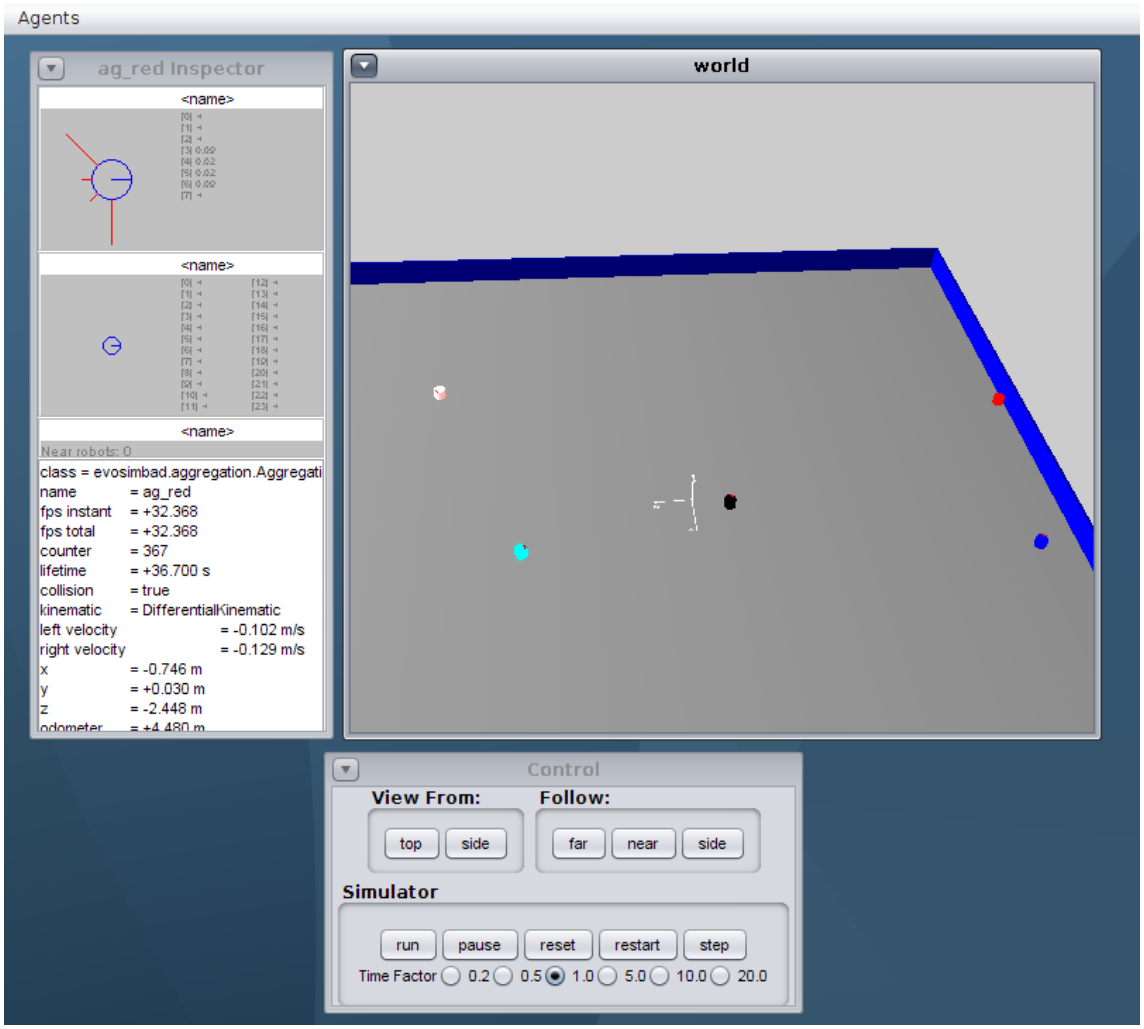

Figure A.1: Simbad user interface. The 3D window displays the robots and environment (top right), there is a window to control the simulation flow (bottom), and a window to monitor the sensors of the robot (left)

Simbad[2] (Hugues and Bredeche, 2006) is a Java 3d robot simulator for scientific and educational purposes. It provides 3D visualisation and sensing, single or multi-robots simulation, vision sensors, contact sensors, range sensors and a Swing user interface for control of the simulations (see Figure A.1). It enables programmers to write their own robot controllers, modify the environment, use the available sensors and define new ones. Simbad is used in the application for the robotics simulation, including the environment and robots definition, sensors and visualisation of the evolved behaviours.

NEAT4J[3] is a Java open-source framework that implements the NEAT algorithm as proposed in (Stanley and Miikkulainen, 2002). NEAT4J allows a complete

[2]Simbad 3d Robot Simulator – `http://simbad.sourceforge.net/`
[3]NeuroEvolution for Augmenting Topologies for Java – `http://neat4j.sourceforge.net`

customisation of the parameters of the NEAT algorithm. It is used in the application as a base for the implementation of the evolutionary methods.

