UNIVERSIDADE D
COIMBRA

João Pedro Gonçalves Teixeira de Macedo

NATURE INSPIRED ALGORITHMS FOR ROBOTIC ODOUR SEARCH

Tese no âmbito do Doutoramento em Engenharia Informática, Sistemas Inteligentes, orientada pelo Professor Doutor Lino José Forte Marques e pelo Professor Doutor Ernesto Jorge Fernandes Costa, e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Maio de 2023

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Nature Inspired Algorithms for Robotic Odour Search

João Pedro Gonçalves Teixeira de Macedo

PhD in Informatics Engineering, Intelligent Systems
PhD Thesis submitted to the University of Coimbra

Advised by Professor Lino José Forte Marques
and
Professor Ernesto Jorge Fernandes Costa

May, 2023

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Algoritmos Inspirados na Natureza para Procura de Fontes de Odores com Recurso a Robôs

João Pedro Gonçalves Teixeira de Macedo

Doutoramento em Engenharia Informática, Sistemas Inteligentes
Tese de Doutoramento apresentada à Universidade de Coimbra

Orientado pelo Professor Lino José Forte Marques
e pelo
Professor Ernesto Jorge Fernandes Costa

Maio, 2023

This work was partially supported by the Portuguese Foundation for Science and Technology (FCT) under Ph.D. grants SFRH/BD/129673/2017 and COVID/BD/152379/2022, co-funded by the European Social Fund, through the Centre's Regional Operational Program (CENTRO 2020). It was also partially supported by the European Regional Development Fund (ERDF), through CENTRO 2020, in the scope of the project UltraBot - Robô para desinfecção por radiação ultravioleta, under Grant CENTRO-01-0247-FEDER-072644.

Life finds a way

- Dr. Malcom, Jurassic Park

Acknowledgements

While writing this thesis I cannot help to remember all the people who, in some way or another, helped me get here. Doing a Ph.D. is not easy and could have never been possible without the help of all of you. So I'd like to take this opportunity to thank you all.

First and foremost, I'd like to express my deepest gratitude to my advisors, Professors Lino Marques and Ernesto Costa, whose guidance and support were crucial throughout this quest. Thank you for accepting to oversee my work, for allowing me to explore different ideas but also for giving me a much needed push every time that I doubted myself.

Secondly I'd like to express my gratitude to my colleagues and friends, thankfully too many to name them all. I'd like to acknowledge my labmates, Rui Baptista and Hugo Magalhães, for all the assistance with the robots. The experiments would have been much more difficult without your help. A special thanks to Nuno Lourenço and João Campos, for all the conversations and for lending me computers to run the experiments when I was out of time. And I'd also like to thank Pedro Silva, Daniel Lopes, Jéssica Parente, Luís Gonçalo, Adriana Leal and Mauro Pinto, for all the lunch breaks that were much needed to regain motivation.

I am also thankful for my family, in particular my parents and grandparents, who were constantly concerned about my struggles and needs. Thank you, and rest assured that I have finally finished the course.

Last but not least, I'm extremely grateful to Luísa, for all your love, patience and understanding throughout these years. I could have never undertaken this journey without your support and encouragement. I hope we can finally find the time to do all the things that were put on hold this past year.

Abstract

In nature, animals use the sense of smell to successfully locate sources of food, danger and mates. This sense may also be used in many other applications, such as locating victims in disaster scenarios or buried landmines, detecting concealed drugs or tracking the sources of pollutants and diseases. While currently many of these tasks are tackled with the help of trained dogs, the dangerous work conditions that are typically involved have been motivating the research community to propose methods to replace the animals with autonomous robots that can be monitored from safe distances. Yet, most of the existing approaches are manually designed and, due to the difficulty of the task at hand, tend to be sub-optimal. A promising approach would be not to hand-design, but rather to automatically produce those controllers though artificial intelligence methods.

This thesis devises nature-inspired artificial intelligence methods (i.e., Evolutionary Algorithms) to evolve robots which, much like animals, are able to locate odour sources both individually and as a group. Similarly to most Evolutionary Robotics (ER) works, this thesis evolves the robotic controllers in simulation, as it is not only faster, cheaper and safer, but also enables testing scenarios that would be difficult to test in the real world. As such, one our first contributions is the development of a fast and realistic simulator for enabling the evolution of search strategies for odour source localisation. Moreover, due to the lack of ER works for odour source localisation, a study is conducted to design an adequate evaluation function for this task, i.e., one that provides a reliable fitness value with a minimum number of evaluations and a minimum amount of prior knowledge to prevent biasing the evolution.

Four approaches were devised to evolve the robotic controllers. The first approach relies on Genetic Programming (GP) to produce human-readable search strategies from bio-inspired perceptions and actions. The ability to visually inspect the controllers is of great importance, as it not only enables verifying their soundness and adjusting their parameters to better suit particular scenarios, but also enables experimenters to draw insights from the search strategies, which may be used to further advance the knowledge of the field. However, GP often suffers from bloat, phenomenon that translates into the uncontrolled growth of the individuals without a corresponding increase in performance. Bloat will inevitably decrease the readability of the evolved controllers and increase their computational requirements. In this thesis, we propose Geometric Syntactic Genetic Programming (GSynGP), a new GP variant that produces compact syntax trees with equivalent performance to those evolved by the standard variant.

The second approach is based on Infotaxis, a popular method to estimate the location of the source by fitting a gas dispersion model to environmental measurements. We propose Evolutionary Infotaxis, an evolutionary approach that addresses one of the main drawbacks of Infotaxis by using a Genetic Algorithm

to automatically parametrise its gas dispersion model, optimising its performance in a given environment. The results show that, contrarily to common belief, the parameters that better approximate the environment are not necessarily the ones that lead to the best performance.

The third approach relies on GSynGP to evolve tree-based controllers that combine infotactic behaviours with bio-inspired building blocks. The experimental results show that this approach outperforms both its counterparts and, the analysis of the evolved solutions shows that Infotaxis is better suited for finding and reacquiring the odour plume, while bio-inspired behaviours are more efficient for tracking it to its source.

Finally, a multi-robot evolutionary approach is devised and the influence of the number of robots and of cooperation is investigated. Furthermore, the multi-robot approach is compared to the single-robot approaches, showing that using multiple robots provides better results than even the most complex single-robot approach.

Preliminary tests of the best evolved controllers are conducted in a wind-tunnel, qualitatively validating the simulation-based results. One of the advantages of their white-box nature is also demonstrated by manually adjusting their parameters to better cope with the differences between the simulated environment and the wind tunnel.

Keywords: Odour Source Localisation, Evolutionary Robotics, Evolutionary Algorithms, Genetic Programming

Resumo

Na natureza, os animais usam o olfato para detectar e localizar fontes de alimento, perigo e outros membros da sua espécie. O sentido olfativo pode ser usado em muitas outras aplicações, tais como localizar pessoas desaparecidas, detectar narcóticos ou encontrar fontes de poluentes e doenças. Atualmente muitas destas tarefas são realizadas com o auxílio de cães, mas os riscos inerentes têm motivado a comunidade científica a propor métodos baseados em robôs autónomos que possam ser monitorizados a partir de distâncias seguras. No entanto, a maioria das abordagens existentes são desenhadas manualmente e, devido à dificuldade desta tarefa, tendem a ser sub-ótimas. Um caminho promissor seria recorrer a métodos de inteligência artificial para produzir estratégias de procura.

Esta tese propõe métodos de inteligência artificial inspirados na natureza (i.e., Algoritmos Evolucionários (AE)) para evoluir robôs que, tal como os animais, são capazes de localizar fontes de odor, tanto individualmente, como trabalhando em equipa. À semelhança da maioria dos trabalhos de Robótica Evolucionária (RE), os controladores robóticos são evoluídos em simulação, dado ser uma forma mais rápida, segura e com menores custos, permitindo ainda testar cenários que seriam difíceis de testar no mundo real. Como tal, uma das nossas contribuições é o desenvolvimento de um simulador rápido e realista para permitir a evolução de estratégias de procura de fontes de odor. Além disso, devido à escassez de trabalhos de RE na procura de fontes de odor, realizou-se um estudo para desenhar uma função de avaliação adequada para esta tarefa, i.e., uma função que produz um valor de aptidão confiável com o mínimo de avaliações e contendo uma quantidade mínima de conhecimento prévio para evitar enviesar a evolução.

Esta tese propões quatro abordagens para evoluir controladores robóticos. A primeira recorre a Programação Genética (PG) para produzir estratégias de procura de fontes de odor interpretáveis por humanos, a partir the percepções e acções bioinspiradas. A capacidade de inspecionar visualmente os controladores é de extrema relevância, uma vez que permite verificar sua solidez e refinar os seus parâmetros para cenários específicos. Para além disso, através da análise dos controladores é possível extrair conhecimento para melhorar as estratégias de procura desenhadas manualmente, avançando o conhecimento da área. No entanto, a PG sofre frequentemente de bloat, um fenómeno que se traduz no crescimento descontrolado dos indivíduos sem um correspondente aumento de desempenho, que diminui a sua legibilidade e aumenta os seus requisitos computacionais. Nesta tese propomos um novo algoritmo de PG (GSynGP) que efectua operações de recombinação geométricas no espaço sintático, produzindo indivíduos compactos com desempenho equivalente aos evoluídos pela PG tradicional.

A segunda abordagem baseia-se na Infotaxis, uma abordagem com grande pop-

ularidade para estimar a localização de fontes de odor com recurso a modelos de dispersão de gás. A abordagem que propomos, Evolutionary Infotaxis, utiliza um Algoritmo Genético para parametrizar o modelo de dispersão de gás da Infotaxis, optimizando a sua performance num ambiente específico e solucionando uma das suas principais desvantagens (i.e., a necessidade de parametrizar o modelo de dispersão de gás manualmente). Os resultados experimentais mostram que, contrariamente ao expectável, os parâmetros que melhor se aproximam do ambiente não são necessariamente os que levam ao melhor desempenho.

A terceira abordagem proposta recorre ao GSynGP para evoluir controladores *white-box* que combinam comportamentos infotáticos com percepções e ações bioinspiradas. Os resultados experimentais mostram que esta abordagem supera ambos os métodos que lhe deram origem e, através da análise dos controladores, é possível verificar que a Infotaxis é mais adequada para (re)encontrar a pluma de odor, enquanto os comportamentos bioinspirados são mais eficientes para segui-la até à sua fonte.

Propomos ainda uma abordagem evolucionária multi-robô, estudando a influência do número de robôs e da cooperação no desempenho da procura. Além disso, a abordagem multi-robô é comparada com as abordagens que usam um único robô, mostrando que o uso de vários robôs origina melhores resultados do que todas as abordagens com um único robô.

Finalmente, realizamos testes preliminares dos controladores num túnel de vento, validando qualitativamente os resultados obtidos em simulação. Uma das vantagens da natureza *white-box* dos controladores é também demonstrada, permitindo o ajuste manual dos seus parâmetros para lidar melhor com as diferenças entre o ambiente simulado e o túnel de vento.

Palavras-chave: Procura de Fontes de Odor, Robótica Evolucionária, Algoritmos Evolucionários, Programação Genética

List of Publications

Awards

- *EvoStar Outstanding Student*: The *EvoStar Outstanding Student* award is given to students who are the first authors of highly marked papers in peer review. In 2020 I received this award with the paper entitled *Locating Odour Sources with Geometric Syntactic Genetic Programming.*

Book chapters

1. Marques, L., Magalhães, H., Baptista, R., and Macedo, J. (2022, December). Mobile robot olfaction state-of-the-art and research challenges. In IET book "Sensory Systems For Robotic Applications", Ravinder Dahiya, Oliver Ozioko, Gordon Cheng, 2022, 97, 213. IET.

Journal publications

1. Macedo, J., Marques, L., and Costa, E. (2019, May). A comparative study of bio-inspired odour source localisation strategies from the state-action perspective. Sensors, 19(10), 2231.

Conference publications

1. Macedo, J., Marques, L., and Costa, E. (2022, July). Hybridizing bio-inspired strategies with infotaxis through genetic programming. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) (pp. 95-103).
2. Macedo, J., Marques, L., and Costa, E. (2021, September). Evolving Infotaxis for Meandering Environments. In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 8431-8436). IEEE.
3. Macedo, J., Marques, L., and Costa, E. (2021, July). Designing fitness functions for odour source localisation. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO) (pp. 103-104).
4. Macedo, J., Marques, L., and Costa, E. (2020, April). Locating Odour Sources with Geometric Syntactic Genetic Programming. In 2020 International Conference on the Applications of Evolutionary Computation (EvoApplications) (pp. 212-227). Springer International Publishing.

5. Macedo, J., Marques, L., and Costa, E. (2019, April). A performance comparison of bio-inspired behaviours for odour source localisation. In 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC) (pp. 1-6). IEEE.
6. Macedo, J., Fonseca, C. M., and Costa, E. (2018, April). Geometric crossover in syntactic space. In 2018 European Conference on Genetic Programming (EuroGP) (pp. 237-252). Springer International Publishing.
7. Macedo, J., Marques, L., and Costa, E. (2017, April). Robotic odour search: Evolving a robot's brain with Genetic Programming. In 2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC) (pp. 91-97). IEEE.

Other contributions

1. Macedo, J., Marques, L., and Costa, E. (2022, November). Evolving Swarm Formations for Odour Source Localisation. In Fifth Iberian Robotics Conference: Advances in Robotics (ROBOT), Volume 2 (pp. 142-153). Springer International Publishing.
2. Magalhães, H., Baptista, R., Macedo, J., and Marques, L. (2020, December). Towards Fast Plume Source Estimation with a Mobile Robot. Sensors, 20(24), 7025.
3. Baptista, R., Magalhães, H., Macedo, J., and Marques, L. (2020, November). 2D thermal wind sensor for mobile robot anemotaxis: Design and validation. In 2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR) (pp. 227-232). IEEE.
4. da Conceição, C. S., Macedo, J., and Marques, L. (2019, November). Detecting indoor smoldering fires with a mobile robot. In Fourth Iberian Robotics Conference: Advances in Robotics (ROBOT), Volume 1 (pp. 606-616). Springer International Publishing.
5. Macedo, J., Marques, L., and Costa, E. (2016, May). Evolving neural networks for multi-robot odor search. In 2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC) (pp. 288-293). IEEE.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

Contents

1.1 Motivation

In November of 2014, an outbreak of legionellosis in Portugal caused 12 deaths and 375 injuries. The outbreak affected various cities of the Lisbon district, before its source being identified as the cooling towers of a fertilizer plant in Forte da Casa, Vila Franca de Xira. Legionella bacteria grow in water and can spread in droplets carried by the wind, similarly to other substances such as airborne pollutants.

Olfaction enables the detection and localisation of distant targets, even if they are silent and invisible, similarlly to the legionella bacteria. In nature, organisms use this sense to locate sources of food, danger and mates. In turn, humans may use this sense in many real world applications, such as locating victims in disaster scenarios or buried landmines, detecting concealed drugs or tracking the sources of pollutants and diseases. Currently, due to the limitations of the human nose, trained dogs are used to assist humans in those tasks. However, this is not a perfect solution, as the dangerous work conditions endanger the well-being of both animals and their handlers. Moreover, the search operations may take a long time, exhausting the searchers and making them more prone to make mistakes.

In order to reduce the risk to both humans and animals, the robotics community has been actively working on methods to replace animals with robots [Jing et al., 2021] that can be monitored from a safe distance. Unfortunately, the existing approaches only work in controlled environments and do not scale well for real scenarios. This is mainly due to the difficulty of the task at hand. Odour sources release molecules at continuous or intermittent rates. The odour particles, once released, flow with the wind, spreading through molecular diffusion and turbulent dispersion. Turbulence creates an intermittent distribution of concentration with many local peaks, hampering direct attempts to estimate gradients. Moreover, whilst the transport of odour molecules in a fluid can be modelled by advection-diffusion equations, the constraints imposed by the interactions between obstacles and the fluid's motion, as well as the resulting turbulent effects, make the inversion of this phenomenon computationally intractable. From a computational perspective, the task at hand can also be seen as a problem of exploring an unknown environment looking for (odour) cues of a deceptive trail and following it to its source (i.e., the chemical source), using robots with imperfect sensors and actuators.

1.2 Working hypothesis

While one approach for locating odour sources could be based on experts teleoperating mobile robots, these tasks' real world scenarios often contain obstacles and structures that affect wireless communications. Moreover, in many applications it would be interesting to deploy several robots working cooperatively, which in a teleoperation setting would imply having one operator per robot, consequently increasing the operation cost. As a result, the robot must have a high degree of autonomy, being able to complete their task with no human

intervention. In order to do it, the robots must have a controller that takes the place of the operator and uses its perceptions to select proper actions. Even though the controller could be hand-designed, it would be difficult for the experimenters to design the optimal controller without having complete knowledge of the world, specially when considering applications involving dynamic environmental conditions (e.g., weather) and the strong possibility of loss of agents. The use of multiple robots working cooperatively provides robustness to the loss of agents and enables simultaneous distributed sensing of the environment, both of which are advantageous when attempting to locate chemical sources. However, these benefits come at the cost of an increased system complexity, often requiring inter-robot communication and coordination mechanisms, and making the task of hand-designing controllers that take advantage of other agents' perceptions to produce the desired results much more difficult. A promising approach would be not to hand-design, but rather to automatically produce those controllers though artificial intelligence methods.

In nature, animals successfully locate odour sources to accomplish various tasks and thus their behaviours have been the source of inspiration for various robotic search strategies. This thesis proposes to also draw inspiration from nature to devise artificial intelligence methods to automatically design robotic search strategies for locating odour sources. Evolutionary Algorithms (EAs) are stochastic search heuristics loosely inspired by the principles of evolution by Natural Selection and Mendel's Genetics. They have been successfully applied to solve problems with no analytical solution, or when finding such solution would be computationally intractable. Odour source localisation is one of such problems, as is the evolution of robotic controllers which, along with the evolution of robot bodies, yielded the research field of Evolutionary Robotics (ER). As a result, our working hypothesis is that, much like animals, EAs can be applied to evolve robotic controllers that are able to locate odour sources, either individually or as a group.

Genetic Programming is a family of EAs that evolve computer programs in the form of syntax trees [Koza, 1992]. In this thesis we focus mainly on Genetic Programming to evolve white-box robotic controllers that can be visually inspected. The ability to visually inspect the evolved controllers is of great importance, as it not only enables the verification of their soundness, as well as enables to draw insights regarding the search strategies, which may be used to further advance the knowledge of the experimenters. This choice of resorting to GP is supported by a previous study [Svec and Gupta, 2012] which used tree-based Genetic Programming in place of neural networks, exactly for its ability to produce human-readable robotic controllers. Moreover, other works have stated that EAs are promising approaches for designing robotic controllers [Scheper et al., 2016], particularly in multi-robot scenarios [Waibel et al., 2009]. Throughout this document, we will often refer to the evolution of robotic controllers or search strategies. In any case, we are referring to high-level decision making processes that use the states of the robot, containing its perceptions and, possibly, those of its team-mates, along with its goals and memories to select the appropriate actions.

1.3 Goals

The main goal of this thesis is to study and develop nature-inspired algorithms that enable one robot or a group of robots to locate odour sources in realistic environments. In order to do it, the robots must fulfil the three phases of the odour source localisation (OSL) process i.e., they must: (1) explore the environment, searching for odour detections; (2) track the chemical plume to the vicinity of its source; and (3) pinpoint the location of the chemical source. In the literature, the third phase of the OSL process is typically assumed to be carried out with additional sensors (such as vision) that are able to detect the source when in close proximity. In this thesis, we also make this assumption and focus on solving the first two phases of the OSL process. Moreover, we shall focus on evolutionary robotics methods to enable each robot to evolve controllers for performing the task both individually and as a member of a group, leading to a system that may use multiple agents cooperatively to improve its performance while remaining robust to the loss of agents.

Research questions

We formulate the main research questions as follows: *How can robots evolve to locate odour sources?*

From these research questions other sub-questions emerge, namely:

1. Which type of evolutionary algorithm enables the evolution human-readable robotic controllers?
2. How can one robot evolve to locate a chemical source?
3. How can a group of robots evolve to locate chemical sources cooperatively?
4. Is it better to use many simple robots or a single, but complex, one?

1.4 Contributions

Throughout the works of this thesis, several original contributions were made, namely:

1. Development of a fast and realistic simulator for enabling the evolution of search strategies for the odour source localisation task [Macedo et al., 2019];
2. Design and study of evaluation functions for odour source localisation strategies [Macedo et al., 2021a];
3. Development of Geometric Syntactic Genetic Programming (GSynGP), a new GP variant that produces compact individuals with equivalent performance to those evolved by the standard variant and its aplication to evolve robotic controllers for odour source localisation [Macedo et al., 2018, Macedo et al., 2020];

4. Development of an Evolutionary Infotaxis approach, which automatically parametrises its inner gas dispersion model to optimise performance in a given environment [Macedo et al., 2021b];
5. Development of GPInfotaxis, a GSynGP-based algorithm for creating human-readable hybrid search strategies combining bio-inspired behaviours and perceptions with Infotaxis [Macedo et al., 2022];
6. Development of an evolutionary multi-robot approach for odour source localisation, studying the influence of the number of robots and of the cooperation in the performance.