JPPF[4], short for Java Parallel Processing Framework, is a Java framework that enables applications with large processing power requirements to be run on any number of computers, in order to dramatically reduce their processing time. It is very well documented and provides a high level of abstraction with very little configurations needed. JPPF comprises four major components: The grid node application that must be running in the computers that will perform the computations; the grid server application that distributes the load to the grid nodes; the grid client library that sends jobs to the server to be executed in the grid; the administration console that provides a graphical user interface for managing the nodes, servers and jobs.

Some libraries of *Apache Commons*[5] are also used throughout the application. Apache Commons is a project dedicated to the creation and maintenance of reusable Java components. It complements the Java API by adding many useful classes and methods that helps speeding up the software development.

A.3 Architecture and Design

To achieve the desired flexibility, many components of the application can be customizable and provided by the user. It is thus necessary an architecture that allows the implementation of new components without having to modify the existing code. This was achieved through the definition of a set of core interfaces that define each component, and classes that work with the functionality described by the interfaces, linking everything together.

The specific implementations of the components that will be used in each experiment are specified by the user in the configuration, and are loaded at runtime and provided to the other software modules that need them. Such architecture is depicted in Figure A.2, where it is specified the boundary between what is provided by the user and what is settled in the core of the application.

The following components were defined via interfaces or abstract classes, establishing the extension points of the application:

- `AgentGenerator`: Responsible for creating the set of robots that will be put in each simulation.

- `EnvironmentGenerator`: Creates the simulated environment where the simulation will occur.

[4]Java Parallel Processing Framework – `http://www.jppf.org/`
[5]Apache Commons – `http://commons.apache.org`

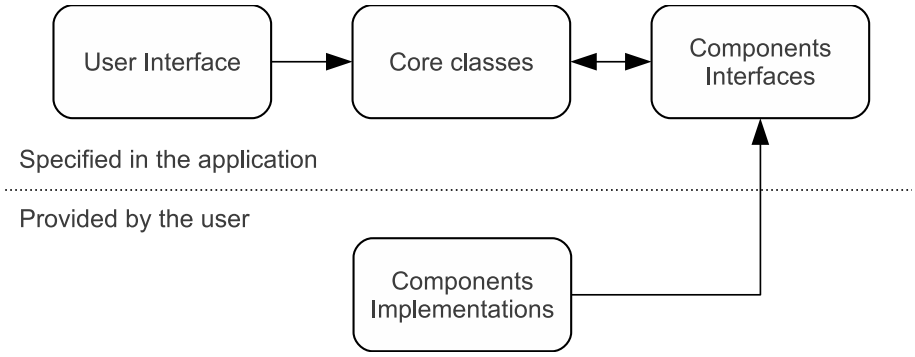

Figure A.2: The architecture of the application. The core classes of the application use the implementations provided by the user through well established interfaces. The user interface is well separated from the rest of the application logic.

- `AgentPlacer`: Given an environment and a number of robots to place, generates the initial positions and directions for them.
- `EvaluationFunction`: Gathers measures along each simulation for evaluating the performance of the robot(s).
- `SimulationBuilder`: Using the previously described components, prepares a set of simulations that will evaluate a robot controller.
- `Simulator`: Runs simulations with a given controller and returns the evaluation of the controller according to the `EvaluationFunction`. Uses the `SimulationBuilder` to generate the required simulations.
- `EvolutionMethod`: Defines the evolutionary method that will drive the evolution of the controllers, using the *Simulator* for the evaluation of the population.
- `Logger`: Defines a set of methods for recording logs of the evolution and controllers generated by it.

The implementation of each component is loaded in runtime by the class `ComponentLoader` and made available to the other software components. The articulation between these classes is specified in the class diagram in Figure A.3, where the key classes and methods are presented.

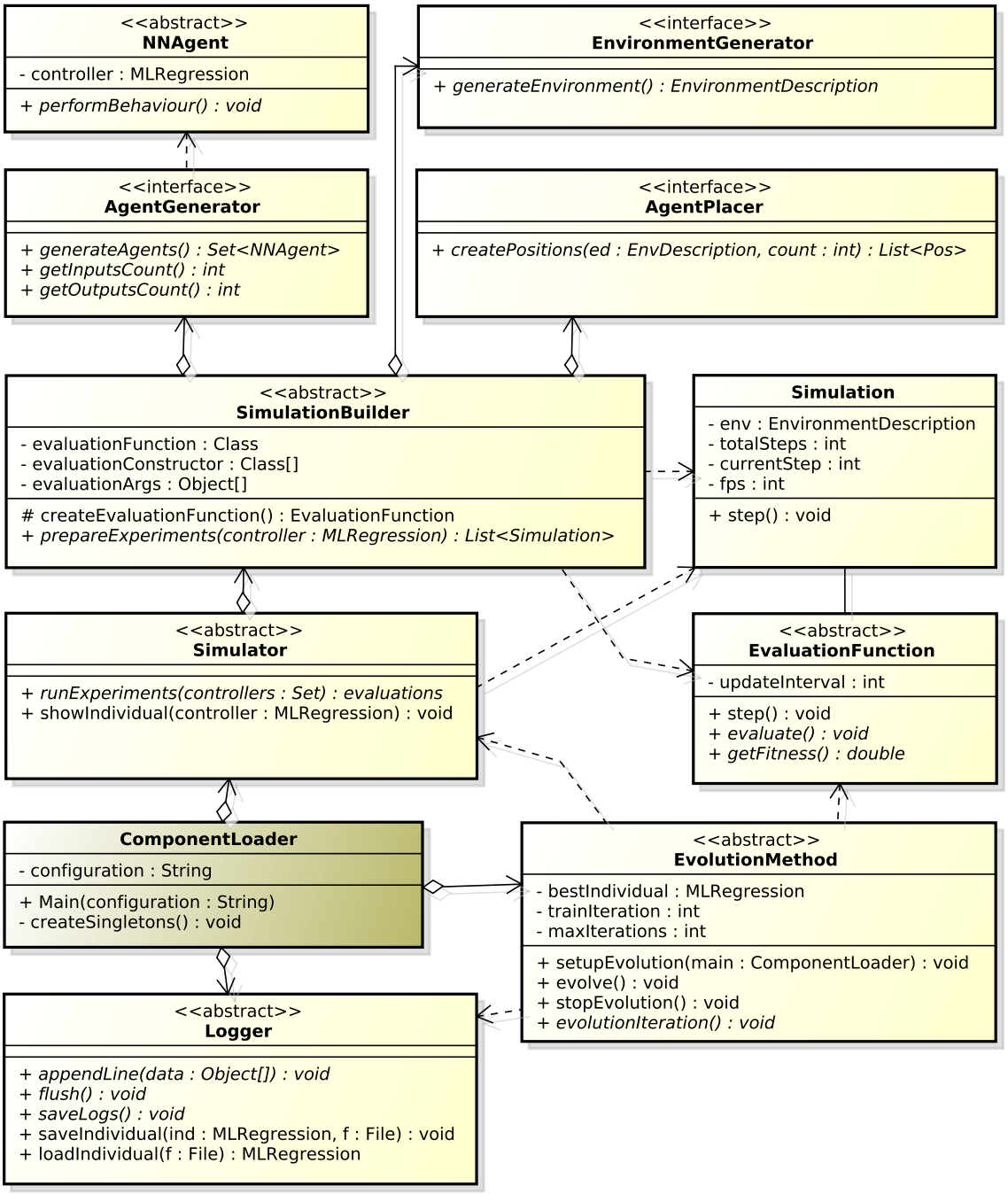

Figure A.3: UML class diagram of the core classes and interfaces of the application. A few auxiliary classes, methods and attributes have been omitted from the diagram.

A.4 Implementation

Configuration and Class Loading

Each experiment can be configured via a text file, where each component of the application is instantiated through the specification of its class and constructor arguments. A typical configuration file looks like this:

Listing A.1: A typical configuration file.

```
environment = evosimbad.commons.SquareGenerator(5)
agents = evosimbad.aggregation.AggregationEpuckGenerator
   (3, 10, 0.25, true, false)
simulator = evosimbad.simulation.GridSimulator()
simulationBuilder = evosimbad.simulation.SimpleSimulationBuilder
   (5000, 10, 10)
evolution = evosimbad.evolution.FitnessNEAT(250)
savePopulation = true
PROBABILITY.MUTATION = 0.1
PROBABILITY.CROSSOVER = 0.25
POP.SIZE = 200
agentPlacer = evosimbad.commons.RandomAgentPlacer(0.1, 0.7)
logger = evosimbad.commons.CSVLogger
   (/home/jorge/Temp, /home/jorge/Aggregation/Fit)
evaluation = evosimbad.aggregation.AggregationFitness(100)
```

For each of the components described in Section A.3, the user specifies the name of the component (e.g. `environment`), the fully qualified name of the class that will be instantiated (e.g. `evosimbad.commons.SquareGenerator`), and the constructor to be used in the instantiation (e.g. `(5)`). During the setup of the experiment, the class is loaded in runtime through its fully qualified name and it is checked if it implements the interface of the corresponding component (or extends the correct abstract class). Then, using Java Reflection, the constructors of that class are iterated until it is found one that fits the given arguments. The class is then instantiated and that instance persists until the end of the experiment. An exception to this is the `EvaluationFunction`, which requires a new instance for each simulation. Instead of keeping a single instance, the right constructor and the arguments are kept and used every time a new instance is needed. All these class loading mechanisms are implemented in the `ComponentLoader` class. It is also possible to define other options by key and value, for example `PROBABILITY.MUTATION = 0.1`. These are stored in a map and can be read by any component in the application at any time.