1.5 Organisation

This document is organised as follows: Chapter 2 introduces the background concepts and the state of the art literature on learning and evolutionary robotics as well as robotic odour source localisation; Chapter 3 describes the developed Geometric Syntactic Genetic Programming algorithm; Chapter 4 details the experiments made with a single robot; Chapter 6 presents the experiments made with groups of robots; and finally, Chapter 7 draws the conclusions from this work and provides insight into future endeavours.

Chapter 2

Background and state of the art

Contents

Figure 2.1: Smoke plume.

This chapter starts by presenting the background concepts on odour source localisation, robotics, evolutionary computation and evaluation functions necessary to understand this thesis. It then carries on to report the state of the art on evolutionary robotics and odour source localisation.

2.1 Background concepts

2.1.1 Odour source localisation

The term *odour* is typically used to refer to scents, including unpleasant ones. It is deeply related to olfaction, the sense through which animals are able to detect volatile chemical compounds, which in turn can be emitted by various sources. In the literature, *odour source localisation* (OSL) refers to the task of finding the source that is emitting an airborne or waterborne chemical substance. In works dealing with airborne chemical compounds, the term *gas source localisation* (GSL) is also used to refer to the same task. Throughout this document, we shall use both terms interchangeably.

The difficulty in locating chemical sources arises from the process of how odour spreads [Stockie, 2011]. As odour molecules are released from the source, they spread mainly through turbulent advection and molecular diffusion, creating intermittent chemical plumes with various local voids and peaks of concentration (see Figure 2.1).

The task of locating chemical sources has received a lot of attention from the research community and it is currently accepted that it comprises three well defined stages:

1. **Plume finding**: where the agent has yet to sense odour and must explore the environment searching for chemical cues;
2. **Plume tracking**: where the searcher is in contact with the plume and must follow it to the vicinity of its source;
3. **Source declaration**: where the searcher has reached the vicinity of the chemical source and must pinpoint its location. This task is often carried out with the help of other sensor modalities, such as vision.

Moreover, various methods have been proposed to model both the time-averaged and instantaneous chemical dispersion, as well as the carrying flow. These models are not only necessary to estimate the dispersion of pollutants in an environment, but also to create simulators that enable experimenters to test new approaches for tackling the odour source localisation task. The following subsection describes the models used in the simulator developed in this thesis.

2.1.1.1 Airflow model

A popular model of airflow has been used by Farrell et al. [Farrell et al., 2002] to efficiently emulate the short time-scale structure of chemical plumes. The airflow is computed through partial differential equations in a 2D grid of square cells that extends over the entire environment, but the models could also be extended to 3D. While the resolution of this grid is adjustable, care should be taken not to make it too fine, as the model is not meant to emulate the small-scale turbulent phenomena of the wind, but rather its large-scale advection dynamics that have a greater impact on gas dispersion. The initial wind velocity is constant and predefined by the user. This velocity is computed in vectorial form, with u and v respectively denoting the components aligned with the x-axis and the y-axis. Over the course of the simulation, the velocity vector of the wind at each vertex of the grid varies according to the following equations:

$$\frac{\partial \bar{u}}{\partial t} = -\bar{u}\frac{\partial \bar{u}}{\partial x} - \bar{v}\frac{\partial \bar{u}}{\partial y} + \frac{K_x}{2} \cdot \frac{\partial^2 \bar{u}}{\partial x^2} + \frac{K_x}{2} \cdot \frac{\partial^2 \bar{u}}{\partial y^2} \tag{2.1}$$

$$\frac{\partial \bar{v}}{\partial t} = -\bar{u}\frac{\partial \bar{v}}{\partial x} - \bar{v}\frac{\partial \bar{v}}{\partial y} + \frac{K_x}{2} \cdot \frac{\partial^2 \bar{v}}{\partial x^2} + \frac{K_x}{2} \cdot \frac{\partial^2 \bar{v}}{\partial y^2} \tag{2.2}$$

where K_x is a diffusivity coefficient. These equations may be approximated through the Finite Differences method, with each cell of the grid being updated as follows:

$$\frac{\partial \bar{u}}{\partial y}(i,j) = \frac{u(i,j+1) - u(i,j-1)}{2\Delta_g} \tag{2.3}$$

$$\frac{\partial \bar{u}}{\partial x}(i,j) = \frac{u(i+1,j) - u(i-1,j)}{2\Delta_g} \tag{2.4}$$

$$\frac{\partial^2 \bar{u}}{\partial^2 y}(i,j) = \frac{u(i,j+1) - 2u(i,j) + u(i,j-1)}{2\Delta_g} \tag{2.5}$$

$$\frac{\partial \bar{u}}{\partial x}(i,j) = \frac{u(i+1,j) - 2u(i,j) + u(i-1,j)}{2\Delta_g} \tag{2.6}$$

$$\frac{\partial \bar{u}}{\partial t}(i,j) = -\bar{u}\frac{\partial \bar{u}}{\partial x}(i,j) - \bar{v}\frac{\partial \bar{u}}{\partial y}(i,j) + \frac{K_x}{2} \cdot \frac{\partial^2 \bar{u}}{\partial x^2}(i,j) + \frac{K_x}{2} \cdot \frac{\partial^2 \bar{u}}{\partial y^2}(i,j) \tag{2.7}$$

$$\frac{\partial \bar{v}}{\partial t}(i,j) = -\bar{u}\frac{\partial \bar{v}}{\partial x}(i,j) - \bar{v}\frac{\partial \bar{v}}{\partial y}(i,j) + \frac{K_x}{2} \cdot \frac{\partial^2 \bar{v}}{\partial x^2}(i,j) + \frac{K_x}{2} \cdot \frac{\partial^2 \bar{v}}{\partial y^2}(i,j) \tag{2.8}$$

where Δ_g is the grid spacing and $u(i,j)$ and $v(i,j)$ are respectively the u and v components of the wind velocity in the grid cell (i,j).

The final step of the velocity update consists on adding zero-mean Gaussian noises to the wind velocity vectors to create meandering, being their standard deviations adjusted to create the desired wind characteristics. The wind sensed by a robot is computed as a weighted average of the wind vectors on the four grid vertexes surrounding it, with the weight of each vertex being the inverse of the robot's relative distance to it.

2.1.1.2 Odour dispersion model

Chemical sources may release odour at constant or intermittent rates. Odour filament model presented in [Farrell et al., 2002] describe chemical dispersion from pointwise sources as filaments. Each filament has a circular shape and a given amount of chemical concentration. Thus, two parameters are used for each chemical source: the average chemical emission rate $\bar{Q}$ and the filament emission rate F_r. The relation between both can be computed as follows:

$$Q = \frac{\bar{Q}}{F_r} \tag{2.9}$$

where Q is the amount of odour within each filament. The concentration provided by each filament (c_f) takes the form of a Gaussian function, as defined by Equation 2.10:

$$c_f = \frac{Q}{\sqrt{8\pi^3}R^3} \cdot e^{\left(-\frac{d^2}{2R^2}\right)} \tag{2.10}$$

where R is the radius of the filament and d is the distance between the centre of the filament and the sensor. Once released from the source, the filaments are carried by the wind. As such, on each time step, the motion of each filament is influenced by the weighted average of its surrounding wind vectors, with an added Gaussian noise to emulate the random motion relative to the plume's centreline. Moreover, the filaments are emitted with an initial radius R_0, which increases over time according to Equation (2.11) to model molecular diffusion:

$$\Delta R = \frac{\gamma}{2 \cdot R} \cdot \Delta t \tag{2.11}$$

where ΔR denotes the change in the filament's radius, γ controls the growth rate and Δt is the time step.

2.1.1.3 Gas sensor model

Each robot senses odour with a simulated gas sensor whose response is modelled as a low pass filter, according to Equation (2.12):

$$c_t = \alpha \cdot C_t + (1 - \alpha) \cdot c_{t-1} \tag{2.12}$$

where c_t is the odour concentration at the surface of the sensor at time t and α is the filter bandwidth. C_t is the instantaneous odour concentration encountered by the robot at that location and it is computed by Equation 2.13:

$$C_t = \sum_{i=1}^{N} c_{f,i} \tag{2.13}$$

where N is the total number of filaments in the environment and $c_{f,i}$ is the chemical concentration contributed by the i-th filament, as described in Equation 2.10.

The output of the sensor (y_t) is the result of bounding the chemical concentration at its surface (c_t) to its detection and saturation thresholds. The sensor outputs 0 if c_t is below its detection threshold. Otherwise, the sensor response is equal to c_t, until reaching the saturation threshold. At that point, the sensor will continue to output a signal corresponding to the saturation threshold until c_t drops below this value.

2.1.2 Robotics

Throughout history, humans have been attempting to build artificial beings. The first known attempt is that of Archytas who, at some time between 400 and 350 B.C., built a steam-powered wooden dove that was able to fly [Otfinoski, 2007]. Much later, in circa 1495, Leonardo DaVinci devised a humanoid robot based on an armoured knight [Moran, 2006]. Currently, many types of robots exist, ranging from home appliances to manipulators and various types of mobile robots. Those used for locating chemical sources fall on this later category, being mobile robots for terrestrial, aquatic and aerial environments [Marques et al., 2022]. In order to successfully locate chemical sources in any environment the robots should have three basic abilities: (1) sense the chemical concentration (and ideally also the flow velocity) at their location; (2) based on a given search strategy and on the environmental measurements, decide which action to take (i.e., where to move next); (3) move to that location and continue this cycle. In the following subsections, we present the concepts related to each of this abilities.

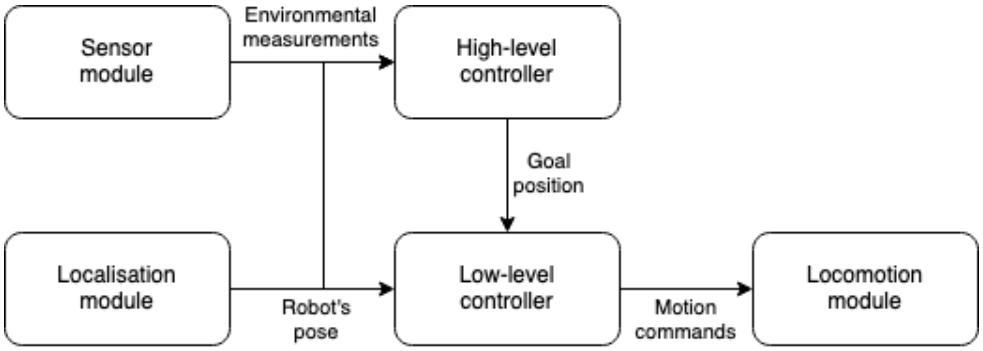


Figure 2.2: Robot architecture.

2.1.2.1 Controllers

In order to provide the previously mentioned abilities, the robots are composed by various modules (Figure 2.2), all of which are linked through two controllers: (1) a high level controller which decides where to move based on the available perceptions of the environment and on the robot's location; and (2) a low-level controller that takes as input the goal position along with the current pose of the robot and outputs motion commands. The high level controllers are the focus of this thesis and shall be described later on. The low-level controller is itself hierarchical, with a first level being responsible for combining the goal vector of the high-level controller with the obstacle avoidance behaviour. This behaviour is based on an artificial potential field, where the robot is attracted to the goal provided by the high-level controller, while simultaneously being repelled from each object detected by its sensors. At the lowest level, a Proportional-Integral-Derivative controller [Tzafestas, 2013] is responsible for assigning appropriate velocities to each wheel, assuring that the robot performs the desired motion.

2.1.2.2 Sensors

Nowadays robots are equipped with various types of sensors. In the scope of the odour source localisation task, the most important ones are those used to measure the chemical concentration and airflow in the environment, as well as those used for safely navigating, i.e., range sensors that measure the distance to nearby obstacles. There are various types of range sensors. Sonar and laser-based proximity sensors rely on the time-of-flight concept. They compute the distance to an obstacle by emitting a signal and measuring the time needed to detect its reflection. In turn, infrared sensors measure distance by emitting a pulsed infrared beam, which is reflected by nearby objects. The reflection is caught by an array of detectors and the angle at which the reflection is detected is used to compute the distance [Bräunl, 2006].

There are currently various types of sensors that may be used for sensing odours. The most commonly used ones can be arranged into the following categories [Francis et al., 2022, Marques et al., 2022]:

- Metal oxide semiconductor (MOX) sensors are the most commonly used

sensors in odour source localisation, due to their reduced cost and dimensions, as well as high sensitivity (in the parts-per-million (ppm) range) and availability for a wide variety of substances. However, these sensors suffer from low selectivity, log-term drift and are also influenced by the ambient air temperature and humidity. Moreover, these sensors have slow recovery times (1-2s), acting as low-pass filters for the chemical concentration in the environment and being influenced by previously encountered high chemical concentrations. These sensors are composed by a metal oxide element, which is heated to a temperature ranging between 150 to 500°C. In atmospheres with oxidising gases, this element decreases its electric resistance, which can be measured to provide gas concentration readings.

- Photoionization Detectors (PID) work by ionizing an air sample with ultraviolet light, breaking down the existing volatile chemical compounds (VOCs) into positive or negative ions. The resulting ions produce an electric current which is related to the chemical concentration. These sensors are fast, but are larger and more expensive than MOXs, while also having low selectivity and being influenced by air humidity.

2.1.2.3 Wheeled robots

Wheeled robots are the simplest type of mobile robots [Bräunl, 2006]. They come in various flavours, such as differential-drive, Ackermann steering and omnidirectional. Differential-drive is the most common design and also the simplest. It contains two drive wheels and possibly a set of caster wheels or pivots to maintain balance. Each wheel is controlled individually enabling the robot to rotate over its axis, move straight or perform curves. Ackermann steering is the design commonly used in passenger cars. While there may be various configurations of Ackermann steering robots, a common example is a four wheeled robot, with one axle containing drive wheels and another containing wheels that turn together to enable changes in direction. Similarly to passenger cars, these robots are unable to rotate on the spot, so manoeuvring is more complex. Omni-directional robots offer the most flexible navigation but at the cost of a more complex design. They typically have four wheels, which similarly to differential-drive robots do not turn. However, the wheels have rollers on their surface, enabling the robot to move straight, perform curves, rotate on the spot and also perform crab-like motions.

2.1.3 Evolutionary computation

Evolutionary Computation (EC) is a field of Artificial Intelligence that draws inspiration from nature to solve problems from various domains. It is a family of stochastic search heuristics loosely inspired by the principles of evolution by Natural Selection and Mendel's genetics. Those heuristics, known as Evolutionary Algorithms (EA), work by iteratively improving a population of candidate solutions, as shown in Figure 2.3. Starting with an initial population of randomly generated candidate solutions, EAs work by perturbing those solutions, creating new ones. The perturbation step starts by selecting the mates, i.e., a

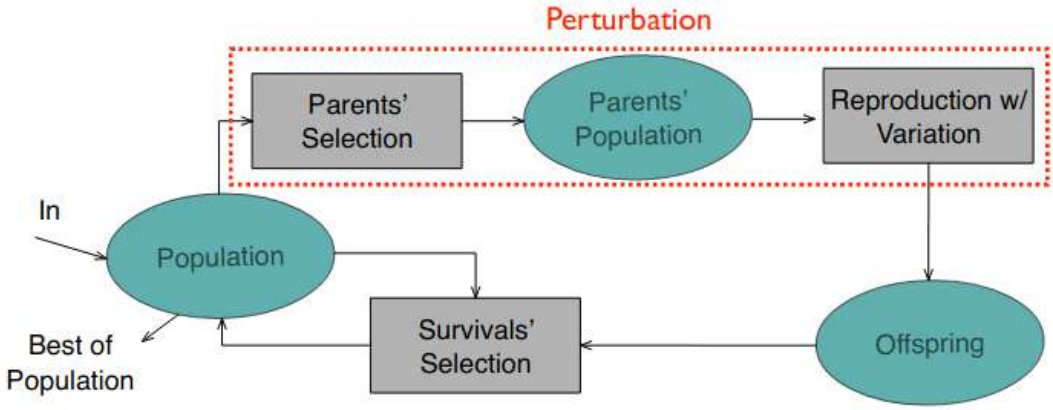


Figure 2.3: Blocks of an evolutionary algorithm

subset of the current population that shall be used to create new individuals. This selection depends on the individual's quality, but other metrics such as their novelty or size may also be used. Commonly used parent selection mechanisms are the tournament, roulette wheel and rank selection. After being selected, the mates are recombined (i.e., undergo crossover) to create offspring. In turn, the offspring may suffer mutations, introducing new genes into the population. Traditionally, the crossover and mutation operators are used to, respectively, perform global and local search. These operations are dependent on the type of individual representation and, as a result, their description shall be made on the corresponding sections. Finally, the new population is built by a survivals' selection operator that selects individuals from the pre-existing population and from its offspring. Merge, steady state, elitist and generational are examples of commonly used survival's selection operators. These steps compose a generation. The EA iterates over various generations until a termination criteria is met e.g., a time-limit or quality threshold is reached.

EAs may be categorised depending on whether they evolve the solutions for a given problem or evolve computer programs that produce those solutions. Genetic Algorithms (GA) [Eiben and Smith, 2015] are part of the first group, whereas Genetic Programming (GP) [Langdon and Poli, 2013], is part of the second group. EAs have been successfully applied over the years to solve problems from the classes of optimisation, learning and design, that have no analytic solution, or where finding that solution would be computationally intractable. Odour source localisation is one of such problems, as it is infeasible to completely and accurately model the entire environment and the interactions taking place within it, that influence the way odour propagates. Moreover, using robots, there is a constant presence of noise, both from their sensors and actuators, that the controllers have to deal with. As a result, the search space of robotic controllers is huge, making it hard to find an optimal controller. The fact that the robot moves in continuous space and the signals from its sensors are also continuous, aids in increasing the size of the search space.

Any EC experiment has a set of design issues. Some, such as the mate and sur-

vivor selection are common to both GAs and GPs. Other, such as the variation operators are dependent on the individual representation and thus are specific to each approach. We shall now describe the most commonly used operators.

2.1.3.1 Selection mechanisms

Selection mechanisms [Luke, 2013] may be split into two categories: parent selection and survivors selection. Parent selection methods choose a subset of individuals from the existing population to create new individuals. An example of such operators is tournament selection. Tournament selection starts by randomly sampling a small set of individuals from the population. Then, the best individual from that subset is selected. The size of the tournament is a way of controlling the amount of selective pressure of the EA. Typical values for this parameter are 2 and 3 individuals, which lead to little selective pressure. In fact, if 1 is used, tournament selection turns into random selection, making it harder for the EA to converge. Conversely, with the increase of the tournament size so does the probability of selecting the population's best individual, leading to a higher selective pressure. High selective pressure increases the likelihood of the population quickly converging to a limited region of the search space and rendering the EA unable to find the global optimum solution, a phenomenon known as premature convergence.

The selection of the survivals is typically the last operation of each generation. The mechanisms that do it take as input the existing population and the offspring to create a new population that will survive into the next generation. The simplest survival's selection method is the generational. It consists on discarding the current population and keeping only the offspring. Elitist selection may be seen as a variant of generational selection, where the best individuals from the current population are kept, being the same amount of worst offspring discarded. As a result, this selection operator prevents the loss of the best individuals found, creating a new population with individuals of different ages. The size of the elite is used as a mean to control selective pressure. If it is zero, elitist selection becomes generational selection, leading to the least amount of selective pressure. Conversely, if the size of the elite approximates the size of the population, only a few offspring will be kept, diminishing the search ability of the EA.

2.1.3.2 Genetic Algorithms

Genetic Algorithms [Eiben and Smith, 2015] are a family of evolutionary algorithms that iteratively improve a population of candidate solutions for a given problem. The algorithms themselves may be applied to a wide range of problems, but for each specific problem it is necessary to define the individual representation, the variation operators and the fitness function. The fitness function is the most specific design issue and essentially provides a quality value for each candidate solution. The individual representation refers to the form of the genotypes themselves and may be permutations, binary vectors, vectors of integers or real numbers. In turn, the variation operators are responsible to create new

individuals out of the existing ones. They may draw inspiration from sexual reproduction, where two or more individuals are used to create the offspring (i.e., crossover); or asexual reproduction, where one individual creates another one (i.e., mutation).

Arithmetic crossover

The arithmetic crossover is a popular recombination operator for vectors of real numbers. Taking two individuals as input, it creates one offspring O through a linear combination of the two parents P_1 and P_2, according to Equation 2.14. By adjusting the value of α, the offspring may be created at various distances from each parent.

$$O = \alpha \cdot P_1 + (1 - \alpha) \cdot P_2 \tag{2.14}$$

Gaussian mutation

The Gaussian mutation operator makes a neighbour of the original individual by probabilistically changing each of its genes g_i with a value sampled from a Gaussian distribution, as described by Equation 2.15.

$$g_i = g_i + N(0, \sigma) \tag{2.15}$$

where $N(0, \sigma)$ is a Gaussian distribution with zero mean and standard deviation σ.

Latin hypercubes

Latin hypercubes are a population initialisation method that promotes the diversity of the initial population, being particularly suited for genetic algorithms representing their individuals as vectors of real numbers. Considering a population of N individuals, this method splits the search space into N equally sized hypercubes (Figure 2.4), i.e., there are as many hypercubes as individuals to be created. The initial population is then created by randomly sampling each individual from the respective hypercube.

2.1.3.3 Genetic Programming

Genetic Programming [Langdon and Poli, 2013] is a type of machine learning that, in its most common form, evolves tree-based individuals composed by terminal (leaves) and non-terminal (inner nodes) symbols, which may be executed to provide the solutions for a given problem. Common applications are problems of symbolic regression, classification and decision making, including robotic controllers. GP differs from other EAs in the representation-specific operators, namely population initialisation, crossover and mutation. Also, the fitness evaluation is different as it requires executing the computed program encoded by the individual.

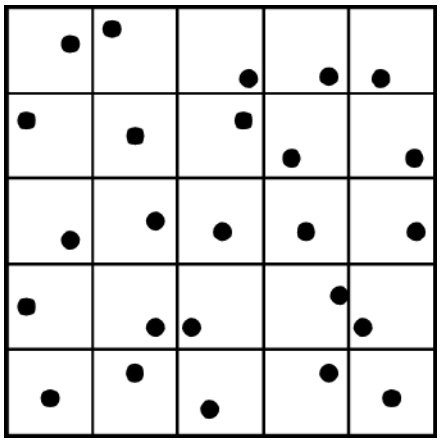


Figure 2.4: In Latin hypercubes, the 2D search space is divided into equally sized grids and one individual (dot) is sampled from each grid.

Ramped half-and-half

In GP, there are two simple methods for creating a random tree: full and grow. Both methods start from the root of the tree and recursively select a terminal (variable, constant, action, etc.) or non-terminal symbol (function, perception, sequence, etc.). The full method creates complete trees, selecting terminal symbols only when the depth limit is reached. In turn, the grow method may either choose terminal or non-terminal symbols everywhere with the exception of when the depth limit is reached. In such case, only terminal symbols may be selected. The ramped half-and-half is a combination of both approaches, creating half the population with the grow and the other half with the full method. Moreover, the individuals are typically created in equal batches of depth, from 2 to a maximum allowed depth. Thus, the initial population contains a diverse set of trees, with various sizes and shapes.

Subtree Crossover

The simplest form of subtree crossover (Figure 2.5) implements no restrictions on the size of the resulting offspring. It works by randomly selecting a subtree from each parent and replacing the subtree on a copy of the first parent by a copy of the subtree from the second parent, producing a single offspring. Variations to this method include biasing the choice towards inner nodes (increasing the impact of the crossover) or restricting the size of the chosen subtrees, so that the offspring does not violate the size constraints.

Node mutation

Mutation is an operator intended to do two things: (1) introduce new genetic material in the population; and (2) perform local search surrounding an individual. While in GAs it may be easier to quantify the impact of mutation, in GP a small change on the genotype may lead to a large modification on the phenotype, a phenomenon known as low locality. As a result, a commonly used type of mutation is the node mutation, which makes a small modification to the

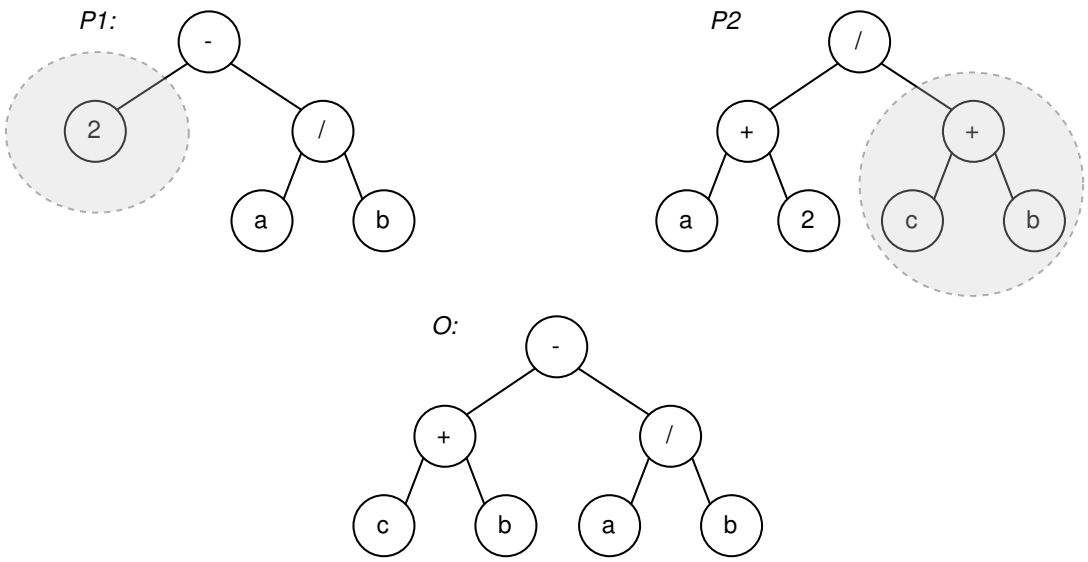


Figure 2.5: Subtree crossover.

individual by randomly selecting one of its nodes and replacing its value with a symbol of the same type (i.e., either a function or a terminal).

2.1.3.4 Geometric Operators

In the field of Evolutionary Computation, geometric variation operators are representation-independent operators based on a distance defined in the search space interpreted as a metric space [Moraglio, 2008]. A geometric crossover operator produces offspring that are on a shortest path (i.e., line segment) linking its parents. In turn, a geometric mutation operator produces an individual in the neighbourhood of the original individual, i.e., within a hypersphere centred on the original individual, whose radius defines the magnitude of the mutation. The advantage that arises from their geometry is the ability to better control the region of the search space being explored. While most crossover operators for GAs are geometric (e.g., arithmetic, uniform, N-point crossovers), those typically used in GP are not (e.g., subtree-crossover). Geometric Semantic Genetic Programming (GSGP) [Moraglio et al., 2012] is a GP variant that proposes to make geometric crossover and mutation operations. Motivated by the low-locality of GP, GSGP acts differently from the operators commonly available for GAs, modifying the parents in a way that guarantees that the distances between them and the offspring hold in the semantic space (i.e., the space of the individual's outputs given a set of inputs) rather than in syntactic space (i.e., the space of the individual's genotypes). Its crossover operator combines two parent individuals P_1 and P_2 to create a single offspring O as follows:

$$O = (P_1 \cdot F) + ((1 - F) \cdot P_2) \tag{2.16}$$

where F is a randomly generated mathematical functions that output real values contained in the [0,1] interval. In turn, its mutation operator creates a mutated

individual P' from a parent individual P as follows:

$$P' = P + ms \cdot (F - G) \tag{2.17}$$

where ms is the mutation step i.e., a coefficient controlling the size of the mutation, and F, G are two randomly generated mathematical functions.

The main drawback of GSGP arises from the definition of these operators: its individuals grow very quickly. An efficient implementation does exist, [Vanneschi, 2016] making it feasible to make longer runs of GSGP, but the resulting solutions are still too large to be human readable.

2.1.4 Evolutionary robotics

One of the main challenges in the field of robotics is the ability to adapt a robot's behaviour, to a particular task or environment. Over the years, researchers have resorted to various Artificial Intelligence and Evolutionary Computation techniques for providing adaptation [Bongard, 2013]. EAs have been used to evolve the robot's bodies [Murata and Kurokawa, 2007] and their controllers [Nolfi and Floreano, 2000], yielding the field of Evolutionary Robotics (ER). There are many applications for learning or evolution in robotics. At a lower level, the sensors (gas, wind, camera, odometry) provide information to the robot that must be perceived, and knowledge should be extracted from it. For instance, combining odometry with a laser range finder, the robot can produce a map of the environment and localise itself in it through simultaneous localisation and mapping (SLAM). Using memory, gas and wind sensors, a robot may learn to recognise whether it is within an odour plume, or just waiting for the sensor to recover from previously encountered odour. At higher levels, the robot may learn strategies that use the perceptions of the environment, together with the current and past states of the robot to decide which action is the most advantageous for achieving its goals.

There are many considerations to be made when devising an evolutionary robotics approach to odour source localisation, such as the type of controller to evolve, how the evolution is carried out and the number of robots to use. This section reports existing works, classifying them according to these aspects.

2.1.4.1 Types of controllers

The type of controller to evolve is one of the main design issues of ER. The choices range from black-box controllers, with typically less design bias, to white-box controllers that may be human-readable. Moreover, some works propose end-to-end controllers that take the sensors signals as inputs and output low-level motion commands, while others prefer to use hierarchical controllers that separate the high-level decision making from motion control, as presented in Figure 2.2.

Artificial Neural Networks

Artificial Neural Networks (ANN) are amongst the most popular types of controllers for mobile robots [Doncieux et al., 2015, Nolfi and Parisi, 1996, de Croon et al., 2013, Kuwana et al., 1996], specially in the case of robotic swarms [Nolfi and Floreano, 2000]. The popularity of ANNs to serve as robotic controllers is due to them being universal function approximators, having a high representational power which has been used in an attempt to evolve controllers with limited amount of design bias (i.e., the bias that an experimenter inadvertently introduces into the system and limits the ability of evolution to explore the search space). The minimization of the design bias is typically done by producing end-to-end controllers, that take as input the signals from a robot sensors (or low level perceptions) and output motor commands. That is the case of evostick [Francesca et al., 2014], which evolves the connection weights of hand-designed ANNs to serve as robotic controllers.

The high representational power of ANNs also creates one of its main downfalls: overfitting to the training environment. Overfitting the training environment occurs when the evolved controllers perform very well in the training settings but poorly in another settings. This may either be overfitting a particular maze, environmental (weather) conditions or, considering multiple robots, the number of other robots in the team. Overfitting is particularly likely in experiments evolving the controllers in simulation or involving dynamic and uncertain conditions, where the ANN's representational power may be used by the evolutionary process to exploit particular details of the training environment that do not generalise to others. The poor generalisation ability of overfit controllers not only reduces their ability to cope with new scenarios, but also to cross the reality gap, i.e., to operate similarly simulation and the real world (see Section 2.1.4.3). These phenomena are also applicable to swarm robotics contexts, where the robots must interact to fulfil their task. Thus, the system becomes inherently dynamic and increases the chances for ANNs to perform poorly due to lack of generalisation.

One approach to cope with the reality gap would be to hand-tune the evolved controller when transferring it to the real world. However, due to their black-box nature, ANNs cannot be easily interpreted by humans and consequently are harder to be adjusted when moving from simulation to the real world, becoming more susceptible to the reality gap. That was the case with evostick [Francesca et al., 2014], where the white-box method proposed was able to transfer better to the real world, whilst the ANN-based method suffered a considerable loss of performance. Another approach to reduce the effects of the reality gap consists on not attempting to evolve end-to-end controllers, but rather ANNs that focus only on the decision making process. Such networks act as high level controllers that arbitrate between hand-designed behaviours [Duarte et al., 2012a] or other ANNs trained for specific tasks [Duarte et al., 2012b]. Whilst in the first case the behaviours could be designed independently for simulation in the real world, the second approach requires training the low-level neural networks in both environments.

As previously said, the black-box nature of ANNs hinders the experimenters ability to understand the reasons behind their decisions and also to hand-tune them to cope with the reality gap. Moreover, their high representational power may be actually a downfall, as they become more prone to overfit. As a result, some researchers have focused on using graph-based controllers. Whilst various types of graph-based controllers do exist, they all share the disadvantage of potentially having higher design bias than ANNs, as they often include hand-designed perception and action blocks rather than using and outputting sensor and motor signals. However, they all also share the advantage of being potentially more robust to the reality gap, as the EA only combines (and sometimes parametrises) elementary behaviours and perceptions, which can be carefully programmed independently for simulation, reality and also for different robots. The existing graph-based approaches may be categorised into finite-state machines and tree-based structures, which in turn can be further divided depending on the type of trees used.

Rule-based systems

Ferrante et al. proposed to use a particular form of Genetic Programming known as Grammatical Evolution (GE) [Ryan et al., 1998] to evolve rule-based robotic controllers for a robotic swarm [Ferrante et al., 2013]. GE is an evolutionary algorithm that represents the individuals as arrays of integers and the genotype-to-phenotype mapping is based on a context-free grammar. In turn, rule-based systems are, as the name indicates, sets of rules that may be used to control agents [Hayes-Roth, 1985]. Each rule is in the if-then form, i.e., they check a condition to activate an action. Initially, they were proposed to encode hand-designed controllers and for that reason they are prime candidates to encode expert knowledge in robotic controllers. The rules in the robotic controllers evolved by Ferrante et al. use a set of preconditions (i.e., boolean perceptions extracted from the robot's observations of the environment) to arbitrate between individual low-level behaviours (i.e., motion primitives that are executed for a limited time period using feedback from the sensors), both of which are predefined by the experimenters [Ferrante et al., 2013].

Probabilistic Finite-state machines

Automode [Francesca et al., 2014] is an evolutionary approach to automatically design robotic controllers for swarms. The controllers take the form of probabilistic finite state machines (PFSM), which are created through the combination of previously hand-designed parametric modules. Automode does undertake the traditional evolutionary loop to optimize the PSFMs. In its original version (Vanilla) it uses F-race [Birattari et al., 2002], which randomly samples an initial set of PFSMs and then iteratively evaluates them to find the best performing one. However, no improvements are made to the initial individuals. A series of developments have been made since the first version of Automode. Its second version (Chocolate) [Francesca et al., 2015] uses Iterated F-race [Balaprakash et al., 2007] to create the PFSMs. Iterated F-race performs various iterations of the original F-race. The initial population of PFSMs for the first iteration is

uniformly sampled from the entire search space. In turn, the populations of the subsequent executions of F-race are sampled from distributions that prioritize the neighbourhoods of the promising solutions found so far. Later, the Maple version of Automode [Kuckling et al., 2018] focuses on studying the impact of the type of controller, by using Behaviour Trees (BT) built with the same modules and comparing their performance to PFSMs created with Automode Chocolate. Their results showed that the performance of the swarms controlled by PFSMs and BTs was equivalent, supporting the use of tree-based controllers that can be evolved with Genetic Programming.

Tree-based controllers

Genetic Programming, as proposed by Koza, evolves computer programs in the form of syntax trees [Koza, 1992]. Since their proposal, these trees have been used for encoding programs for various tasks, such as symbolic regression, classification and controllers for both agents [Koza, 1992] and robots [Koza and Rice, 1992]. More recently, Villarreal et al. proposed to evolve tree-based controllers for controlling a single mobile robot for locating chemical sources [Villarreal et al., 2016]. Apart from syntax trees, other tree-based controllers are also present in the literature:

- **Decision trees** (DT) [Rokach and Maimon, 2005] are commonly used in classification problems to provide a class depending on the values of the features. Their inner nodes typically correspond to comparisons between feature values and optimized thresholds, whereas each leaf node corresponds to a class. In robotics, the leaf nodes may correspond to actions and the inner nodes may correspond to conditions based on the current and past state of the robot and the world, which it perceives through its sensors. DTs have been used in mobile robot motion control [Dönmez and Kocamaz, 2020] and navigation [Swere and Mulvaney, 2003, Hamzei et al., 1999, Hammad et al., 2019], robotic soccer [Huang and Liang, 2002, Sungkono et al., 2016], manipulation [Shah and Gopal, 2010], industrial robots [Moctezuma et al., 2012], security [Vuong et al., 2015] and safe human-robot collaboration [Kovincic et al., 2020].
- **Behaviour Trees** (BT) [Colledanchise and Ögren, 2018] are another type of Directed Acyclic Graphs that are commonly used for modelling the behaviours of non-player-characters in video games [Scheper et al., 2016, Sekhavat, 2017], but that have recently received some interest in the field of robotics [Jones et al., 2018, Sprague et al., 2018]. BTs are composed by inner and leaf nodes, all of which share the same interface, i.e., each node returns a status (e.g., *Success*, *Failure*, *Running*) that dictates how the tree is traversed. Inner and leaf nodes are also known respectively as flow control and execution nodes. Four types of flow control nodes are commonly used: Sequence, Fallback, Parallel and Decorator. Sequence nodes execute all of their child nodes from the left to the right until one of them returns *Failure*, returning *Success* only if all of its child also return *Success*. Fallback nodes execute all their child nodes from the left to

the right until one of them returns *Success*. It only returns *Failure* if all of its children return *Failure* as well. Parallel nodes execute all of their children simultaneously, returning *Success* if a user-defined amount of them succeed. Decorator nodes have a single child and are used to manipulate its return status. As an example, an inverter decorator may be used, returning *Failure* if its child succeeds and vice versa. Execution nodes may either be actions or conditions to be executed or tested.

Similarly to the previously described types of controllers, manually designing BTs can be a cumbersome task, often leading to sub-optimal behaviours. Some works, have resorted to Genetic Programming to evolve behaviour trees for controlling flying robots [Scheper et al., 2016] and ground robots that are part of a swarm [Jones et al., 2018]. Others [Perez et al., 2011], resorted to Grammatical Evolution, where the production rules were purposefully designed to produce meaningful and compact BTs.

From a macroscopic perspective, BTs are composed by nodes that either execute a sequence of children or constrain that execution by a set of conditions and by nodes that encode the actions to be executed. As a result, they may be reduced to the syntax trees evolved by Koza for controlling simulated agents [Koza, 1992] and robots [Koza and Rice, 1992]. Moreover, despite their popularity, BTs have rather complex structures, requiring the introduction of several constraints in the Genetic Programming algorithm and thus being more suitable to be evolved by Grammatical Evolution. Nevertheless, BTs have the advantage of enabling experimenters to manually fine-tune them when crossing from simulation to reality [Scheper et al., 2016], a feature shared with other white-box controllers that effectively reduces the impact of the reality gap.

2.1.4.2 Types of adaptation

Evolutionary Robotics (ER) approaches can also be categorised based on *when* the adaptation takes place. Following this criteria, two main categories emerge: offline and online adaptation. A third category may be adopted as a combination of the previous two. Eiben et al. [Eiben et al., 2010a] proposed a taxonomy to classify ER works according to three dimensions: (1) when; (2) where; and (3) how the evolution takes place.

When does the evolution take place

Regarding the moment when the evolution takes place, the ER works may be divided into offline and online evolution. In offline evolution, the adaptation of the individuals (robotic controllers, robotic bodies [Buchanan et al., 2020] or both) is done prior to their deployment in performing the actual task. This adaptation is typically carried out in simulation, being the best solution found validated on the real world without any further changes [Jakobi et al., 1995] (even though many works end up not performing real world validation [Nolfi and Parisi, 1996, Ziegler and Banzhaf, 2001, de Croon et al., 2013, Tuci and Trianni, 2014]). The lack of further adaptation in the real world often leads

to issues related with the Reality Gap (Section 2.1.4.3), as the individuals may perform differently in simulation and in the real world. Nevertheless, offline evolution is the most common approach in the literature, as simulations can sometimes be made to run orders of magnitude faster than real time [Ziegler and Banzhaf, 2001]. Simulations are also safer, as there is no danger of damaging the real robots, cheaper as no consumables are spent and, at times, more fair. When testing controllers in the real world, the approach often consists in doing so sequentially, with the next evaluation starting from where the previous one terminated. This means that the evaluations are not fair, as they are done under different conditions. In simulation, this problem can be easily solved by restarting the environment and using the same initial conditions for each evaluation.

Online adaptation [Haasdijk et al., 2010, Eiben et al., 2010b] refers to works where the solution to the problem is evolved in real time, and tested on performing the target task (usually in the real world). In the field of ER, this typically involves Embodied Evolution (EE) approaches [Watson et al., 2002, Nordin and Banzhaf, 1997], where the evolution of the robotic controllers and their evaluations are carried out on the physical robot, whilst it is acting on the real world. While online evolution effectively avoids the reality gap, it is much more time consuming that offline adaptation and may lead to increased wear and damages of the robots (caused by poor controllers). There is also the previously mentioned issue of unfair evaluations. The evaluation of controllers in the real world is typically carried out sequentially, where one controller starts being evaluated from the pose (state) that the previous controller ended. Thus, the fitness value of an evaluation is not only dependent of the controller, but also of the initial state that it was dealt, and consequently it is unfair to compare the quality of controllers simply through their fitness values. A method for preventing the repetition of poorly performing controllers is the use of a tabu list, as proposed in odNEAT [Silva et al., 2015].

As previously said, offline and online evolution distinguish themselves based on the moment when evolution takes place. If the controllers are evolved in simulation, and only the result of evolution is deployed on the EA without further modification (at least by the EA), we are in the presence of an offline evolutionary system. Conversely, if the evolution, and consequently the evaluation of candidate solutions, is carried out whilst the robot is performing its task, then we are faced with an online evolutionary system. The simplest combination of both is the use of offline evolution to evolve solutions in simulation which serve as good starting points for online evolution to refine [Eiben et al., 2010a]. Such approach is particularly beneficial when evolving controllers that need to be adapted to each specific robot or environment, specially when the environment is not known beforehand [Bredeche et al., 2018]. Online evolution may also be used to cope with changes to the robot over time (e.g., damages) [Cully et al., 2015, Allard et al., 2022]

Where does the evolution take place

The evolution of candidate solutions may be carried out in the robots themselves (on-board or intrinsically) or in a remote server (off-board or extrinsically). In the first case, the robots are responsible for the entire evolutionary process, ranging from individual selection, mating and evaluation. While this adds additional overhead to the robots, possibly resulting in increased energy consumption, it also enables them to adapt autonomously. In turn, in extrinsic systems the robots act only as puppets of the remote server, which performs all the steps of the evolutionary process, provides them with controllers and collects performance metrics from them. In extrinsic systems the robots are unable to improve their performance by themselves and thus, in realistic scenarios where the robots may be unable to communicate with the server, intrinsic systems would be preferable. Intrinsic and extrinsic evolution has been compared in [Nordin and Banzhaf, 1997] with the authors claiming that the only small advantage of intrinsic evolution being the lack of need to have a cable connecting the robot to the computer, which would interfere with its motions. Nowadays, the use of wireless communications would solve this issue. Thus, the main disadvantage of intrinsic systems would be the increased energy consumption, whereas their main advantage would be the ability to continue adapting even if the robots became unable to communicate with the server.