Common Components

A number of implementations of components were made that are useful in most of the experiments conducted. Some of these are presented next.

`VariableAgentGenerator implements AgentGenerator`
Abstract implementation that generates a set of homogeneous agents. The number of generated agents is random, varying within user-defined limits. Each agent is assigned a different name and colour.

`Epuck extends NNAgent`
Represents a robot similar to the e-puck educational robot (Mondada et al., 2009), including active IR sensors for the detection of obstacles, and passive IR sensors for the detection of other robots.

`SquareGenerator implements EnvironmentGenerator`
Generates a square arena with walls around it. The environments are implemented extending Simbad classes. An example of such environment can be seen in Figure A.1.

`RandomAgentPlacer implements AgentPlacer`
Places the robots in the environment with random positions and directions, keeping a given minimum distance between them.

`NEATEvolution extends EvolutionMethod`
It was implemented an abstract class NEATEvolution that represents evolutions guided by the NEAT algorithm, described in Section 2.4, using the implementation available in the NEAT4J library.

`SimpleSimulationBuilder extends SimulationBuilder`
Generates N simulations for the evaluation of each individual, using the user specified implementations of `AgentGenerator`, `EnviornmentGenerator`, `AgentPlacer` and `EvaluationFunction`. N is a constant value specified by the user in the configuration.

`MultiThreadedSimulator extends Simulator`
This simulator implementation runs multiple simulations at the same time in the same computer, using a number of pre-specified cores, thus taking full advantage of the multi-processor architectures.

`GridSimulator extends Simulator`
This implementation runs the simulations in a JPPF grid, which will be detailed in Section A.5.

`CSVLogger extends Logger`
This implementation generates CSV (Comma Separated Values) files with the recorded logs, and bundles everything in an experiment folder together with the configuration used and the saved individuals. CSV files are practical because they can be read by many spreadsheet and statistical analysis applications.

`ImageTracer`
Traces the trajectories of the robots in a simulation and produces a single image with those trajectories. These images are useful for a quick and static analysis of the swarm behaviour.

User Interface

The user interface is shown in Figure A.4. The main menu (1) provides options for i) save and load experiment configurations, ii) control the flow of the experiment, iii) save and view individuals, and iv) manipulate the logs. The configuration pane (2) provides a text area for writing and editing experiment configurations. The progress bar (3) provides visual information on the progress of the experiment. The statistics bar (4) provides useful real-time information, such as the best fitness found so far, the elapsed time in the experiment and the time remaining. The output window (5) shows a log sent by the evolution method, with statistics such as the average fitness in each generation.

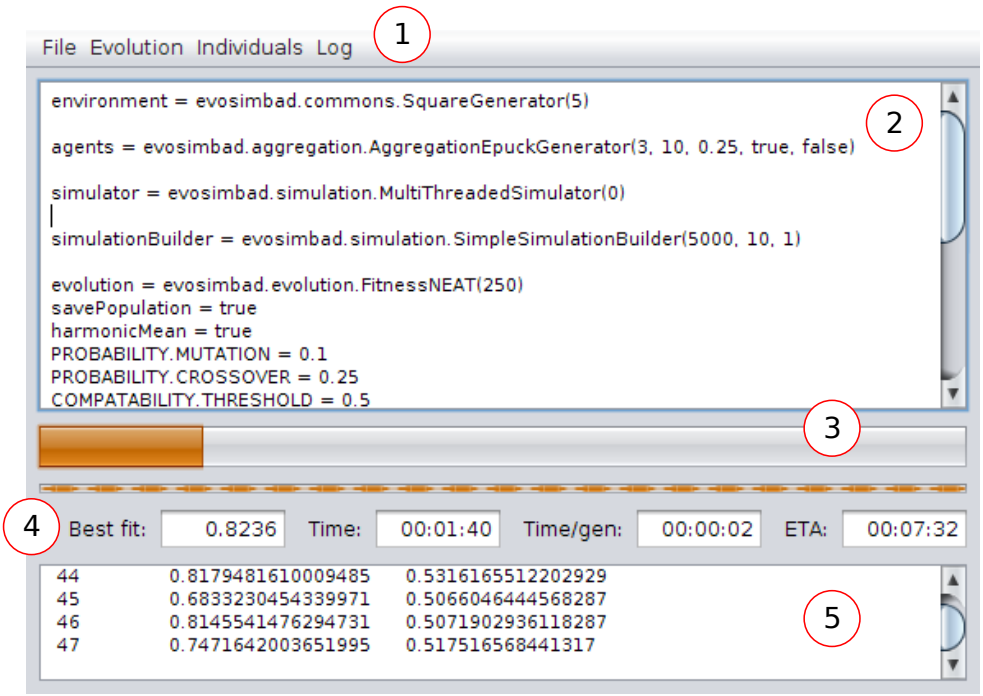

Figure A.4: The EvoSimbad graphical user interface.