How does the evolution take place

Regarding how the evolution takes place, there have been mostly two types of approaches: distributed and encapsulated. In distributed approaches, each robot carries a single candidate solution that can only be improved through interactions with other robots (mating) [Watson et al., 2002]. In turn, encapsulated approaches [Nordin and Banzhaf, 1997, Haasdijk et al., 2010, Eiben et al., 2010b] resemble island models, with each robot carrying a population of controllers that it is able to evolve independently of others. Moreover, each robot in the community may use a different EA to evolve its population, which is executed in the traditional manner, being locally centralised. The individuals in the population are sequentially evaluated by having control of the robot for a given time period, leading to the previously mentioned unfair evaluations. Nevertheless, the combination of distributed and encapsulated approaches may be the most promising approach, as it enables each robot to evolve its own population of controllers and share them with others nearby (through mating or migration). As such, the robots that stray from the community are able to keep evolving by themselves, whilst those with neighbours may exchange genetic material to prevent premature convergence. The works of Silva et al. and Usui and Arita [Silva et al., 2015, Usui and Arita, 2003] are examples of combinations between encapsulated and distributed approaches.

Embodied Evolution

Bredeche et al. [Bredeche et al., 2018] define Embodied Evolution (EE) as an evolutionary system that is implemented over a community of at least two robots

that evolve while performing the target task. However, studies do exist where one robot is capable of evolving in isolation [Floreano and Keller, 2010, Nordin and Banzhaf, 1997] (i.e., they are online, encapsulated approaches). We consider that Embodied Evolution consists on distributing the evolutionary process over one or a community of robots, which evolve and evaluate whilst performing the target task through distributed, encapsulated or a combination of both approaches. EE systems exhibit three main characteristics: they are (1) decentralized; (2) online; and (3) parallel. The decentralization of evolution in EE is achieved by having no central authority that selects the parents for mating, evaluates the performance of the offspring and selects the survivors for the subsequent generations. Instead, the robots evaluate their performance based on locally available information and the exchange of genetic material is often constrained by their geographical proximity. As a result, the exchange of genetic material is not only dependent on the relative quality of the individuals and some predefined heuristics, but also on the behaviours exhibited by the robots over time, that lead them closer or further apart to other members of the community. The online nature of EE implies that the robots change their controllers during execution. As new solutions are evolved, they must be evaluated. That evaluation is typically carried out by allowing the new individuals to control the robot while acting on the real world. As such, the performance of the robots may actually deteriorate, as poor controllers are tested. Nevertheless, this continuous adaptation mechanism is valuable to improve the robot's performance, adapting its controller to the current state of the robot and the environment at any given moment. Finally, EE systems are parallel as the community of robots concurrently evolve solutions in an often asynchronous manner, interacting to exchange genetic material and, possibly, to perform the task.

2.1.4.3 Reality gap

In robotics, simulators are often used to perform experiments as they are a faster, safer, cheaper and overall more convenient way than performing experiments in the real world. Moreover, simulators enable testing various parameters configurations and assessing the influence of those parameters in the methods being proposed, which may be infeasible to do in the real world. However, simulators are only emulations of the real world, often existing mismatches between their environment and robot models and their physical counterparts. Those mismatches lead to the robots performing differently in simulation and in the real world, a phenomenon know as the reality gap [Collins, 2022]. The existing simulators have various degrees of fidelity, which is due to a trade-off between the realism of the simulators and computational efficiency. More realistic simulators model their components with further detail and consequently becoming slower and thus less appropriate for works that require many simulations. Evolving and learning robotic controllers are among such works, with both communities endowing efforts to tackle the reality gap [Mouret and Chatzilygeroudis, 2017, Salvato et al., 2021] in what are known as sim-to-real methods [Collins, 2022]. The existing approaches may be divided into methods that attempt to improve the simulations and those that attempt to produce controllers that cope

better with the reality gap.

Improving the simulations

Improving the simulations typically requires collecting real-world data to update (or calibrate) the simulator (often by adjusting its parameters) [Mehta et al., 2020]. An example of techniques that attempt to improve the simulations is the back-to-reality approach by Zagal et al. [Zagal et al., 2004, Zagal and Ruiz-Del-Solar, 2007]. This method co-evolves the robot controller and simulator through a three-stage iterative process: first, a genetic algorithm is applied to evolve the parameters of the robotic controller in simulation; second, a set of well-performing controllers are evaluated in the real world and a reinforcement learning [Sutton and Barto, 2018] algorithm (i.e., a machine learning paradigm where an agent learns through its interactions with the environment, receiving a reward for each action made) is applied to improve them; thirdly, a genetic algorithm is applied to evolve the parameters of the simulator, with the goal of minimizing the difference between the performance of the robotic controllers in simulation and in the real world. It is important to note that they tested this approach in a locomotion task, where the subject of evolution was a set of parameters for a hand-designed controller. Also, only a set of parameters for the environment were evolved. Thus, this approach may not yet be applicable to more complex problems and even where it is applicable, it results in an increased overhead of having to iterate over simulated and real world evaluations to improve the simulator.

A different path was suggested by Mouret and Chatzilygeroudis, who claimed that in the future it is unlikely that simulators will be fast and accurate enough to enable the evolution of controllers with no reality gap [Mouret and Chatzilygeroudis, 2017]. Instead, they suggest that the simulators should be able to provide an estimate of their confidence in the fitness value assigned to each individual, which may then be used by the individual selection schemes. Such simulators could be based on Monte Carlo methods (e.g., running many experiments with different simulation parameters and assessing how they transfer), or through crowd-based methods, where experimenters worldwide would build a database with their results in transferring evolved robots to the real world, along with the parameters used.

Creating robust controllers

There are many methods through which the evolved controllers may become more robust to the reality gap. One of the earliest approaches [Jakobi et al., 1995] evolved ANNs to work as controllers for a Khepera (a small two-wheeled differential-drive robot) and studied how the injection of noise into the simulation affects the transferability of those controllers. They started by modelling the noise of the robot's sensors and actuators and used those models to create three types of simulations: (1) zero noise, where no noise is added; (2) normal noise, where Gaussian noise equivalent to that observed is added to the simulation; and (3) double noise, where the Gaussian noise added to the simulation

has double the standard deviation of that observed in the real world. They compared the behaviours of the robots in simulation and in the real world and concluded that the controllers transferred best when the noise injected into the simulator was the most similar to that of the real world.

Other researchers have noted that increasing the level of abstraction may increase the ability to cope with the reality gap. Some types of controllers are more susceptible to the reality gap than others. That is the case of ANNs whose high representational power may often lead them to overfit details of the simulation that do not translate to the real world [Francesca et al., 2014]. In turn, graph-based controllers enable experimenters to manually fine-tune them when crossing from simulation to reality [Scheper et al., 2016], effectively reducing the impact of the reality gap. Lee at al. [Lee et al., 2018] support these ideas by showing that by focusing on learning the high-level decision making policy and delegating the low-level perceptions and control to proven frameworks or pre-trained models, the policy can be transferred to the real world with no additional adaptation. Also, graph-based high-level controllers are often applicable to a diverse variety of robotic platforms, provided that the actions and perceptions are devised to each specific robot. The aforementioned abstraction may also be achieved with ANNs, by evolving them to act solely as higher level controllers that arbitrate between hand-designed behaviours [Duarte et al., 2012a] or other ANNs specially trained for specific tasks [Duarte et al., 2012b], rather than ANNs that attempt to fulfil the complete task, including sensor signal processing, decision making and outputting the motor commands. Similarly, [Duarte et al., 2014] the ANNs evolved to perform behaviour primitives (e.g., move to waypoint, patrol a region, etc) may be combined by a behaviour arbitrator, possibly containing multiple layers, which may either be evolved, manually designed or a blend of both.

A third approach is to evolve initial controllers offline and later refine them online [Nolfi et al., 1994]. Such approach not only provides faster evolution over online methods, but also reduces the risk of damaging the robot or poor controllers leading it to undesirable regions of the environment. The online adaptation is not only useful for adapting the controller to the intricacies of each robot, but also enables adapting to the state of the robot over time, accommodating wear, damages or simply changes to the sensors, actuators or the environment.

In turn, Mouret and Chatzilygeroudis [Mouret and Chatzilygeroudis, 2017], claimed that techniques such as the transferability approach, MAP-Elites and novelty search with local competition may be promising paths to coping with the reality gap, by evolving individuals that are expected to transfer better to the real world.

MAP-Elites [Mouret and Clune, 2015] is an evolutionary approach which evolves not one, but a set of diverse and high-performing candidate solutions for a given problem. In order to do so, the user must define not only a fitness function, but also a set of phenotypic features that characterize a solution. As an example, in the case of evolving robot morphologies, those features could be related with the shape of the robot, amount of wheels or limbs, number of sensors, energy

efficiency, etc. The user must also define a discretisation for each feature, so that an N-dimensional grid can be created. MAP-Elites then searches within each cell for the best performing solution. In [Cully et al., 2015] MAP-elites is used to evolve not one, but a set of controllers in simulation. Once the robot is deployed in the real world, it monitors its performance, considering any performance drops as indications of damages unanticipated in simulation. In such event, the robot engages in an iterative process, testing the pre-evolved controllers in its archive, to find one that copes acceptably with its damage. In this approach no modification is made to the actual controllers, but rather the archive of pre-evolved controllers is searched to find one that suits the current state of the robot. As a metaphor, the authors compare this approach to an animal attempting to find the best way to limp, minimising the pain arising from an injury.

The transferability approach [Mouret et al., 2013] uses a supervised learning [Jo, 2021] algorithm (i.e., a machine learning paradigm where a labelled dataset is used to train a model) to learn the limits, i.e., to learn mappings between behaviour descriptors and predictions of simulation accuracy. Those mappings are then used by the EA to select the individuals that not only have good fitness values in simulation, but also are expected to transfer well to the real world.

Finally, novelty search [Lehman and Stanley, 2011] is an evolutionary method that attempts to avoid local optima by explicitly searching for novel behaviours. It does so by replacing the fitness function with a novelty metric, being the novelty of each individual computed as the difference between the behaviours of the current individual and those in the current population and in an archive of past individuals. Local competition is added to novelty search so that only individuals with similar behaviour compete with each other through fitness. Through the combination of novelty search with local competition and the transferability approach [Cully and Mouret, 2016], the authors were able to evolve a repertoire of locomotion behaviours that enabled a legged robot to reach every location within a working space with a limited number of real-world evaluations.

2.1.4.4 Evaluation functions for evolutionary robotics

Learning and evolutionary approaches for automatically designing robotic controllers differ in various ways. Yet, both require evaluation (also called fitness or reward) functions to guide their search. In Reinforcement Learning, the evaluation (reward) function provides a reward for each performed action, enabling the algorithm to adjust the policy to maximise the reward. In ER, the fitness function provides a quality value for each candidate solution, enabling comparing them and ultimately influencing their ability to survive and reproduce.

When faced with a function optimisation problem, the target function itself is the fitness function. In classification tasks, one may resort to well-established metrics such as the precision, recall or F1-score. In ER, the evaluation function should accurately reward the desired behaviours without specifying their low-level implementation [Nelson et al., 2009], as to prevent restricting the EAs'

ability to find novel ways to perform the task. Moreover, the fitness function should be as informative as possible, avoiding plateaus that cannot quantify small differences between candidate solutions, causing the evolution to stagnate. The ideal fitness function for OSL should be able to evaluate how well a given strategy *searches* for the chemical source, rather than evaluating how well it *finds* the source, doing so as quickly as possible and without restricting the EAs' ability to find novel ways to fulfil the task.

Designing fitness functions for ER is not an easy task, as there is the possibility of misaligned goals, unfair and noisy evaluations and, in the case of OSL, dynamic environments.

Misaligned goals

One of the difficulties in designing fitness functions for ER and specially for OSL is known as misaligned goals [Zhuang and Hadfield-Menell, 2020]. The goals expressed in the fitness function may either be an incomplete representation of the user's desires, or represent slightly different things. The common result is that the EA will find candidate solutions that have a good fitness value, but that do not behave as the user intended.

The phenomenon of misaligned goals often arises when attempting to make a sparse evaluation function more dense. Evaluation functions may be categorised into dense and sparse functions. A dense evaluation function provides a non-zero reward for each action of the agent. In turn, sparse evaluation functions only provide non-zero feedback on specific events, such as when colliding with an obstacle. As a result, the agent often performs a variable-length sequence of actions without any feedback [Sutton and Barto, 2018]. Dense evaluation functions are more desirable and often used in academic reinforcement learning problems. In the real world, devising evaluation functions that provide meaningful rewards to the agent at each time step is a difficult task as often the experimenters are not able to specify how a task should be made, but only whether it has been done successfully.

The process of making the evaluation functions more dense is known as reward shaping and the concept was firstly introduced by Mataric [Mataric, 1994]. This approach has several disadvantages, such as being a trial-and-error process that requires human experts. Moreover, it often restricts the learning ability of the algorithm by introducing bias. Another disadvantage is the aforementioned alignment problem, where the learning agent finds a way to exploit features of the evaluation function, achieving high rewards without performing the intended behaviour and effectively overfitting the evaluation environment.

Unfair evaluations

The evaluations are said to be unfair if two candidate solutions are evaluated under different conditions. Unfair evaluations are common in embodied evolution, a type of ER where the controllers are evaluated online, while operating in the real world. In embodied evolution the candidate solutions are typically evalu-

ated sequentially, with the following controller starting from where the previous one ended. Thus, the controllers may either be at an advantage or disadvantage relative to other members of the populations depending on their start conditions, and consequently it is not fair to directly compare their fitness values.

Noisy evaluations

The evaluations in ER are often noisy, i.e., multiple evaluations of the same candidate solution produce different results [Jin and Branke, 2005]. The noise arises not only from slight misalignments in the robot's initial pose, but also from the robot itself. Robots are inherently noisy entities. Their sensors provide uncertain readings and their actuators (e.g., wheels) are subject to changes in friction, wear, and other environmental conditions that make them behave differently from what was intended. As an example, a robot moving on a surface may encounter different materials or irregularities that cause one wheel to slip and consequently an alteration to the robot's trajectory. Even if there is no slippage, the motion of the wheels is controlled according to the readings of encoders with finite resolution and the tolerances in gearboxes and couplings also contribute to deviations from the intended route.

Dynamic environments

Evaluating OSL search strategies becomes even harder in the presence of dynamic environments. While attempting to locate an odour source, a robot must take into account its state, the state of the environment and, possibly, the states of its teammates, all of which are continuously changing over time. While in realistic scenarios the airflow and chemical dispersion patterns are uncontrolled (and consequently unrepeatable), in simulation, they are created through stochastic processes, being possible to create different conditions between evaluations. These dynamics further contribute to the noise of the evaluations, as two separate evaluations may have quite distinct instantaneous chemical dispersion and airflow patterns, even if their general characteristics are the same.

Naive approach - multiple evaluations

A common approach to cope with noisy evaluations is to extract quality metrics from multiple evaluations of each solution [Coppola et al., 2020, Divband Soorati and Hamann, 2015, Jin and Branke, 2005]. The fitness value is often the sum or average of the fitness of each evaluation and ten evaluations are often used [Jones et al., 2018, Ampatzis et al., 2008, Francesca et al., 2014].

Another reason for performing multiple evaluations is the possibility of the EA overfitting the evaluation scenario. The chances of overfitting are particularly higher when performing a single evaluation with a simple evaluation function, which enable poor strategies to attain good fitness by chance. In turn, performing multiple evaluations with distinct conditions is likely to reduce overfitting by exposing the controller to various scenarios. As an example, Jones et al. [Jones et al., 2018] evolve robotic controllers (BTs) for foraging. They perform

various evaluations with a group of robots starting from the same position but with different orientations.

In the case of OSL, evaluating a search strategy through the final distance to the chemical source or the time needed to reach it, evaluates only the strategies' ability to reach the source and not how well it searches for it. Thus, even a random walk might attain a satisfying fitness. The aforementioned approach of performing multiple evaluations also aids in preventing overfitting, as long as each evaluation is performed under different conditions [Coppola et al., 2020]. In the case of OSL, the robustness of the evolved strategies could be increased by performing multiple evaluations with the same overall parameters, but differing on the position of the chemical source, the initial pose of the robot and different instantaneous airflow and chemical dispersion.

Performing multiple evaluations with simple functions is an easy way to cope with noisy evaluation and reduce the possibility of overfitting, while introducing no additional bias into the evolutionary process. Unfortunately, the evaluations in ER are typically very time consuming [Coppola et al., 2020], rendering approaches that perform multiple evaluations less desirable, specially considering online evolution applications.

Other approaches

An alternative to using multiple evaluations was explored by Lehman et al. [Lehman et al., 2012], who propose a method to increase the robustness (success rate in testing) of evolved search strategies from a single evaluation. The method consists on encouraging the agent to pay attention to the environment through reactivity. This is done by computing the mutual information (i.e., the mutual dependence) between the signals from the sensors and the actuators. The higher the mutual information, the more dependence exists between the sensors and actuators. This approach was tested with a khepera robot in a maze navigation task. Hence, only the signals from 6 proximity sensors were used and the actuators could only perform one of three actions: moving forward, turning left or right. The authors compared their approach with a baseline method, using only one evaluation and not rewarding reactivity. They also compared with three methods that used multiple evaluations, each with a different level of noise applied to the robots' sensors and actuators. The results showed that, in the task at hand, the proposed method often evolved more robust strategies that repeated noisy evaluations, particularly at lower noise levels.

Another way to increase the robustness of the evaluations is by increasing the amount of prior knowledge in the fitness function, rewarding other traits exhibited by the candidate solutions. Nelson et al. [Nelson et al., 2009] propose a taxonomy for the fitness functions in ER, classifying them into seven groups, sorted by increasing amount of prior knowledge:

- **Aggregate** fitness functions are the simplest type of fitness functions enunciated and contain the least amount of prior knowledge. These functions evaluate only the robot's ability to complete the task, with no regard

to how it is achieved. As an example, in a foraging task, the fitness value could simply be the amount of items deposited at the nest location by the end of the evaluation. These functions are often dismissed in the ER community as they provide little to no gradient for the EA to evolve, which is particularly necessary in the initial generations. Nevertheless, they introduce the least amount of bias, allowing for the production of complex and unexpected behaviours.

- **Competitive and co-competitive** fitness selection contain low to moderate amounts of prior knowledge. Competitive fitness selection consists of evaluating multiple individuals from the same population while they operate simultaneously in the same environment, so that the actions of one may influence the others. As an example, while performing a given task in the same environment, two robots may collide, lowering their performance and consequently receiving worse fitness values. Co-competitive fitness selection methods apply to systems co-evolving at least two populations of controllers, each for a different task. The controllers of the existing populations are evaluated simultaneously on the same environment, influencing each other's behaviours. Such systems are typically used for evolving predator-prey controllers and often result in more complex behaviours than if the populations evolved in isolation. This is the result of one population acquiring better performance, pushing the other to evolve better controllers in an attempt to reach an equilibrium.
- **Environmental incremental** fitness functions consist of increasing the complexity of the environment where the robot operates over the course of evolution. As a result, as the EA is able to evolve controllers to operate on a simple environment, the environment becomes more complex, gradually approximating the target. In these approaches, the prior knowledge is introduced through the environments, as the user must define scenarios that increase in complexity in a way that aids the evolution of the controllers.
- **Tailored** fitness functions can be seen as an extension to aggregate fitness functions, encompassing components for measuring both how the robots behave and how well they achieve the target task. In their paper, the authors exemplify tailored fitness functions with the evolution of controllers for a phototaxis task. In such scenario, the controller could simply be evaluated by whether it reached the light source i.e., it either succeeds or fails to reach the goal position. A behavioural component could also be included to encourage the robot to face the light source. Such behavioural component may make the evolution faster, but may also lead to controllers that perform poorly in more complex environments and may restrict the EAs ability to evolve novel ways to perform the task. The combination of these aggregate and behavioural terms constitutes a tailored fitness function.
- **Functional incremental** fitness functions are deeply connected to incremental evolution and are meant to evolve complex behaviours, which are difficult to evolve from scratch. They consist on complexifying the goal

of the evolution over time, starting with selecting individuals according to their ability to perform a simple desired ability upon which the intended complex behaviour can be built. Once an acceptable fitness level has been achieved, the fitness function is modified to move the evolution towards a more complex behaviour. This cycle is repeated until the target behaviour is achieved. The main disadvantages of functional incremental fitness functions are the possibility of restricting the EA's ability to evolve novel ways to perform the target task and the difficulty inherent to decomposing complex behaviours into simpler ones, which can be evolved incrementally.