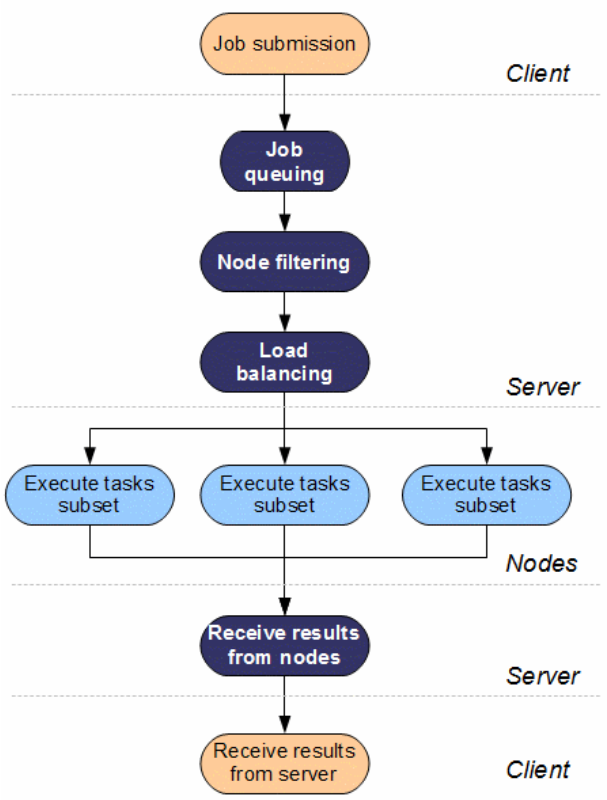

Figure A.5: The common flow of a job's execution in the grid using JPPF. This chart shows the different steps involved in the execution of a job, and where each of them takes place with regards to the grid component boundaries. Figure from `http://www.jppf.org`.

A.5 Distributed Computing

The most significant computational cost in the evolutionary process corresponds to the simulations needed to evaluate each individual in the population. A possible way of dealing this cost is by splitting the computation into multiple computers. This was achieved with the Java Parallel Processing Framework (JPPF). Figure A.5 explains the common flow of JPPF when executing a job on the grid and the function of each component.

To parallelize the evaluation of the population, the evaluation of each population individual is isolated in a distinct task. The set of tasks (one for each individual) constitutes a job. In each generation of the evolutionary algorithm, a job is submitted to the JPPF grid for the evaluation of the current population. It is thus necessary to wait for the completion of the entire job before moving on with the evolutionary algorithm.

JPPF requires tasks to implement the *Serializable* interface, which excluded the most natural solution of modelling each task as a simulation, as the `Simulation`

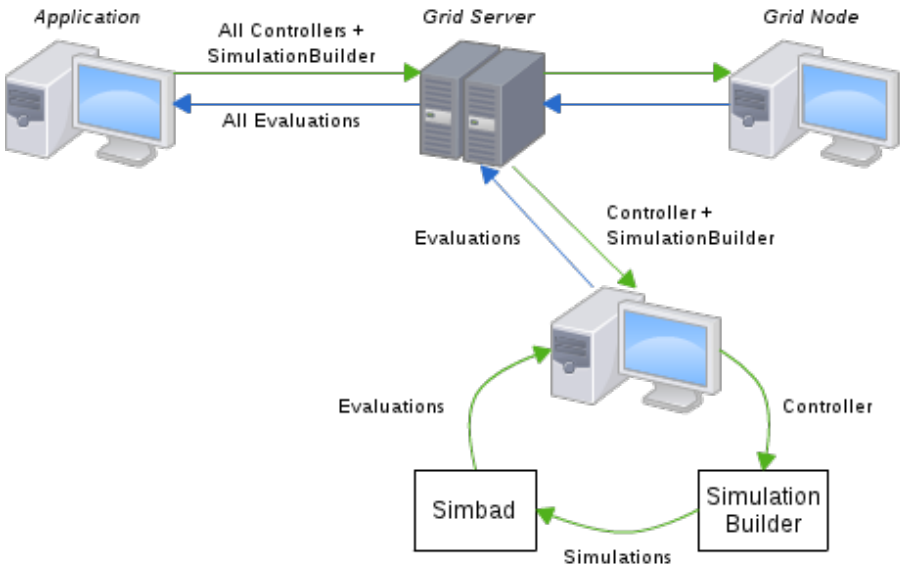

Figure A.6: The flow of information when distributing the evaluation of the population individuals (controllers).

object encapsulates an `EnvironmentDescription` object (from the *Simbad* library) that is not *Serializable*. To overcome this issue, instead of sending the `Simulation` objects, it is sent the instructions on how to build these simulations. Each task consists of a `NeuralNetwork` object that is the controller to be evaluated, and the `SimulationBuilder` object that will build the simulations to evaluate it. The result of the execution is `EvaluationFunction` objects produced by the simulations. The diagram in Figure A.6 summarises this flow of information.

To assess the improvements offered by the distribution of the work load, a few tests were made with a configuration of a typical evolution. With the configuration used, each generation requires the execution of 1000 simulations, each one with 4000 time steps, corresponding to 400s of simulated time. The performance test was made by first executing the application locally and then executing it on the grid, with a variable number of computers linked to the grid. The Table A.1 exposes the results, showing the average time spent in an evolution iteration with different grid setups.

If we compare the increase in processing power with the increase of performance (decrease in processing time), as showed in Figure A.7, we can see that there is some deceleration in the performance improvements. These losses can be explained by the network overhead of distributing the tasks and waiting for every computer to complete theirs. These results confirm that the system escalates with more computers. However, as the number of grid nodes grow, it can be expected greater system overhead, and the should not increase linearly with the total processing power.

Table A.1: Results of distribution of the computation.

Setup	Number of cores	Time per generation
1 Intel Core 2 Duo 3GHz	2	159,2 s
2 Intel Core i7 3,4GHz	8	40,2 s
3 Intel Core 2 Duo 3GHz 2 Intel Core i7 3,4GHz	14	29,1 s
6 Intel Core 2 Duo 3GHz 2 Intel Core i7 3,4GHz	20	22,9 s
7 Intel Core 2 Duo 3GHz 2 Intel Core i7 3,4GHz 1 Intel Xeon 3GHz	26	20,6 s

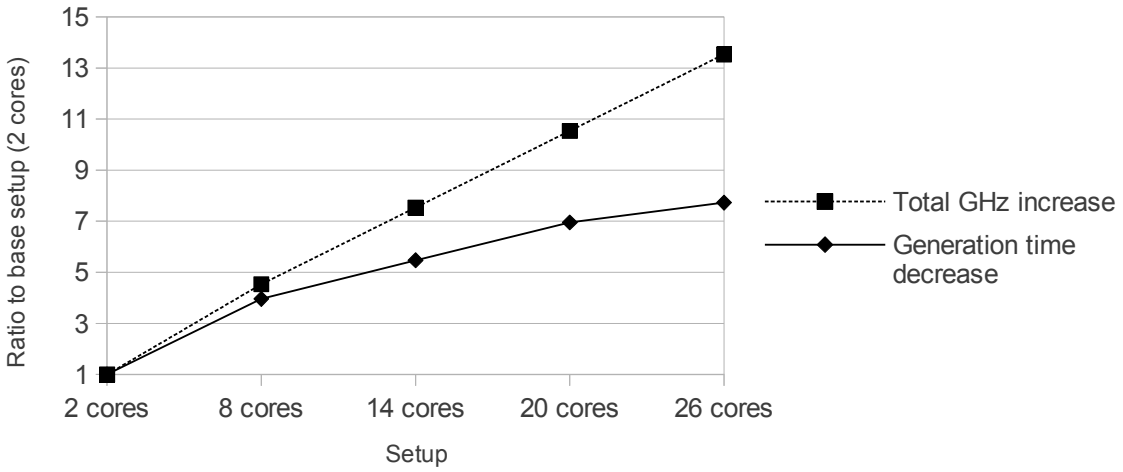

Figure A.7: The increase in processing power compared to the increase in system performance, using the data in Table A.1.

Bibliography

E. Bahgeçi and E. Şahin. Evolving aggregation behaviors for swarm robotic systems: A systematic case study. In *Swarm Intelligence Symposium*, pages 333–340. IEEE, New York, NY, 2005.

Gianluca Baldassarre, Stefano Nolfi, and Domenico Parisi. Evolving mobile robots able to display collective behaviors. *Artificial Life*, 9(3):255–268, 2003.

Gianluca Baldassarre, Vito Trianni, Michael Bonani, Francesco Mondada, Marco Dorigo, and Stefano Nolfi. Self-organized coordinated motion in groups of physically connected robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 37(1):224–239, 2007.

André Bastos. Experiments in evolutionary computacional robotics. Master's thesis, Faculty of Sciences, University of Lisbon, Portugal, 2011. URL `http://docs.di.fc.ul.pt/jspui/handle/10455/6758`.

Levent Bayindir and Erol Şahin. A review of studies in swarm robotics. *Turkish Journal of Electrical Engineering and Computer Sciences*, 15(2):115–147, 2007.

Randall D. Beer and John C. Gallagher. Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1(1):91–122, 1992.

Y. Uny Cao, Alex S. Fukunaga, and Andrew B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–27, 1997.

Mauro Castelli, Luca Manzoni, and Leonardo Vanneschi. A method to reuse old populations in genetic algorithms. In *Progress in Artificial Intelligence, 15th Portuguese Conference on Artificial Intelligence*, volume 7026 of *Lecture Notes in Computer Science*, pages 138–152. Springer, Berlin, Germany, 2011.