- **Behavioural** fitness functions have the second highest amount of prior knowledge introduced. These are task-specific functions, often composed by various components that evaluate how the robots behave, rather than how well they fulfil the task. The authors give an example of evolving controllers for obstacle avoidance, where the experimenter may design an evaluation function that rewards robots that turn when sensing an obstacle in front of them. In such situation, the evolution is not seeking controllers that avoid obstacles, but rather those that exhibit the behaviour designed by the experimenter (even though it may be sub-optimal in performing the task).
- **Training data** fitness functions consist on providing the EA with a dataset of desired state-action mappings and evaluating how well the evolved controller matches this dataset. It introduces the most amount of prior knowledge into the evolution, as the dataset thoroughly describes the intended controller response to each input. These fitness functions are often used to mimic the behaviour performed by a human controller.

Various authors attempted to devise guides for designing proper fitness functions. Wilkerson and Tauritz [Wilkerson and Tauritz, 2011] proposed a guide for designing fitness functions for common applications and argue that the problem requirements should be converted into components of the fitness functions. They also state that those components should not be binary, but rather provide smooth and informative gradients to aid evolution, avoiding plateaus that do not reward small performance changes. More recently, Soorati and Hamann [Divband Soorati and Hamann, 2015] conducted a study of the influence of different fitness functions in evolutionary robotics (ER). They use the taxonomy proposed by Nelson et al. [Nelson et al., 2009] to divide the fitness functions into different classes depending on the amount of prior knowledge included. Following this classification, the evaluation functions typically used in OSL can be classified as aggregate fitness functions, as they only evaluate what was achieved (i.e., was the source found) rather than evaluate how well the search was performed. This type of evaluation functions have the advantage of introducing little prior knowledge, thus reducing the possibility of biasing the evolutionary process. However, they allow poor strategies (e.g., random search) to receive good performance values by chance. Examples of aggregate fitness functions are the distance travelled and the time spent [Lochmatter, 2010], [Liberzon et al., 2018]. Works such as that of Gongora et al. [Gongora et al., 2017] attempt to produce more meaningful evaluation functions through combining variables.

Apart from the success rate, they evaluate the strategies through an accuracy index (which is based solely on the distance to the odour source) and, more interestingly, through an efficiency index which combines the final distance to the odour source and the time spent. Tailored fitness functions introduce some prior knowledge in the form of how the robot should behave, coupled with metrics of task completion. As an example, Croon et al. [de Croon et al., 2013] used an evaluation function that evaluates: (1) how well a strategy identifies an odour source; (2) the mean concentration sensed during the trial; and (3) the final distance to the odour source. The authors evaluate each strategy multiple times, being its fitness value the average of all evaluations. This poses similar limitations to the evaluation through success rates (used in [Lochmatter, 2010], [Gongora et al., 2017]): it slows down the learning process.

Many real world problems require optimising multiple objectives. One of the simplest ways of doing so, is to combining the multiple objectives into a single objective functions by assigning weights to them. However, these objectives often have different magnitudes and assigning weights to them is typically not trivial. Some works propose to use desirability functions [Mostaghim et al., 2010, Trautmann and Weihs, 2006] to map each objective into the [0,1] interval in a non-linear manner. While such transformations solve the problem of different magnitudes, they do not solve the issue of proper weight assignment. On the other end of the spectrum, Multi-Objective Evolutionary Algorithms (MOEA) [Deb et al., 2002, Emmerich and Deutz, 2018] typically attempt to approximate the Pareto-front, producing a set of non-dominated solutions. These approaches have the advantage of requiring less design effort, but they also have the disadvantage of assigning equal importance to all objectives. In the middle of the spectrum sits Lexicographic Parsimony Pressure [Luke and Panait, 2002], which was proposed with the purpose of optimising the quality and size of GP trees. This method works by optimising the objectives in sequence. In the original work, the method was implemented as a modified tournament selection operator, where if two trees attained the same performance, the smallest one was chosen. This way, it is possible to optimise various objectives by order of preference. To cope with problems where there are few individuals with the same fitness, the authors proposed two bucketing methods. These methods consist of assigning the individuals from the population into buckets, depending solely on their fitness. In the modified selection operator, the buckets are used instead of the actual fitness values, being the individuals from the same bucket considered equivalent in terms of performance and selected only based on their sizes.

2.1.5 Robotic communities

The works using more than one robot may be divided in two categories: multi-robot or swarm robotic systems. Multi-robot systems encompass small teams of highly capable, possibly heterogeneous, robots that despite being able to solve the task individually, cooperate to increase their performance. They differ from robotic swarms (RS) in the sense that swarms are composed by many simple agents that would either be unable to perform the task on their own or their performance would significantly improve by cooperating. The robots are often

guided by simple and homogeneous controllers and through their local interactions, complex behaviours emerge that enable them to accomplish their task as a group [Şahin, 2004, Trianni, 2008]. Despite the aforementioned distinctions, the line separating multi-robot and swarm approaches is often blurry. As a result, in this thesis we shall treat them together.

Robotic swarms have various advantages over single-robot approaches [Şahin, 2004, Brambilla et al., 2013, Hamann, 2018, Coppola et al., 2020]:

- **Robustness** to the loss of agents is among the main advantages of robotic swarms. It implies that the swarm should be apply to accomplish the task after loosing part of its agents, even if at the cost of efficiency [Hamann, 2018]. The robustness to the loss of one (or a few) robots is due to the behaviours emerging from local interactions of the agents and existing no centralised point of failure. As each robot operates through local interactions and locally obtained information, its loss should only produce local effects that could be compensated by nearby robots. The robustness to the loss of agents is also increased by the high levels of redundancy arising in swarms composed by homogeneous robots, so that the functions of a broken down robot can be assured by another nearby agent. Moreover, the typical simplicity of robots applied in swarms when compared to those employed in single-robot approaches for the same task makes them less prone to malfunctions and the spatially distributed sensing inherent to having multiple agents increases the robustness of the readings by increasing the signal-to-noise ratio. Note that the level of robustness to the loss of agents is dependent on the size of the swarm as well as on the percentage of lost agents.
- Another advantage is the ability to employ **inexpensive robots**, which are typically less expensive than robots capable of performing the same task on their own. This may also be seen as a type of robustness, as reduces the cost of loosing agents. Such robots are typically too limited to fulfil the task by themselves, but do so by cooperating. Yet, there are some fundamental functions that each robot must be able to perform for the swarm to succeed. Autonomous and safe navigation along with the ability to sense the environment and other robots as well as relative localisation capabilities are among those basic functions [Coppola et al., 2020].
- Due to the inexistence of centralised control and each robot interacting only with its local neighbours and performing its own computations, robotic swarms attain high levels of implicit **scalability**. As a result, swarms are able to cope with changes in size, caused by robots straying away, malfunctioning or new robots being introduced into the environment, without drastic effects in their performance. Due to the aforementioned reasons, there may be virtually no limit to the size of a swarm, existing however the need to maintain an adequate density of robots [Hamann, 2018]. An example of large swarm is the Kilobot project [Rubenstein et al., 2014], where 1024 robot autonomously self-assemble into pre-defined 2D forma-

tions with various shapes.

- **Flexibility**: Due to the organisation of the robots being based solely on local interactions, the swarm can reconfigure to cope with the dynamics of the environment or to tackle different tasks, such as cooperative transportation, moving through tight spaces or even self-assemble to overcome obstacles. The ability of swarms to adapt to various tasks is heightened through cooperation, which enables the robots to overcome their limitations in actuating, sensing and communicating [Hamann, 2018].

As previously mentioned, the emergent behaviour of the swarm is a result of the local interactions between the agents, which in turn is a consequence of their communication limitations. The robots that compose swarm systems typically do not possess global communication capabilities. Instead, they may communicate through one of three means: (1) stimergy, which consists of sensing and leaving marks on the environment. The most common example of stimergy is the deposit and sense of pheromones, similarly to ants; (2) using the sensors (e.g., cameras, microphones) to sense the environment and nearby robots (which may communicate through leds, dance like the bees, sound, etc.); and (3) messages sent through low-range wireless communication. These messages may either be directed to a specific robot or broadcast to all robots within range. [Nedjah and Junior, 2019].

Robot formations

In the literature, the works involving robotic communities arrange the robots in one of three modalities: (1) rigid formations, where the robots are either physically connected [Baldassarre et al., 2007, Şahin, 2004]) or the position of each robot is clearly specified [Ristic and Gilliam, 2019]; (2) flexible formations (e.g., flocking), often created by virtual force fields [Reynolds, 1987]; or (3) use no formation at all, being the robots free to move independently of others [Bredeche and Fontbonne, 2022]. Robotic communities have many applications in the real world [Coppola et al., 2020]. In some instances, such as foraging, it may be more fortuitous to move with no specific formation, scattering the agents in the environment. In turn, rigid formations may be useful for collective transportation, while flexible formations may perform better in complex environments.

2.2 Source seeking approaches

Source seeking approaches attempt to use environmental measurements to select actions that guide the robot to the location of the chemical source. They are often inspired by natural behaviours, such as those of the *E. coli* bacteria, dung beetle or silkworm moth [Russell et al., 2003]. In order to propose such approaches, researches have made several efforts to understand how animals behave while performing this task. As an example, Duistermars et al. [Duistermars et al., 2009] investigated how fruit flies responded to stimuli in their antennas to track odour plumes, while Weissburg and Dusenbery [Weissburg and Dusenbery, 2002] and Dickman et al. [Dickman et al., 2009] experimented with blue crabs

in controlled environments in order to find a correlation between their behaviour when tracking a food source and their detection of its scent. While this works are extremely interesting, in this section we shall focus on literature involving robots.

2.2.1 Single-robot approaches

Most source seeking approaches employ a single robot, as it tends to be simpler by not requiring any coordination or communication mechanisms to be implemented over multiple agents. Due to animals' abilities to locate odour sources, many plume tracing methods are directly inspired by natural behaviours, such as those of the male silkworm moth, dung beetle and *E. coli* [Russell et al., 2003] or combine traits from different species such as the behaviour of bacteria and Lévy walks [Nurzaman et al., 2011].

Source seeking methods may be further divided into chemotactic approaches, which rely solely on the chemical information, and anemotactic approaches, which also use information of the flow (be it air or water). In the remaining of this section we shall present some of the most relevant chemotactic and anemotactic methods.

Chemotaxis

Chemotactic approaches are among the simplest methods for locating chemical sources. They are designed for environments deprived of a strong airflow, where the odour spreads mainly through molecular diffusion. In such environments, the agent relies solely on information regarding chemical concentration to guide its search. In environments with strong winds, the turbulent effects of the airflow create intermittent odour plumes, with many voids and local peaks of concentration, that are likely to deceive a gradient-based approach. Nevertheless, Ishida et al. [Ishida et al., 2012] note that, in regions close to the odour source, the chemical gradient can be informative enough to be followed and thus, approaches designed for diffusion-dominated environments may still be able to successfully locate the odour source.

One of the earliest chemotaxis works was conducted by Rozas et al. [Rozas et al., 1991]. It consists on a robot equipped with an array of chemical sensors to measure the concentration in four (ninety degree-spaced) directions and moves in the direction of highest concentration. If after this motion no gradient is detected, the robot moves in one of the other three directions. Conversely, if the gradient lowers, the robot moves back to the previous position and proceeds to move in one of the other three directions. This method was tested with a real robot in distances up to three meters from the source.

The chemotactic strategy inspired by the behaviour of the *E. coli* bacteria [Marques et al., 2002b] is one of the most popular ones. It owes most of its popularity is to its simplicity, as it is a biased random walk composed only of rotations and linear motions. On each time step, the agent measures the local chemical concentration and compares it to the previous odour measurement. If

the current concentration is higher, the agent makes a small rotation followed by a large straight motion, continuing searching in the same approximate direction. Otherwise, it makes a probabilistically larger rotation followed by a short straight motion, directing the search to a different direction.

Another simple approach for locating odour sources in diffusion-dominated environments is inspired by Braitenberg's Vehicles [Braitenberg, 1986]. This strategy attempts to estimate the local chemical concentration gradient and follow it to its source. There have been many different variants of this approach [Russell et al., 2003], but all have some things in common. Generally, they rely on a mobile robot carrying two front-mounted chemical sensors, one on its left and another on its right. The controlling strategy typically consists of a loop, which makes the robot move forward whilst turning towards the sensor sensing the highest chemical concentration.

Grasso et al. [Grasso et al., 2000] further developed the gradient approach and used it to control a purpose-built robot for mimicking a lobster. This robot is a two-wheeled differential-driven unit, equipped with two front-mounted chemical sensors. They proposed two control strategies. The first only differs from the traditional gradient approach by including a threshold of concentration, below which the robot does not turn. The second approach, adds a retreat behaviour, designed to move the robot back where it came from, whenever the chemical concentration sensed drops below a given threshold.

Spiral [Ferri et al., 2009] is another chemotactic method for locating odour sources in diffusion-dominated environments. It consists of making consecutive spiral motions, restarting every time the robot considers that the gas source is closer than in the previous step. The robot estimates the distance to the source by stopping and taking chemical concentration measurements for a predefined time period, which are then used to compute a proximity index (PI). If the current PI value is higher than the stored PI, the robot considers that it is currently sensing odour (hit) and restarts the spiral. Otherwise, it considers it as a non-detection (miss). Finally, the authors also devised an escape movement that is triggered when a spiral ends without any hit. In such case, the robot resets the stored PI value, rotates to a random direction and moves straight for a predefined length, starting to explore a different region of the environment.

Anemotaxis

In environments where there is a strong airflow, animals typically employ strategies that use information regarding its direction for orienting the search, i.e., perform anemotaxis. This section describes some of the most popular anemotactic methods.

The Silkworm Moth algorithm [Kowadlo and Russell, 2008, Russell et al., 2003] is inspired by the behaviour of the Male Silkworm Moth while tracking a trail of Bombykol pheromone released by a female moth. This algorithm assumes that the robot is equipped with two chemical sensors, mimicking the moth's antennas. The signals from these sensors are used to compute a concentration gradient

that the robot uses to select the direction of some of its motions. The behaviour created by this approach is based on three basic types of movements: (1) straight line upwind surges when detecting odour, and (2) upwind-centred zigzag and (3) circular motions whenever contact with the plume is lost. A flow chart of the complete behaviour inspired by the Silkworm Moth is depicted on the left side of Figure 2.6. The robot starts by waiting for a chemical detection, upon which it moves straight upwind. At the end of this upwind surge, if the robot continues to sense odour, it will repeat the same motion. Otherwise, it will engage in an upwind zigzag in an attempt to re-encounter the plume. If it manages to re-encounter the plume, the robot will go back to perform upwind surges. Otherwise, it will perform circular motions, halting as soon as odour is sensed. At the end of these motions, the robot will go back to waiting for chemical detections to re-initiate the search behaviour. Liberzon et al. proposed another algorithm inspired by the Silkworm Moth, that differs from the previous ones by using only a single binary chemical sensor and no stereoscopic information [Liberzon et al., 2018].

Similarly to the Silkworm Moth, researchers took interest in the behaviour of the Dung Beetle tracking a cow's pat [Russell et al., 2003]. However, contrarily to the Silkworm Moth's approach, in the Dung Beetle algorithm the robot starts with a plume finding behaviour, moving crosswind in search for odour cues. Upon detecting odour, it assumes an odour-centred upwind zig-zag behaviour for tracking the plume to its source. A flow chart of this behaviour is presented on the right side of Figure 2.6.

Harvey et al. [Harvey et al., 2008] described a casting behaviour for plume finding inspired by the wasp *Cotesia rubecula*. It consists of moving back and forth across the wind, with straight motions of increasing length. In their paper, each straight motion has double the length of the previous one. The authors also compared plume-tracking methods inspired by the behaviours of flying insects:

- **Surge-Anemotaxis:** This behaviour consists of, upon detecting a chemical concentration above the predefined threshold, moving the robot upwind for a fixed length, whilst continuously readjusting its heading. If the chemical concentration sensed during the upwind surge drops below a predefined threshold, the robot will resort to a casting behaviour.
- **Bounded Search:** This behaviour consists on exploring the upwind region of the location where odour is first detected, in an attempt to find the chemical source. To do so, upon sensing odour, the searching agent starts making subsequent crosswind and upwind motions, creating a triangular of parabolic shaped trajectory. The trajectory is reset whenever the chemical concentration sensed is above a predefined threshold. Conversely, if a predetermined time limit is exceeded, the agent resorts to a casting behaviour.
- **Counter-turning:** The behaviour consists of performing an upwind zigzag motion, while sensing odour. The angle and length of each motion is determined by the chemical concentration sensed. If the concentration is

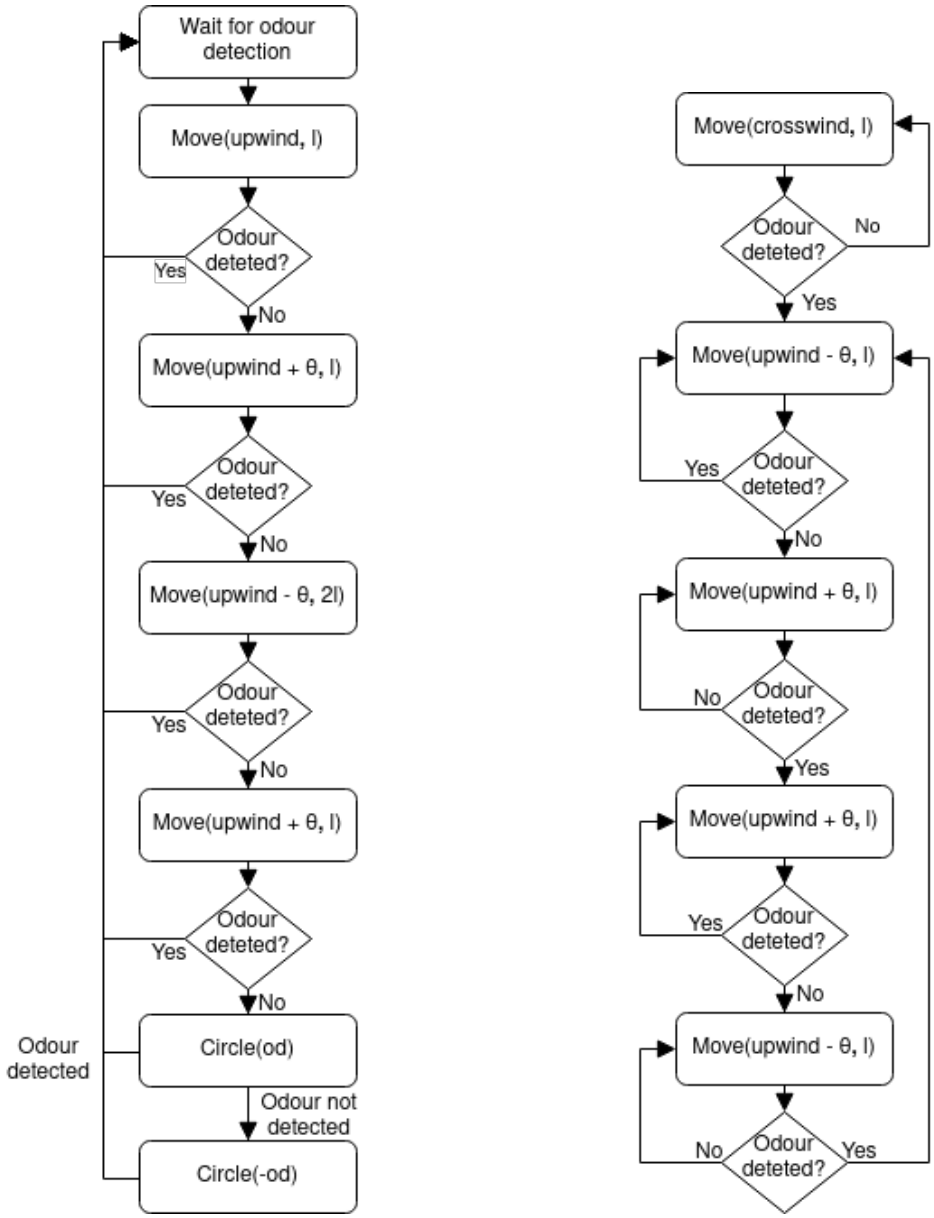


Figure 2.6: Flow charts of the modified Silkworm Moth (left) and Dung Beetle (right) algorithms, adapted from [Russell et al., 2003]. θ and s are user-defined parameters that, respectively, control the amplitude of the rotations and the length of the straight motions.

high, the robot will robot will move for a short distance and with a small offset to the upwind direction. On the other hand, when the concentration is low, the robot will move for longer distances and with larger offsets that may approximate crosswind.

Real-world environments often have dynamic conditions where the wind velocity varies, existing periods with no airflow and others with strong winds. For coping with such scenarios, Ishida et al. [Ishida et al., 1995] proposed a method that combines several bio-inspired behaviours. When the robot is within the plume, this strategy employs an anemotactic behaviour inspired by the upwind surges of the Silkworm Moth. Whenever the plume is lost, the robot resorts to a casting behaviour to attempt to re-encounter it. Casting is particularly important, as due to the random nature of odour plumes, the robot may lose contact with it. However, this anemotactic strategy was found to fail in some situations, such as whenever there are various wind sources. To tackle this problem, the authors added a chemotactic strategy which uses solely the chemical information to attempt to locate the gas source. The decision to switch between strategies is based on the chemical concentration measured. Whenever the chemical concentration is below a predefined threshold, the robot employs the chemotactic strategy. Otherwise, it will resort to the anemotactic approach. They also added a timeout condition that changes between strategies if no progress is made in a predefined time interval. A flow chart of this behaviour is depicted in Figure 2.7 and comprises five distinct stages:

1. **Waiting for chemical detection**: The first step of this strategy consists on waiting for an initial chemical detection.

2. **Follow the chemical concentration gradient to the source**: The second stage of this process consists of performing chemotaxis, i.e., using the chemical concentration gradient to move closer to the odour source. It is employed when the gas concentration is below a predefined threshold ($C_{chemotaxis}$) which is considered to be caused by unstable wind conditions.

3. **Retreat:** Once the robot loses contact with the plume, it resorts to the retreat behaviour, which consists of moving back in the direction from where it came from. As soon as odour is detected, the robot moves back to stage 2, unless if in the previous step it has sensed the highest chemical concentration in its memory. In such case, it considers that the current location is the most promising for detecting chemical information, and moves to stage 1.

4. **Track the chemical plume:** If the sensed chemical concentration is higher than a threshold, the robot resorts to an upwind search behaviour for tracking the plume. During this behaviour, the concentration gradient is used to bias the motion of the robot towards the centre-line of the plume.

5. **Crosswind search for plume finding:** When the robot loses contact with the plume during the tracking phase, it resorts to search crosswind. As soon as the plume is re-encountered, the robot goes back to stage 4, tracking the plume upwind. If the two crosswind directions have been searched twice and no gas has been detected, the robot goes back to stage 2.

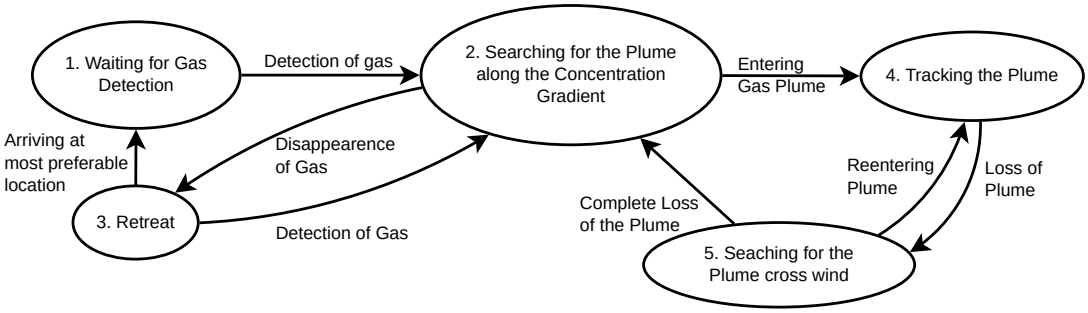


Figure 2.7: Flow chart of the Multiphase strategy proposed by Ishida et al. [Ishida et al., 1995].

2.2.2 Multi-robot and swarm approaches

While most odour source localisation methods employ a single robot, some methods using multiple robots do exist. As previously said, these approaches may be divided between multi-robot and swarm approaches which differ, among other things, by multi-robot approaches often having more complex interactions between the robots than swarm approaches. As an example, Hayes et al. [Hayes et al., 2002] proposed a multi-robot approach where the robots communicate by broadcasting messages. Three types of communication are compared:

- NONE, where the robots do not communicate with each other and effectively search for the source individually;
- KILL, which as soon as one robot senses odour forces all other robots to stop searching. This type of communication effectively uses all robots to find the chemical plume and only one for tracking it to its source, thus reducing the overall energy consumption of the robotic team;
- ATTRACT, which makes the robots not sensing odour move towards those that are sensing it.

Their experiments showed that the KILL communication method effectively reduces the energy consumption of the group and that the ATTRACT strategy does not produce any performance gains in their scenario, spending more energy than KILL. Another multi-robot approach was proposed by Majorvi et al. [Marjovi et al., 2009], who used a group of robots to explore indoor structured environments and locate the sources of fires. The robots had no prior knowledge of the environment and cooperated to build a shared topological map of it. The map is shared in a central server and is the sole mean of communication among the robots. The robots use a frontier-based approach to explore the environment. Each robot selects the next frontier to explore based on its cost, which is proportional to the distance and is computed through the A* algorithm. This algorithm not only provides the cost of each frontier, but also the path to it. Fires are detected by a sensor apparatus mounted on each robot, capable of sensing the concentration of various chemicals as well as temperature and humidity. The navigation of each robot to a target location relies on an artificial potential field, composed by repulsive forces from the nearby obstacles and an attractive

force from the goal location. When a robot senses a possible fire, it performs a sequence of motions attempting to triangulate its source or discarding a false positive.

Fluxotaxis [Zarzhitsky et al., 2005] is a swarm-based approach for locating odour sources that resorts to artificial physics to keep the robots in a lattice formation. The robots share their environmental measurements to compute the derivatives of flow-field variables (i.e., wind velocity and chemical concentration derivatives) and, by following the gradient of the mass flux (i.e., the product of chemical density and flow velocity), the robots are able to locate the chemical source. Another swarm-based approach for locating chemical sources was proposed by Lochmatter et al. [Lochmatter et al., 2013]. Their work focused only on the tracking stage, and they propose to create loose line formations along the crosswind direction through a set of attractive and repulsive virtual forces. The robots track the plume by proceeding upwind while attempting to keep the formation centred on the plume's centerline. The authors experiment with formations of three and five robots in a real wind tunnel, using a laminar airflow and a stationary source emitting at a constant rate. There are three possible start conditions, but in all of them, at least one robot is sensing odour. The results show that three robots are sufficient for successfully finding the source, with the five robot formation not attaining any significant improvements. The authors perform a second experiment, where the source moves crosswind, but it is still successfully found by the three-robot formation.

2.2.2.1 Meta-heuristic approaches

A different strand of work formulated the odour source localisation problem as an optimisation problem, where the function to be optimised is the chemical dispersion in the environment. A number of meta-heuristic based methods have been proposed for this task, be it Particle Swarm Optimisation (PSO) [Marques et al., 2006, Jatmiko et al., 2007, Cabrita et al., 2013, Feng et al., 2020, Duisterhof et al., 2021], Genetic Algorithms [Marques et al., 2002a] or Evolutionary Strategies [Marques et al., 2003].

Regarding the use of EAs, Marques et al.'s approach iteratively evolved a population of candidate locations for the source, which are encoded as vectors of real-valued coordinates [Marques et al., 2002a]. The evaluation of the candidate solutions is made by having the robots navigate to them, being the concentration measured at the target location used as fitness. They only used crossover, as the assignment of the locations to each robot and their navigation was considered to introduce enough randomness into the process. This method was able to perform well in simulation, in an environment without wind-flow, being the odour dispersion dictated only by molecular diffusion. They also studied the influence of the robots' start position, showing that the algorithm performed better when the robots start from random positions but that it is slow to converge when they start clustered in a corner of the working space, which may be considered to be a more realistic scenario. Later, Marques et al. improved on this work, introducing a directed mutation operator that biased the search towards upwind

if odour was detected at the current location or towards crosswind if no odour was detected [Marques et al., 2003].

Marques et al. proposed one of the first methods based on Particle Swarm Optimization (PSO) for locating multiple chemical sources with a group of robots [Marques et al., 2006]. They tackle the plume search and tracking separately, being the search sub-problem addressed simply through crosswind motions. In turn, they propose to treat the tracking process as an optimisation problem, where the goal is to find the location of maximum odour concentration. The robots act as particles of the PSO, measuring the local chemical concentration and sharing them with their neighbours. The algorithm itself does not cope with locating various sources. Instead, once a source is found, it is disabled, halting the emission of chemical substances, and the robots implicitly attempt to locate other sources. The authors compared the performance of the PSO with a gradient search method and an *E. coli* inspired biased random walk, showing that the PSO outperforms the others as the environments become more unstable.

Jatmiko et al. proposed CPSO, a modification to the PSO algorithm for tracking various sources [Jatmiko et al., 2007]. Once a promising region of the search space is found, the swarm splits into two groups: neutral and charged robots. Inspired by Coulomb's law, repulsive forces are applied to the charged robots, modifying their velocities and driving them away from the others. As a result, they escape the local optima and resume exploring the search space. In turn, the neutral robots are not subject to repulsive forces, performing local search in the promising region. They compare CPSO with DRPSO, a PSO approach devised for dynamic environments that considers that the environment has changed if the global best has not changed for an extended time period. In such case, the robots are commanded to scatter for a predefined distance, in an attempt to escape the local optimum. Their simulations showed that CPSO locates the chemical source faster than DRPSO, whilst the standard PSO method would often get trapped in local optima.

Another PSO-based method was proposed by Cabrita et al. to locate multiple chemical sources [Cabrita et al., 2013]. Their approach works by estimating the distribution model of the chemical emitted by each source and virtually cancelling it. The robots then proceed to search for other chemical sources by finding odour filaments that are consider to not have been emitted by the previously found sources.

More recently, Feng et al. proposed a PSO-based method to locate odour sources with a group of robots without using wind information [Feng et al., 2020]. They modify the original PSO algorithm by introducing random variations into the personal and global best positions at each velocity computation, which aid the algorithm to escape local optima. Moreover, they employ an initial divergence search strategy, leading the robots to scatter in different directions until the first chemical filament is sensed. The authors tested this approach with three robots in an indoor uncontrolled environment, achieving higher success rates than the standard PSO, at the cost of longer search times. Duisterhof et al. also proposed a PSO-based method for locating odour sources, this time with a

group of nanocopters [Duisterhof et al., 2021].

2.2.3 Automatically designed approaches

Evolutionary Robotics is a field of Artificial Intelligence that focuses on using Evolutionary Computation techniques for automatically designing the shape [Murata and Kurokawa, 2007] or the controllers for one or more robots [Nolfi et al., 2016]. As mentioned in the previous section, EAs have already been for directly controlling the robots searching for odour source, by estimating waypoint locations [Marques et al., 2002a, Marques et al., 2003]. In this section, we focus on works that resort to EAs to automatically produce robotic controllers. Most existing works propose to evolve the controllers in the form of Artificial Neural Networks (ANN) [Beer and Gallagher, 1992, de Croon et al., 2013, Macedo et al., 2016]. While [Beer and Gallagher, 1992, de Croon et al., 2013] focused on only evolving the connection weights and time constants of Continuous-Time Recurrent Neural Networks (CRTNN), [Macedo et al., 2016] used a direct representation of the individuals to evolve both the connection weights and topologies of ANNs. Each evolved network combines a set of attractive-repulsive forces into a potential field that guides a robotic swarm. The method was tested in simulation, being able to consistently move the swarm closer to the odour source than the algorithm inspired by the *E. coli* bacteria. Izquierdo et al. also evolved the parameters of ANNs to perform Klinotaxis, i.e., chemotactic searches in salt gradients [Izquierdo and Lockery, 2010]. However, they did not use a mobile robot, but rather a worm-like agent which move by contracting and relaxing neck muscles.

Singh et al. [Singh et al., 2023] trained Recurrent Neural Networks to act as the controller for a simulated agent searching for a chemical source. The networks were trained through Proximal Policy Optimization [Schulman et al., 2017], a Deep Reinforcement Learning algorithm. At each time-step, the controller receives the egocentric wind velocity and chemical concentration and produces an angular and linear velocities. They analysed the behaviours exhibited by the trained controllers and split them into modules that resemble the behaviours of insects for tracking and reacquiring chemical plumes.

Genetic Programming has also been used for OSL by Villareal et al., who evolved chemotactic strategies for locating a chemical source in a small indoor environment with a single robot [Villarreal et al., 2016]. The controllers included functions with different arities and expecting different types of arguments (i.e., real-valued thresholds and variables to reason about chemical concentrations, motion primitives or more complex subtrees). The terminal symbols consist on different methods for measuring chemical concentration and motion primitives that can be programmed differently for various types of robots (e.g., rotate 45^{o} to the most promising direction depending on the last chemical measurement). It is an offline evolutionary approach, being the controllers evolved in simulation and only the best individual tested in the real world.

2.3 Source estimation approaches

Source estimation approaches attempt to use environmental measurements to fit a chemical dispersion model, estimating the location of the source without the need to reach its location.

2.3.1 Single-robot approaches

To the best of our knowledge, Farrell et al. proposed the first method based on Hidden-Markov Models for source term estimation [Farrell et al., 2003]. Their method could not only estimate the source probability map, but also determine which cells are more likely to contain odour; determine the most likely trajectory of odour between an assumed source location and a target cell; and determine which path between two locations and with a given length is more likely to result on odour detections. Their approach works by dividing the environment into a grid, where each cell stores the probability of the chemical source residing in that location. This method still requires improvements, as it assumed that the flow velocity is invariant over the entire search space. Also, it relied solely on boolean odour detections, discarding information regarding the concentration, which may aid the estimation of the distance to the chemical source.

Following the work of Farrell et al, Vergassola et al. proposed Infotaxis [Vergassola et al., 2007], which to this date is the most popular source estimation approach. It is also based on Hidden Markov Models and Bayesian Inference to compute the probability map for the location of the odour source, but contrarily to Farrel et al.'s work, it also provides a movement strategy for the robot. On each control step, the robot is commanded to move in the direction that locally maximises the expected information gain (i.e., it moves in the direction that is expected to provide the largest reduction in the entropy of the probability map). This method also disregards the actual gas concentration values and considers only detections (hits) and non-detections (misses). Considering an agent moving in an environment, T_t encodes the sequence of positions and odour hits/misses of said agent up to time t. The probability map ($P(r_{src}|T_t)$) for the odour source location (r_{src}) is computed through the Bayes' law:

$$P(r_{src}|T_t) = \frac{P(T_t|r_{src})P(r_{src})}{P(T_t)} \tag{2.18}$$

where $P(r_{src})$ is the prior for the position of the source. $P(T_t|r_{src})$ is the likelihood for the position of the odour source and, assuming that the misses and hits are independent, can be computed as:

$$P(T_t|r_{src}) = \prod_t P(h(r_t)|r_{src}) \tag{2.19}$$

where r_t is the (hypothetical) location of the robot at time t and $h(r_t)$ is 1 if the robot senses odour at r_t and 0 otherwise. The probability of detecting odour at r_t ($P(h(r_t) = 1|r_{src})$) is 1 - $P(h(r_t) = 0|r_{src})$ (i.e., 1 minus the probability of not

detecting), which in turn is computed as:

$$P(h(r_t) = 0|r_{src}) = e^{(-R(r_t,r_{src})\Delta_t)} \tag{2.20}$$

where Δ_t is the duration of the time step and $R(r_t, r_{src})$ is the average detection rate at position r_t, given the source location r_{src}. The average detection rate (Equation 2.21) is computed based on a gas distribution model that takes three parameters which are not easily measured in real scenarios: the odour source emission rate R (measured in particles per second), the particle lifetime τ and the isotropic effective diffusivity D_f.

$$R(r_t|r_{src}) = \frac{aR}{|r_t - r_{src}|} e^{-\frac{|r_t - r_{src}|}{\lambda}} e^{\frac{(x_t - x_{src})V}{2D_f}} \tag{2.21}$$

where a is the radius of the searching agent's sensor, V is the average wind speed along its mean direction and λ is defined as:

$$\lambda = \sqrt{\frac{D_f \tau}{1 + \frac{V^2 \tau}{4D_f}}} \tag{2.22}$$

At each time step the robot may take one of 9 possible actions, i.e., it may remain still or move to one of the 8 neighbouring cells. The action selected is the one that maximises the expected entropy reduction in the source probability map. One of the main drawbacks of Infotaxis is the computational overhead of computing the required probabilities for each cell of the grid. Another, perhaps worse, drawback is the need for carefully selecting the parameters of the gas distribution model, so that it accurately matches the real world. The importance of these parameters has been assessed in previous studies, showing that inadequate values greatly reduce the performance of Infotaxis [Ruddick et al., 2018, Rodríguez et al., 2017, Martin Moraud and Martinez, 2010]. As a result, the majority of Infotaxis' literature perform the experiments exclusively in simulation, using the same exact model for mimicking the real world and for computing the probability of the odour source location. Sampling these models does not accurately emulate real world experiments, as they provide a distribution of the average odour concentration, rather than an instantaneous plume. The realism of those experiments is further reduced by the common assumption that the odour detections and non-detections are independent events.

Ristic et al. [Ristic et al., 2016] proposed Infotaxis II, which differs from the original Infotaxis by replacing the grid-based probability map computation by a particle filter. This modification not only enables reducing the computational cost, as well as overcoming the limited resolution of the position estimation, which in original Infotaxis is dictated by the choice of the grid cell size. The authors compare three reward functions: (1) the original one from Infotaxis, which evaluates the expected entropy reduction from moving to each location through the probability of finding the source or sensing odour at that location; (2) Infotaxis II reward function, which is a modification of the original reward function consisting of discarding the term relative to the possibility of finding the source

at the next movement; and (3) a reward function based on Bhattacharyya distance, which measures the distance between posteriors at the current and next time step. Their results showed that there is little difference between the performance attained with the three reward functions. Moreover, they concluded that the ratio between the area of the search space and the area with detectable odour is of the utmost importance, with the source estimation strategies performing worse than systematic search when the area with detectable odour is very small. In turn, in scenarios where odour is detectable in at least half the search space, the source estimation strategies become very efficient, with the two new reward functions providing better results than the original Infotaxis reward function.

More recently, Hutchinson et al. [Hutchinson et al., 2018] proposed another variation of Infotaxis termed Entrotaxis. Similarly to Infotaxis II, this method relies on a particle filter to estimate the source parameters, thus reducing the computational cost over the original Infotaxis. It differs from Infotaxis II in the decision to where move next. While Infotaxis II moves in the direction that minimizes the expected entropy of the posterior distribution, Entrotaxis moves in the direction of maximum entropy of the predictive distribution of odour measurements. In short, Infotaxis moves the robot in the direction that is expected to maximise the information gain, whereas Entrotaxis commands the robot to move in the direction where least is known regarding the odour that is to be detected. By not having to compute all hypothetical posteriors in the decision making process, Entrotaxis manages to be less computationally expensive than Infotaxis II. Moreover, their results show that Entrotaxis attains similar success rates to those of Infotaxis II, while doing so with smaller mean search times.

2.3.2 Multi-robot and swarm approaches

To the best of our knowledge, the few existing multi-robot source estimation approaches are based on Infotaxis. In 2019, Ristic et al. [Ristic and Gilliam, 2019] proposed a distributed particle filter-based Infotaxis approach. Their method consists of a community of robots, which form a connected communication graph. Each each robot estimates the location of the source independently of the others and all robots are treated equally, making it a completely decentralised approach that is robust to failures. The robots share their environmental measurements with their neighbours, and use all available information in their estimates. Each robot selects the action that is expected to maximise the entropy reduction of its probability map and shares its intention with its neighbours, engaging in a iterative process to reach a consensus on which direction to move. Reaching a consensual action is a requirement of this approach, as the shape of the robot formation must be kept unchanged. Thus, this approach is decentralised, yet synchronous, as the robots must first reach a consensus with all others before being able to move. If for some reason a robot gets lost from the others, it might re-encounter the group, but only by chance. The search for the source is also based on a consensus, with all robots halting their search as soon as one of their neighbours reaches the termination criteria.

A similar approach was proposed by Park and Oh [Park and Oh, 2020], who proposed to distribute Infotaxis II [Ristic et al., 2016] over a community of agents. This approach requires the robots to form a fully-connected communication graph and also synchronises when the sampling and moving takes place across all robots. Similarly to the original Infotaxis, on each control step the robots may take one of five possible actions: remaining still or moving to the front, back, left or right. The authors focused on studying the influence of different levels of coordination on the performance of the robots: non-coordination, passive coordination and negotiated coordination (also called cooperation). In non-coordination, each agent acts independently of the others, sharing no information. In the passive coordination, each agent computes its best action using the measurements from all robots. In the negotiated coordination, the robots engage in a negotiation to reach a consensual group action. Their simulation results showed that using more robots with no cooperation does not lead to any performance gains. Passive cooperation leads to higher success rates and faster search times than no coordination, but the negotiated cooperation method is the overall most successful and fastest of the three.

A different strand of work focuses on how information should be shared among the robots. Song et al. [Song et al., 2019] proposed Social-Infotaxis, an extension to the original Infotaxis to a multi-robot setting, where the robots share their environmental measurements to estimate the position of the chemical source. Each robot estimates its own probability map and the measurements from the other robots are weighted before being used to update the probability map. The weights are assigned in the [0,1] interval, where 0 implies discarding other robots' perceptions and searching individualistically, whereas 1 means assigning the same importance to the neighbours' measurements as the robot's own measurements. The authors propose two methods of setting the weights: (1) fixed homogeneous coupling, where the weights are pre-defined by the experimenter and kept fixed over the robots and throughout the entire experiment; and (2) dynamically heterogeneous coupling where the weights vary in time as well as between each pair of robots. The weights depend on the Kullback-Leibler divergence between the source probability maps of the two robots. The higher the divergence, the lower the weights, causing the robot to assign less importance (or even disregard) the neighbour's measurements. On the other hand, equal probability maps cause the robot to assign the same importance to the other robot's measurements as it does to its own. This approach is particularly interesting for scenarios with unreliable measurements, preventing the robot's probability map to be corrupted by incorrect measurements from its teammates. However, this approach suffers from a high computational cost, not only due to using the original Infotaxis approach, but also by requiring the robots to exchange their entire probability maps to compute the Kullback-Leibler divergence. This drawback was later addressed in [Song et al., 2020], where a particle-filter was used to estimate the source probability map and a Gaussian density function was fitted to the particles, so that only its mean and covariance matrix needs to be exchanged between robots to compute the weights for social estimation of the source location.