Nikolaus Correll and Alcherio Martinoli. Modeling self-organized aggregation in a swarm of miniature robots. In *IEEE International Conference on Robotics and Automation*, pages 379–384. IEEE, New York, NY, 2007.

Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*, pages 10–20. Springer, Berlin, Germany, 2005.

Giuseppe Cuccu and Faustino J. Gomez. When novelty is not enough. In *European Conf. on the Applications of Evolutionary Computation*, volume 6624 of *LNCS*, pages 234–243. Springer, Berlin, Germany, 2011.

Giuseppe Cuccu, Faustino J. Gomez, and Tobias Glasmachers. Novelty-based restarts for evolution strategies. In *IEEE Congress on Evolutionary Computation*, pages 158–163. IEEE, New York, NY, 2011.

Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Hoboken, NJ, 2001.

Stéphane Doncieux and Jean-Baptiste Mouret. Behavioral diversity measures for evolutionary robotics. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, New York, NY, 2010.

Marco Dorigo and Erol Şahin. Guest editorial: Swarm robotics. *Autonomous Robots*, 17(2-3):111–113, 2004.

Dario Floreano and Francesco Mondada. Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot. *From animals to animats*, 3:421–430, 1994.

Dario Floreano and Francesco Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(3):396–407, 1996.

David E. Goldberg. Simple genetic algorithms and the minimal, deceptive problem. In *Genetic Algorithms and Simulated Annealing*, Research Notes in Artificial Intelligence, pages 74–88. Pitman Publishing, London, UK, 1987.

David E. Goldberg and Jon Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Lawrence Erlbaum, Mahwah, NJ, 1987.

Faustino Gomez and Risto Miikkulainen. Incremental evolution of complex general behavior. Technical Report AI96-248, Dep. of Computer Sciences, The University of Texas, Austin, TX, 1996. URL `ftp://ftp.cs.utexas.edu/pub/AI-Lab/tech-reports/UT-AI-TR-96-248.ps.gz`.

Faustino J. Gomez. Sustaining diversity using behavioral information distance. In *Genetic and Evolutionary Computation Conference*, pages 113–120. ACM, New York, NY, 2009.

Inman Harvey, Philip Husbands, and Dave Cliff. Issues in evolutionary robotics. In *Second International Conference on Simulation of Adaptive Behavior*, pages 364–373. MIT Press, Cambridge, MA, 1993.

Simon Haykin. *Neural networks: A comprehensive foundation*. MacMillan, New York, NY, 1994.

John H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI, 1975.

Gregory Hornby. ALPS: the age-layered population structure for reducing the problem of premature convergence. In *Genetic and Evolutionary Computation Conference*, pages 815–822. ACM, New York, NY, 2006.

Jianjun Hu, Erik D. Goodman, Kisung Seo, Zhun Fan, and Rondal Rosenberg. The hierarchical fair competition (hfc) framework for sustainable evolutionary algorithms. *Evolutionary Computation*, 13(2):241–277, 2005.

Louis Hugues and Nicolas Bredeche. Simbad: An autonomous robot simulation package for education and research. In *From Animals to Animats 9*, volume 4095 of *Lecture Notes in Computer Science*, pages 831–842. Springer, Berlin, Germany, 2006.

Marcus Hutter and Shane Legg. Fitness uniform optimization. *IEEE Transactions on Evolutionary Computation*, 10(5):568 –589, 2006.

Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Advances in Artificial Life, Third European Conference on Artificial Life*, volume 929 of *Lecture Notes in Computer Science*, pages 704–720. Springer, Berlin, Germany, 1995.

Raphael Jeanson, Colette Rivault, Jean-Louis Deneubourg, Stephane Blanco, Richard Fournier, Christian Jost, and Guy Theraulaz. Self-organized aggregation in cockroaches. *Animal Behaviour*, 69(1):169–180, 2005.

Chris V. Jones and Maja J. Mataric. Behavior-based coordination in multi-robot systems. In *Autonomous Mobile Robots: Sensing, Control, Decision Making, and Applications*, page 549–569. Marcel Dekker, New York, NY, 2005.

Joshua Knowles, Richard Watson, and David Corne. Reducing local optima in single-objective problems by multi-objectivization. In *Evolutionary Multi-Criterion Optimization*, volume 1993 of *Lecture Notes in Computer Science*, pages 269–283. Springer, Berlin, Germany, 2001.

Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

Peter Krcah. Solving deceptive tasks in robot body-brain co-evolution by searching for behavioral novelty. In *10th International Conference on Intelligent Systems Design and Applications*, pages 284–289. IEEE, New York, NY, 2010.

Joel Lehman and Kenneth O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *Proceedings of the Eleventh International Conference on Artificial Life (ALIFE XI)*, Cambridge, MA, 2008. MIT Press, Cambridge, MA.