2.4 Robotic communities for related tasks

The existing controllers for swarms can be categorised depending on whether they are manually or automatically designed. [Nedjah and Junior, 2019]

Manually designed controllers

Most existing approaches rely on experimenters to carefully design the individual behaviour of each robot [Brambilla et al., 2013]. This is typically a cumbersome trial-and-error process which becomes much more daunting when considering multiple robots that must interact with each other to fulfil a given task. As a result, they tend to be sub-optimal but also better understood [Coppola et al., 2020] and, in some instances, it is possible to guarantee the convergence of the system to the desired properties [Saulnier et al., 2017]. This implies that hand-designed controllers tend to be more predictable and thus it is easier to verify if they meet the safety requirements. For that reason, many real-world implementations of multi-robot systems rely on hand-designed controllers [Chung et al., 2018], particularly when considering aerial robots. The same concerns have made their way into the automatic design community, motivating the choice of white-box controllers that may be visually inspected [Francesca et al., 2014, Francesca et al., 2015, Kuckling et al., 2018, Jones et al., 2019].

The works proposing hand-designed controllers for robotic communities may follow a bottom-up approach, or a top-down [Mermoud et al., 2014, Coppola et al., 2020]. Bottom-up (or microscopic) approaches focus on designing individual behaviours and interactions, iterating on them until the desired collective behaviour is achieved as a consequence of the agents' interactions. Such approaches assume that the global state of the system is too hard, costly or even impossible to obtain, so they make due with the locally accessible information, obtained by the agent and its neighbours [Crespi et al., 2008]. The main advantage of bottom-up approaches is its direct correspondence with reality, enabling the precise design of each individual behaviour and being faced with the limitations of the robots right from the start. In turn, their main drawback is the difficulty in analysing the system as a whole, depending heavily on the designer's intuition to achieve the desired collective behaviour. On the other hand, top-down (or macroscopic) approaches start by devising a high-level model for the intended collective behaviour, being the individual controllers designed a posteriori with certain guarantees of convergence [Mermoud et al., 2014, Coppola et al., 2020]. These approaches typically start by being designed as centralized methods, assuming global knowledge of the system [Crespi et al., 2008]. That assumption is often relaxed at a later design stage, with the system becoming decentralized through communication between agents. However, the agents are still expected to be able to get or estimate the global state of the system within a certain accuracy and time delay [Crespi et al., 2008]. One of the disadvantages of this type of design approach is the need for strong assumptions which may not hold in the real world (e.g., perfect localisation, absence of sensor and actuator noise and even discrete environments), leading to loss of performance. Another disadvantage of top-down approach is their difficulty, due to the experimenter being

faced with the entire complexity of the swarm system.

The most popular types of hand-designed swarm controllers are finite-state machines (FSM) and their probabilistic variants (PFSM) along with physics-based design [Brambilla et al., 2013]. PFSMs are composed by states and transitions. Each transition between states is associated with a probability, which may either be fixed during the entire execution or be the result of a mathematical function dependent on a set of parameters of the system. PFSMs are often applied in tasks involving aggregation [Soysal and Sahin, 2005], chain formation [Nouyan et al., 2008], self-assembly [O'Grady et al., 2010], task-allocation [Liu et al., 2007, Labella et al., 2006] and collective transport [Wilson et al., 2014]. Another example is the work of McGuire et al. [McGuire et al., 2019] who designed a FSM inspired by bug algorithms to guide each agent in a group of six drones to explore an unknown environment. The devised controller is quite simple, leading the robots to move away from others, avoid obstacles by adopting a wall following behaviour and moving randomly otherwise.

In turn, using virtual physics-based controllers, the robots are led by artificial potential fields, created through the sum of attractive/repulsive forces. In single-robot approaches, the potential fields typically attract the robot through a goal whilst repelling it from obstacles. In swarm robotics, the potential fields may also include forces attracting/repelling the robot to/from its neighbours. The main advantage of its approach is its simplicity, as once the forces are properly designed and balanced, a single mathematical function maps each robot's sensory inputs to its motion command. Moreover, it also lends itself to theoretical analysis through which properties such as robustness and scalability may be proven. Virtual physics-based approaches are typically employed in tasks requiring robot formations, such as pattern formation [Spears et al., 2004, Shucker and Bennett, 2007, Shucker et al., 2008], collective exploration [Howard et al., 2002] and coordinated motion [Maxim et al., 2009, Ferrante et al., 2012].

Some hand-designed methods make the robots move in formations, as they may be a more robust and predictable way of sampling the environment or may actually be necessary to perform the target task (e.g., cooperative transportation). Several formation control algorithms are reviewed in [Chung et al., 2018]. However, in some scenarios moving in rigid formations may be detrimental. Flocking methods create flexible formations by attempting to mimic the behaviours of animals. The first attempts to simulate flocking behaviours date back to the 1980's [Aoki, 1982, Reynolds, 1987] and were not related with robotics, but rather abstract agents, respectively inspired by fish and birds, but with no particular sensors or motion restrictions. The formations in flocking methods are kept through a virtual force field composed by three simple rules:

- **Separation** is the process through which an agent attempts to move away from its neighbours, which is ultimately useful for collision avoidance;
- **Alignment** consists in attempting the match the velocity of nearby agents, so as to move in the same average direction and at the same speed;
- **Cohesion** is the rule that keeps the formation together. It leads each

member of the swarm to move towards the centroid of its neighbours;

Additional rules may be included to enable the swarm to fulfil various tasks. As an example, collision avoidance may be achieved by including additional repulsive rules from the obstacles in the environment [Saska, 2015]. Flocking approaches have been demonstrated in the real world [Hauert et al., 2011] with 10 robots.

Evolutionary design

Given the hurdles of manually designing the swarm controllers, some efforts have been made to design them automatically, namely through evolutionary methods. As an example, Genetic Algorithms have been used to evolve the connection weights of manually designed ANNs to serve as controllers for drones, tasked with creating a communications network between ground stations [Hauert et al., 2009] or making formations [Scheper and De Croon, 2017]. In [Hauert et al., 2009], the authors highlighted the difficulty in hand-designing the controllers and analysed the behaviour of the evolved controllers to provide insight for future hand-design attempts. In turn, in [Scheper and De Croon, 2017] two networks were actually evolved, one performing low-level motion control and another performing high-level decision making.

White-box controllers have also been evolved for robotic swarms. Szabo [Szabo, 2015] evolved behaviour trees to perform collision avoidance, which showed to be not only smaller than the author's hand-designed controller, but also performed better. Another example is the work of Jones et al. [Jones et al., 2018], who evolved behaviour trees for a swarm of robots in a foraging task. The robots have a short communication range and use hop counts to estimate the distance to the nest and food regions. Despite the robots being able to communicate, it is left to the evolution to decide what to communicate. Also, each robot may use a simple form of memory, whose meaning is also left for evolution to define.

Multi-robot and swarm controllers may be evolved offline or through Embodied Evolution (EE, Section 2.1.4.2). Regarding EE, due to the capabilities of the robots, encapsulated or hybrid approaches are more suitable for multi-robot systems, whereas fully distributed approaches are more suitable for swarm robotics. In swarm robotics, the controllers are typically homogeneous and the fitness evaluates the performance of the entire swarm [Waibel et al., 2009]. Being homogeneous, they are immune to issues inherent to heterogeneous multi robot systems, such as task allocation and credit assignment [Parker, 2008]. According to Waibel et al. [Waibel et al., 2009], works involving robotic swarms may be categorised based on two dimensions: (1) how the performance is measured, i.e., whether the fitness evaluation measures the quality of the entire swarm or that of a single individual; and (2) whether the swarm is composed by homogeneous or heterogeneous agents. As an example, if EE is employed to evolve swarm controllers, it is likely that the controllers are not homogeneous. Yet, this may arguably be considered as a more realistic swarm approach, as the robots not only rely on local interactions to fulfil their task, but also to evolve the controllers. Other common characteristics of evolutionary swarm robotics works

include: (1) using a classical, centralised EA; (2) using large populations (in the scope of ER) with usually 100 individuals; (3) making multiple evaluations of each controller to cope with fitness noise. Existing works use between 3 and 100 evaluations of each candidate solution, but most of them use 10; (4) most works rely only on simulations and, those that use real robots, often do not test them in realistic scenarios [Francesca and Birattari, 2016].

One example of embodied evolution is the work of Bredeche et al. [Bredeche and Fontbonne, 2022], who proposed to completely evolve controllers for a robotic swarm in the real world, while the robots are performing their task. They focused on a foraging task, and studied the social learning effects of each robot sharing the connection weights of its governing perceptron with the other robots that it encounters while moving in the world. In turn, the receiving robot chooses to accept the genes depending on the self-assessed fitness of both itself and the sender. This is yet another difference to common EAs. Here, there is no centralised controller that assesses the performance of the robot, or the performance of the team as a whole. Instead, each robot evaluates its own performance, which must contribute as much as possible to the performance of the team. As there is only local communication, the robot cannot infer the performance of the whole swarm, and must estimate it from its own performance. The authors exemplify with a foraging task where the performance of the swarm is directly proportional to the amount of items collected. As a result, the more items each robot collects, the better should the performance of the swarm be. However, if there are too many robots competing for resources in a constrained region, they end up spending most time avoiding collisions. Thus, they would be better off exploring other regions than greedily attempting to collect the same items. In another work [Bredeche, 2014], Bredeche employed mEDEA to evolve the controllers for a swarm of robots. mEDEA is a fully distributed Embodied Evolution approach, where the robots are considered to have limited communication capabilities and thus can only exchange genes with others nearby. Being fully distributed, the robots are unable to adapt their behaviours individually. Instead, they broadcast mutated copies of their gene to their neighbourhood. Nearby robots accept the genes and store them on an archive. As soon as the evaluation time of the current gene runs out, a new gene is randomly selected from the archive and all others are removed. Thus, the selective pressure is not based on the fitness value, but rather on the ability of a gene to drive the robot to the vicinity of many others.

Chapter 3

Geometric Syntactic Genetic Programming

Contents

The existing crossover operators for Genetic Programming often cause bloat, i.e., the uncontrolled growth of the size of the candidate solutions without a corresponding increase in performance. Such growth is not only detrimental for the execution of the candidate solutions, which require more memory and time resources, but also hinders their readability. The readability of the evolved solutions is of particular importance in the scope of this thesis, as we aim to evolve robotic controllers than can be inspected by humans and manually tweaked for specific conditions. Moreover, the efficiency of the controllers is also highly relevant, as it enables using robots with limited computational capabilities. This chapter presents a geometric crossover operator that acts on the syntactic space of Genetic Programming, implicitly preventing bloat.

3.1 Geometric syntactic genetic programming

As described in Section 2.1.3.4, geometric crossover operators produce offspring that are on a shortest path linking its parents, while geometric mutation operators produce new individuals in the neighbourhood of the original ones. As a result, they enable a better control of the exploration/exploitation balance of the EA.

Geometric Syntactic Genetic Programming (GSynGP) [Macedo et al., 2018] differs from the other GP algorithms by performing geometric crossover between two individuals in the syntactic space. The genotype of each individual is a string that encodes a syntax tree in prefix notation. The crossover operation uses the Longest Common Subsequence (LCS) to align the genomes of the two parent individuals. Two modification masks are created, marking the locations where genes must be added, removed or replaced to make a copy of one parent more similar to the other. In order to create valid offspring, the proportion of non-terminal and terminal symbols must be kept. As a result, each iteration of this crossover consists of performing one of four modifications: (1) removing one terminal symbol and inserting another of the same type; (2) removing a non-terminal symbol and inserting another of the same type; (3) removing a terminal and a non-terminal symbol; and (4) inserting a terminal and a non-terminal symbol.

The proposed crossover operator starts by computing the LCS through dynamic programming, to determine the similarity between two parent individuals (A, B) and stores that information in a matrix C. Afterwards, using C, Algorithm 3.1 computes two modification masks, M_A and M_B, that contain the common and non common genetic material. The masks are created by going through matrix C, aligning the common genetic material and inserting blank spaces or markers in the positions where symbols must be added or removed from A, to make it more similar to B. The markers are provided by function *get_symbol*, which returns the symbol passed as a parameter along with a prefix to denote whether it belongs to the terminal ($T\_$) or function set ($F\_$).

The remaining steps of the crossover operator, presented in Algorithm 3.2, con-

Algorithm 3.1: Modification masks generated from the Longest Common Subsequence.

```
1  Function LCS_MASKS(A, B, C):
2      M_A, M_B ← []
3      i ← len(C) − 1
4      j ← len(C[0]) − 1
5      while i ≥ 1 or j ≥ 1 do
6          if i > 0 and j > 0 and C[i − 1][j − 1] = C[i][j] then
7              M_A ← get_symbol(A[i − 1], function_set)
8              M_B ← get_symbol(B[j − 1], function_set)
9              i ← i − 1
10             j ← j − 1
11         else if i > 0 and C[i − 1][j] = C[i][j] then
12             M_A ← get_symbol(A[i − 1], function_set)
13             M_B ←' '
14             i ← i − 1
15         else if j > 0 and C[i][j − 1] = C[i][j] then
16             M_A ← ' '
17             M_B ← get_symbol(B[j − 1], function_set)
18             j ← j − 1
19         else if i > 0 and j > 0 and C[i − 1][j − 1] = C[i][j] − 1 then
20             M_A ← A[i − 1]
21             M_B ← B[j − 1]
22             i ← i − 1
23             j ← j − 1
24     return reverse(M_A), reverse(M_B)
```

Algorithm 3.2: Geometric Syntactic Crossover Operator

```
1  Function crossover(M_A, M_B):
2      candidates ← []
3      if 'F_' in M_A and 'T_' in M_A then
4          candidates ← candidates ∪ delete(M_A, M_B, function_set)
5      else if not' F_' in M_A and 'T_' in M_A then
6          candidates ← candidates ∪ replaceT(M_A, M_B, function_set)
7      else if 'F_' in M_B and 'T_' in M_B then
8          candidates ← candidates ∪ insert(M_A, M_B, function_set)
9      else if 'F_' in M_B and 'F_' in M_A then
10         candidates ← candidates ∪ replaceF(M_A, M_B, function_set)
11     return random(candidates)
```

Algorithm 3.3: Insertion of a function and a terminal symbol

```
1  Function insert(M_A, M_B):
2      combs ← all_combinations()
3      while len(combs) > 0 do
4          (f, t) ← random(combs)
5          M_A[t] ← M_B[t]
6          M_A[f] ← M_B[f]
7          if check_indiv(M_A, function_set) then
8              return M_A
9          else
10             M_A[t] ←' '
11             M_A[f] ←' '
12     return M_A
```

sist simply in checking four conditions to select the appropriate operations. The four possible operations are: inserting a function and a terminal symbol (Algorithm 3.3), removing a function and a terminal symbol (Algorithm 3.4), removing a terminal symbol and inserting another one and removing a function symbol and inserting another one (Algorithm 3.5). The operation of deleting a terminal and inserting another one is identical to what is presented in Algorithm 3.5, with the difference that $F\_$ should read $T\_$ and *function_set* should read *terminal_set*. The function *all_combinations*() returns all pairs of symbols to be tested in each case, that is, in Algorithm 3.3, the combinations of all function and terminal symbols that are present in B and absent in A; in Algorithm 3.4, all pairs of terminal and function symbols that are present in A and absent in B and; for the operations of deleting and inserting symbols of the same type, all pairs of symbols of the desired type that are present in one parent and absent in the other one.

Algorithm 3.4: Deletion of a function and a terminal symbol

```
1  Function delete(M_A, M_B):
2      combs ← all_combinations()
3      while len(combs) > 0 do
4          (f, t) ← random(combs)
5          v ← [M_A[f], M_A[t]]
6          M_A[t] ←' '
7          M_A[f] ←' '
8          if check_indiv(M_A, function_set) then
9              return M_A
10         else
11             M_A[t] ← v[1]
12             M_A[f] ← v[0]
13     return M_A
```

Algorithm 3.5: Deletion of a function and insertion of another function symbol

```
1  Function replaceF(M_A, M_B):
2      combs ← all_combinations()
3      while len(combs) > 0 do
4          (f_b, f_a) ← random(combs)
5          if M_A[f_b] = ' ' then
6              M_A[f_b] ← M_B[f_b]
7              v ← M_A[f_a]
8              M_A[f_a] ← ' '
9              if check_indiv(M_A, function_set) then
10                 return M_A
11             else
12                 M_A[f_b] ←' '
13                 M_A[f_a] ← v
14         else if 'F_' in M_A[f_b] then
15             M_A[f_b] ← M_B[f_b]
16         else
17             v ← M_A[f_a]
18             M_A[f_a] ← M_B[f_b]
19             if check_indiv(M_A, function_set) then
20                 return M_A
21             else
22                 M_A[f_a] ← v
23     return M_A
```

An individual is then selected from the set of valid generated offspring, i.e., all offspring whose genomes can be converted into valid syntax trees with no exceeding genes. These operations create an individual that is one step away from the first parent. In order to create offspring at different distances from each parent, Algorithm 3.2 may be iterated multiple times, with the offspring of one iteration taking the place of A in the following iteration. As an example, if the operator is applied twice, on the first iteration it will be applied to parents A and B, outputting an offspring O_1. In the second iteration, the offspring O_1 will take the place of A in the crossover, creating the individual O_2. Generally speaking, in an iteration where all operations are possible, the offspring created only has a 25% chance of becoming larger than its parent, and that growth will be only by 2 nodes.

In the scope of this operator, the distance between two individuals $D_{A,B}$ may be computed through the number of modifications necessary to make A equal to B, as follows:

$$D_{A,B} = i + d + r_f + r_t \tag{3.1}$$

where i, d, r_f and r_t respectively stand for the number of insertions, deletions and replacements of function and terminal symbols needed to make A equal to B. Each of these quantities is computed as follows:

$$i = \min(B_f, B_t) \tag{3.2}$$

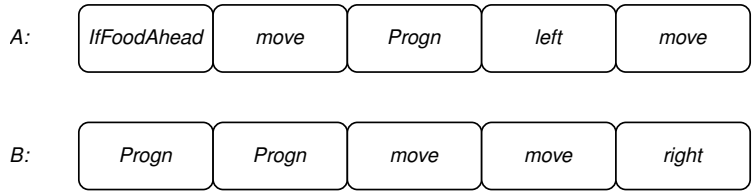
$$d = \min(A_f, A_t) \tag{3.3}$$

$$r_f = A_f - d \tag{3.4}$$

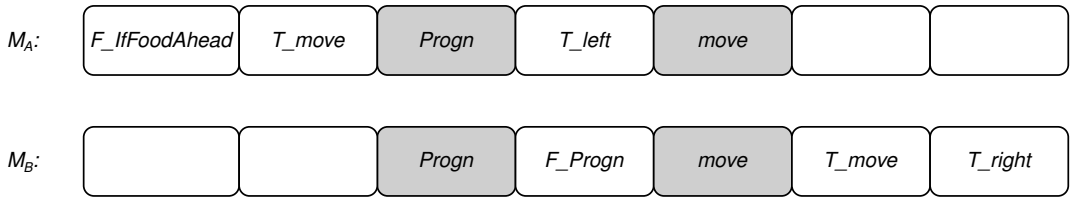
$$r_t = A_t - d \tag{3.5}$$

where A_f, A_t are respectively the number of function and terminal symbols of A which are not in B and B_f, B_t are respectively the number of function and terminal symbols of B which are not in A.

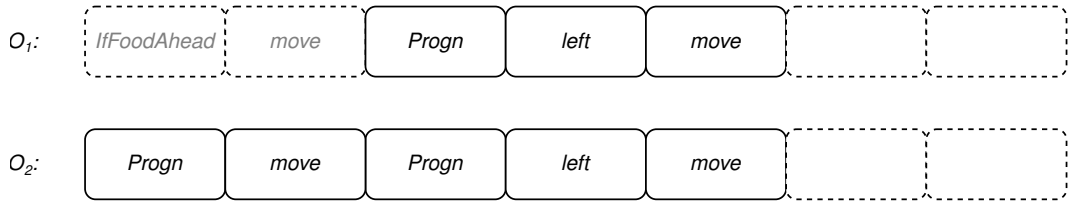
As an example, consider the Santa Fe Ant trail benchmark problem [Koza, 1992], for which the terminal set is {*left*, *right*, *move*} and the function set is {*IfFoodAhead*, *Progn*}. Two possible individuals are:



The Longest Common Subsequence between the two individuals is [*Progn, move*], and the modification masks created by GSynGP are:



The modification masks contain three types of symbols: (1) the aligned symbols that constitute the longest common subsequence (i.e., the aligned common symbols, whose nodes are presented with a grey background); (2) blank spaces, with those of M_A and M_B respectively marking where insertions or deletions must be made in A; and (3) the non-common symbols, marked with a $T\_$ or $F\_$ depending on whether they belong to the terminal or function sets, and that should either be deleted or inserted. The crossover operator uses these masks to make a copy of parent A more similar to parent B. This process can be repeated for various iterations, generating individuals at different points in the paths linking the original parents. Two possible offspring for the first iteration of this crossover operator are:



where O_1 results from deleting *IfFoodAhead* and *move*, whereas O_2 is created by deleting *IfFoodAhead* and inserting *Progn* in its place.