Joel Lehman and Kenneth O. Stanley. Revising the evolutionary computation abstraction: minimal criteria novelty search. In *Genetic and Evolutionary Computation Conf.*, pages 103–110. ACM, New York, NY, 2010a.

Joel Lehman and Kenneth O. Stanley. Efficiently evolving programs through the search for novelty. In *Genetic and Evolutionary Computation Conference*, pages 837–844. ACM, New York, NY, 2010b.

Joel Lehman and Kenneth O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223, 2011a.

Joel Lehman and Kenneth O. Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Genetic and Evolutionary Computation Conf.*, pages 211–218. ACM, New York, NY, 2011b.

Wenguo Liu, Alan F. T. Winfield, and Jin Sa. Modelling swarm robotic systems: A case study in collective foraging. In *Towards Autonomous Robotic Systems (TAROS 07)*, pages 25–32, 2007.

François Michaud and Etienne Robichaud. Sharing charging stations for long-term activity of autonomous robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2746–2751, 2002.

Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stéphane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alcherio Martinoli. The e-puck, a robot designed for education in engineering. In *9th Conf. on Autonomous Robot Systems and Competitions*, pages 59–65. IPCB, Castelo Branco, Portugal, 2009.

Jean-Baptiste Mouret. Novelty-based multiobjectivization. *New Horizons in Evolutionary Robotics*, 341:139–154, 2011.

Jean-Baptiste Mouret and Stéphane Doncieux. Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. In *IEEE Congress on Evolutionary Computation*, pages 1161–1168. IEEE, New York, NY, 2009.

Jean-Baptiste Mouret and Stéphane Doncieux. Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evolutionary Computation*, 20(1): 91–133, 2012.

Jean-Baptiste Mouret and Stéphane Doncieux. Incremental evolution of animats' behaviors as a multi-objective optimization. In *From Animals to Animats 10*, volume 5040 of *Lecture Notes in Computer Science*, pages 210–219. Springer, Berlin, Germany, 2008.

Angélica Muñoz Meléndez, François Sempé, and Alexis Drogoul. Sharing a charging station without explicit communication in collective robotics. In *Proceedings of the 7th International Conference on Simulation of Adaptive Behavior on From Animals to Animals*, ICSAB, pages 383–384. MIT Press, Cambridge, MA, 2002.

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

Stefano Nolfi and Dario Floreano. *Evolutionary Robotics: The Biology,Intelligence,and Technology*. MIT Press, Cambridge, MA, 2000.

Sebastian Risi, Sandy D. Vanderbleek, Charles E. Hughes, and Kenneth O. Stanley. How novelty search escapes the deceptive trap of learning to learn. In *Genetic and Evolutionary Computation Conf.*, pages 153–160. ACM, New York, NY, 2009.

Andrea Soltoggio and Ben Jones. Novelty of behaviour as a basis for the neuro-evolution of operant reward learning. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 169–176. ACM, New York, NY, 2009.

Onur Soysal, Erkin Bahgeçi, and Erol Şahin. Aggregation in swarm robotic systems: Evolution and probabilistic control. *Turkish Journal of Electrical Engineering and Computer Sciences*, 15(2):199–225, 2007.

Corné Sprong. Common tasks in evolutionary robotics, an overview. Technical report, Faculty of Sciences, University of Amsterdam, Netherlands, 2011. URL `http://www.few.vu.nl/nl/Images/werkstuk-sprong_tcm38-217791.pdf`.

Kenneth O. Stanley. *Efficient Evolution of Neural Networks Through Complexification*. PhD thesis, Dep. of Computer Sciences, The University of Texas, Austin, TX, 2004. URL `http://nn.cs.utexas.edu/downloads/papers/stanley.phd04.pdf`.

Kenneth O. Stanley and Risto Miikkulainen. Evolving neural network through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

Vito Trianni. *On the Evolution of Self-Organising Behaviours in a Swarm of Autonomous Robots*. PhD thesis, Université Libre de Bruxelles, Brussels, Belgium, 2006. URL `http://laral.istc.cnr.it/trianni/docs/thesis-trianni.pdf`.

Vito Trianni, Roderich Groß, Thomas Halva Labella, Erol Şahin, and Marco Dorigo. Evolving aggregation behaviors in a swarm of robots. In *European Conf. on Artificial Intelligence*, volume 2801 of *LNCS*, pages 865–874. Springer, Berlin, Germany, 2003.

Vito Trianni, Stefano Nolfi, and Marco Dorigo. Cooperative hole avoidance in a swarm-bot. *Robotics and Autonomous Systems*, 54(2):97 – 103, 2006.

L. Darrell Whitley. Fundamental principles of deception in genetic search. In *Foundations of Genetic Algorithms*, pages 221–241. Morgan Kaufmann, San Mateo, CA, 1991.