Extended GSynGP

The original version of Geometric Syntactic Genetic Programming considered syntax trees where each node contains a single symbol. While this is enough for problems such as the Santa Fe Ant Trail, even the simplest robotic actions (e.g., rotate and move) must be parametrised. While the parametrisation could be done via subtrees of different arities, a simpler approach is to have a behaviour and its parameters encapsulated in a single node. As a result, an extension to GSynGP was proposed [Macedo et al., 2020], where the symbols in the function and terminal sets, called main symbols, take a list of parameters. The proposed variant of the crossover operator works in the same manner as before when two nodes (one of each type) are to be removed or inserted. The novelty is when a node is to be deleted and another of the same type is to be inserted. For the sake of clarity, the node to be deleted shall be referred to as N_d, whereas the node to be inserted shall be called N_i. The new crossover operator works as follows:

1. A new node N_n is created with the main symbol of N_i;
2. The parameters that are present only in N_d are ignored, whereas those that only exist in N_i are added to the new node.

3. The parameters that are common to N_d and N_i are merged as follows: if a parameter takes a numerical value, it takes the mean value from the parents; otherwise, k randomly chosen parameters take the value from N_d, while the remaining take the value from N_i. In this work k was set to 1.

As an example consider the following two nodes, whose symbols are contained in the terminal set used in the experiments of the following chapters:

- N_d: $moveUpwind(d = 1.0)$
- N_i: $moveTowards(n_t = so,\ d = 1.0,\ r = 5.0)$

Further consider that during a crossover operation, N_d is to be deleted and N_i is to be inserted. The creation process of the new node N_n is depicted in Figure 3.1. As before, N_n is created with the symbol *moveUpwind*. The parameters n_t, and r from N_d are not present in N_i and thus are ignored. The parameter d is present in both nodes and takes a numerical value, so the mean of the values in both parents is used in N_n. N_n is then inserted in the appropriate place using the same method as in the original version of the crossover operator. The proposed extension to GSynGP enables it to perform smaller, geometric modifications to the individuals, rather than simply replacing the different nodes as a whole.

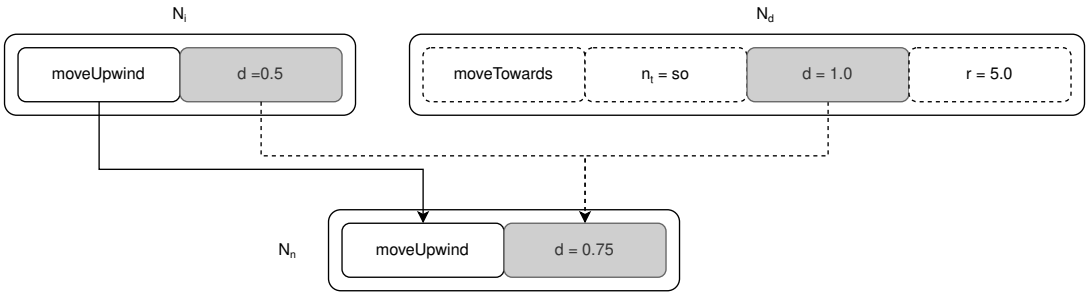


Figure 3.1: Creation of a new node, merging the parameters of its parent nodes.

3.2 Validation

The proposed genetic programming algorithm (GSynGP) will be compared with a standard version (SGP) to assess its ability to evolve high quality solutions (fitness), to maintain population diversity (both at the genotypic and semantic levels) and to control bloat (measured both through the size of the evolved individuals), in benchmark problems from two application domains.

3.2.1 Experimental setup

3.2.1.1 Algorithm parameters

The two GP algorithms to be compared share most operators, differing only in the crossover. As a result, they also share most parameters (Table 3.1), with GSynGP having an additional one to control the amount of iterations of its crossover operator. Both algorithms create the initial populations with the ramped

Table 3.1: Common parameters of the evolutionary algorithms

Parameter	Value
Population size	500
Generations	1000
Crossover probability	0.7
Mutation probability	0.3
Elite size	3
Tournament size	2
Maximum depth of initial trees	6
Maximum tree depth	17
Number of independent trials	30

half-and-half method. On each generation, a set of mates is chosen through tournament selection and recombined through either GSynGP's crossover or subtree crossover.

GSynGP's crossover was already described. In turn, the subtree crossover starts by selecting a cut node from the first parent, biasing the selection 90 % towards inner nodes and 10 % towards leaf nodes. The cut node from the second parent is chosen using the same bias, but restricting the search to nodes whose rooted subtree would not cause the offspring to be larger than the predefined limit. A single offspring is created by replacing the subtree selected from the first parent by the one chosen from the second parent.

The offspring may then be subjected to node mutation. The mutation operator starts by randomly selecting a node from the tree, with no regard to whether it is a function or a terminal. If that node requires parameters, then it is decided, with equal probability, whether its symbol or its parameters are to be mutated. If the choice is to replace the symbol, or if the node takes no parameters, its symbol is replaced by a randomly selected one from the respective set. In such case, the parameters that become obsolete are discarded and the parameters that were not required by the previous symbol, but are required by the new one, are initialised with randomly selected values. Otherwise, if the parameters are to be mutated, one of the node's parameters is randomly selected. If the chosen parameter takes a numerical value, it is equally likely that a new value is sampled uniformly from the corresponding set or that it is subjected to Gaussian mutation, with each σ being equal to 10 % of the respective domain width. Otherwise, the value of the parameter is replaced by another, sampled uniformly from the set of possible values. A new population is finally built through elitist selection and the algorithm carries on to a new generation.

In order to assess the influence of the iterations of the GSynGP's crossover, three variants are created: (1) using only one iteration (G_1); creating individuals midway between both parents (G_H); and creating offspring at random distances ranging from one iteration to half of the distance between both parents (G_R). Also, in preliminary testing, it was concluded that the injection of 35 random immigrants and 15 elitist immigrants (i.e., respectively 7 % and 3 % of the pop-

ulation size) into the population at each generation aided both algorithms in producing better results, and thus the experiments herein use that configuration.

3.2.1.2 Evaluation problems

Selecting proper benchmark problems has been highlighted as an issue of great importance to enable the reproducibility and comparison of different methods [McDermott et al., 2012]. In his book, Koza demonstrated the feasibility of GP through problems of various application domains, namely: optimal control, path planning, symbolic regression and boolean multiplexer [Koza, 1992]. The path planning problem used by Koza is the most related to this thesis and thus we shall use it for testing GSynGP. It consists on evolving the controller for an artificial ant that must explore an environment and collect food laid in an irregular trail. Symbolic regression, on the other hand, consists on evolving mathematical expressions that approximate a set of data and is one of the most commonly used type of problems for testing GP algorithms [McDermott et al., 2012].

The selected problems have been considered to be suitable for benchmarking GP methods in [McDermott et al., 2012]. Later, a community-made survey [White et al., 2013] included the simplest problems in a set deemed to be too easy, yet without providing any specific reasons. As such, and considering that they have continued to be used in more recent works [Boudardara and Gorkemli, 2018], we opted by including them.

Symbolic Regression

The symbolic regression validation shall be based on four benchmark problems [McDermott et al., 2012]: (1) Koza1 (K_1, Equation 3.6), also known as quartic; (2) Koza3 (K_3, Equation 3.7); (3) Paige1 (P_1, Equation 3.8); and (4) Keijzer12 (K_{12}, Equation 3.9). While quartic has been considered to be too easy to provide meaningful results, it is often used and so we opt by including it as well. In turn, Paige1 is considered to be quite difficult [White et al., 2013]. These functions are plotted in Figure 3.2 and their formulas are as follows:

$$K_1(x) = x^4 + x^3 + x^2 + x \tag{3.6}$$

$$K_3(x) = x^6 - 2x^4 + x^2 \tag{3.7}$$

$$P_1(x, y) = \frac{1}{1 + x^{-4}} + \frac{1}{1 + y^{-4}} \tag{3.8}$$

$$K_{12}(x, y) = x^4 - x^3 + \frac{y^2}{2} - y \tag{3.9}$$

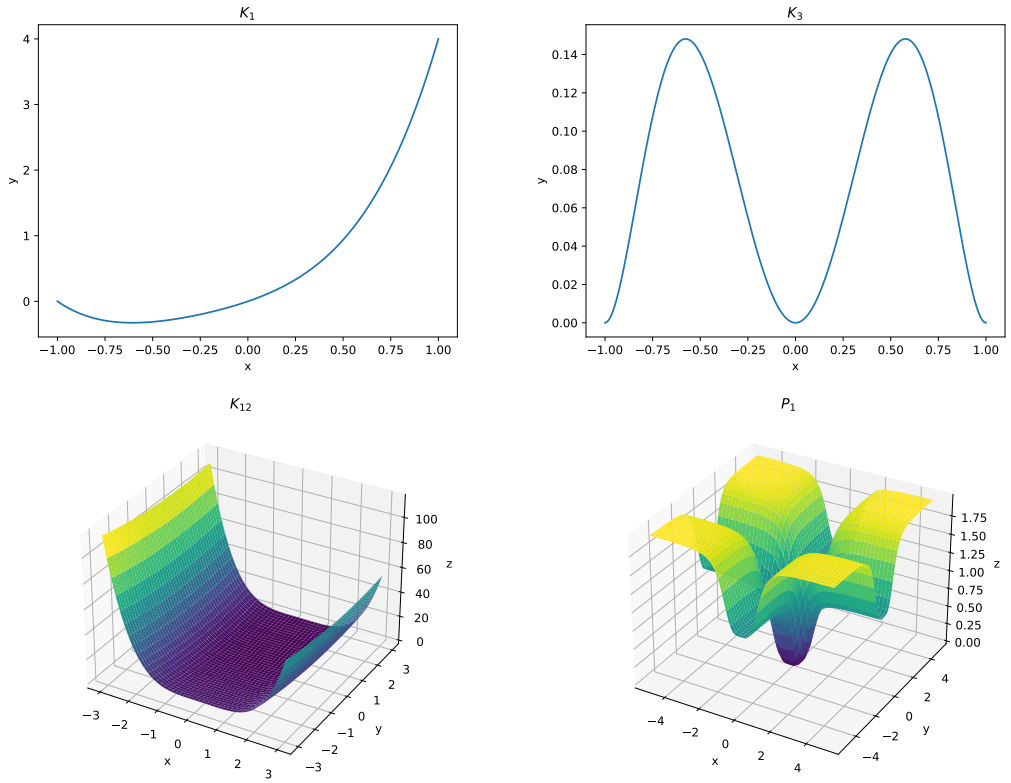


Figure 3.2: Symbolic regression benchmark problems: koza1 (top-left), koza3 (top-right), keijzer12 (bottom-left) and paige1 (bottom-right).

The original works using functions K_1 and K_3 create the datasets with 20 points uniformly drawn from the domain [-1,1], while for K_{12} 20 points are drawn from [-3,3]. In turn, for P_1 it was originally proposed to create a mesh based on its two input variables, each taking a set of points spaced by 0.4 and ranging from -5 to 5. However this creates a grid of 676 points, which is much larger than the datasets used by the previous functions. As such, we opt by also using 20 points in P_1, sampled uniformly from the interval [-5,5]. The datasets are created at the beginning of each trial and used for all evaluations, thus ensuring fair comparisons between individuals of the same trial and different datasets for different trials. Using these datasets, the fitness of each individual is measured by computing the Mean Squared Error (MSE), as follows:

$$MSE = \frac{\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}{N} \tag{3.10}$$

where N is the number of samples, Y_i is the ith target value and $\hat{Y}_i$ is the ith predicted value.

To solve these problems, the two algorithms use the same terminal and function sets, which were selected according to [McDermott et al., 2012]:

- Function set:
  - $+$ - adds the values of the two subtrees.

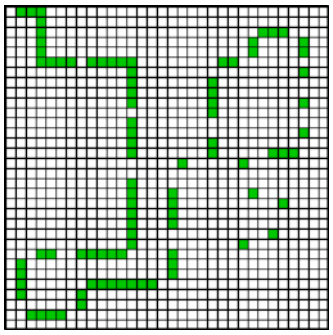


Figure 3.3: Santa Fe Ant Trail.

  - $-$ - subtracts the value of the right subtree to the left subtree.
  - $*$ - multiplies the values of the two subtrees.
  - % - if the value of the right subtree is 0, this function returns 1. Otherwise, it returns the division of the left subtree by the right subtree.
  - *sin* - returns the value of the sine of the left subtree.
  - *cosine* - returns the value of the cosine of the left subtree.
  - *exp* - returns e^n, where n is the value of the left subtree.
  - *lnmod* - returns $ln(|n|)$, where n is the value of the left subtree.
- Terminal set:
  - $x_1, ..., x_n$ - problem variables
  - ephemeral constant - each time this symbol is selected for creating an individual, a randomly sampled constant from the interval [-1, 1] takes its place.

Artificial ant

The artificial ant is a path planning benchmark problem that has been traditionally used to test GP algorithms [Koza, 1992, Koza, 1994]. It consists of a 2D toroidal grid world where the majority of the cells are empty and some contain food pellets. The task is to produce the controller for an artificial ant with the goal of collecting the maximum amount of food pellets within the available time limit. The ant collects a food pellet simply by visiting the cell containing it. Three actions are typically used: *left*, *right* and *move*. The first two actions consist on rotating 90° in the respective direction. The third action consists on moving one cell in the direction that the ant is facing.

The artificial ant problem has been proposed in a number of variants, namely the Santa Fe Trail (Figure 3.3) and the Los Altos Hills Trail (Figure 3.4), which differ in the size of the world, amount of food available and overall difficulty.

Both problems are tackled with the same terminal and function sets:

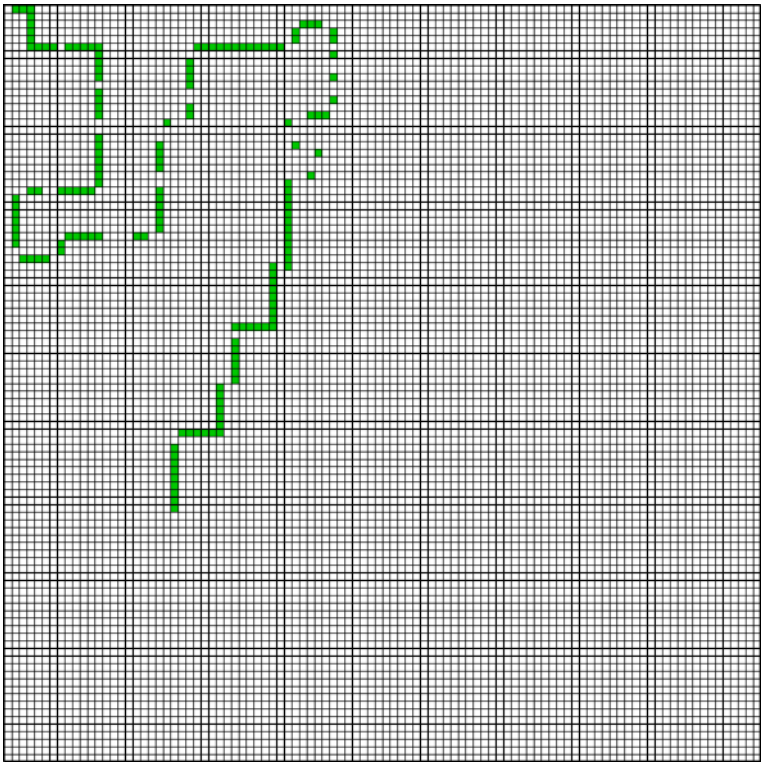


Figure 3.4: Los Altos Hills Trail.

Table 3.2: Parameters for the artificial ant problems

Parameter	Santa Fe Trail	Los Altos Hills Trail
Grid size	32x32	100x100
Food pellets	89	157
Time steps	600	3000

- Function set:
  - *Progn* - executes both subtrees in sequence.
  - *IfFoodAhead* - returns true if there is a food pellet in the cell ahead and false otherwise.
- Terminal set:
  - *left* - rotates the ant 90° to the left.
  - *right* - rotates the ant 90° to the right.
  - *move* - moves the ant one cell to the front.

The fitness of the individuals is typically measured by the number of food pellets eaten, making these maximisation problems. However, to improve readability, we convert them into minimisation problems by computing the fitness of the individuals as the number of food pellets left not eaten, as follows:

$$F_A = P_I - P_E \tag{3.11}$$

where P_I is the initial amount of food pellets and P_E is the amount of food pellets eaten.

3.2.1.3 Statistical analysis: general approach

The results of the experiments in this thesis will be presented through tables and plots, enabling their visual comparison and discussion. According to the central limit theorem, a sufficiently high number of independent trials must be run for each approach. In this work, we use the commonly adopted value of 30 runs. Moreover, to be able to state that two (or more) approaches perform significantly differently (according to a given metric), we must employ statistical hypothesis tests. In order to choose an appropriate test, three questions must be answered:

1. How many data groups are being compared?
2. Do the experiments use the same initial conditions?
3. Can parametric tests be applied?

The first two questions are related to the way the experiments are made. The number of data groups depends on the analysis being made but, in the scope of this chapter, it will typically be four (i.e., SGP and the three GSynGP variants). The second question is concerned with whether the various approaches have the same start conditions. If possible, the approaches being tested should have the same initial conditions to remove one source of variability. In this work, the initial conditions are guaranteed to be matched by setting the seed of the pseudo random generator for each trial, i.e., thirty seeds are used (one for each trial) and shared among the various approaches. Finally, to answer the third question, one must verify that two assumptions for applying parametric tests are met:

1. The data of the various approaches follow normal distributions;
2. Their variances are homogeneous;

Thus, the first step consists on assessing the normality of the data of each approach, which we shall do through the Kolmogorov-Smirnov test, using the following null (H0) and alternative (H1) hypothesis:

- H0: The data follows a normal distribution;
- H1: The data does not follow a normal distribution;

If the p-value outputted by the test is below the significance value (which in this work we set at 0.05 to have a 95 % confidence on the conclusions drawn), the null hypothesis can be rejected and non-parametric tests must be used for the remaining analysis. Otherwise, the Levene test is used to assess the homogeneity of the variances, i.e., whether the data from the various approaches can be considered to have equal variances. This test is applied with the following null (H0) and alternative (H1) hypothesis:

- H0: The data sets have equal variances;
- H1: The data sets do not have equal variances.

If the p-value resulting from this test is below the significance value, then the null hypothesis can be rejected and non-parametric test must be used. Otherwise, parametric tests may be used in the remaining analysis.

The next step depends on the number of approaches (or variants) are being compared. If there are more than two, than a group test must be applied to assess whether there are statistically significant differences between the data of the approaches. Depending on whether the assumptions for using parametric tests are met, the Dependent Anova (parametric) or the Friedman's Anova (non-parametric) is applied with the following null (H0) and alternative (H1) hypothesis:

- H0: All data samples are drawn from the same population;
- H1: There is at least one sample that is drawn from a different population.

If this test yields a p-value below the significance value, the null hypothesis can be rejected and we proceed to perform pairwise comparisons to assess which approaches perform differently. By making multiple comparisons, the chance of making type I errors increases. As a result, the Bonferroni correction is applied, consisting of dividing the significance value by the number of comparisons made. Depending on whether parametric tests may be used, the Dependent t-test (parametric) or the Wilcoxon test (non-parametric) is applied with the following null (H0) and alternative (H1) hypothesis:

- H0: The two data samples are drawn from the same population;
- H1: The two data samples are drawn from different populations;

If the test yields a p-value below the adjusted significance value, the null hypothesis may be rejected, i.e., the approaches can be considered to perform

significantly differently.

3.2.2 Experimental results

This section presents the experimental results attained in the previously described benchmark problems. The quality and size of the evolved solutions is analysed, as well as the algorithms' abilities to maintain population diversity, both at the genotypic and behavioural levels.

3.2.2.1 Fitness of the evolved solutions

We start by analysing the fitness of the resulting solutions. Each run of an EA produces one solution (the best of the final population), yielding 30 solutions per algorithm (one for each independent trial). We assess the fitness of the controllers attained during evolution (train) and also in a previously unseen set of data (test).

Train results

The fitness of the evolved controllers for each benchmark problems are plotted in Figure 3.5. In the symbolic regression problems, the boxplots indicate that SGP produces the most fit solutions (lower values), followed by G_1. Moreover, as the number of GSynGP's iterations increases, the fitness of the evolved solutions tends to worsen. Interestingly, the same trend is not followed in the ant problems, where SGP produces similar values to G_1 but, in the SFT, increasing the number of iterations leads to better fitness values.

In order to draw statistically supported conclusions, we apply statistical hypothesis tests, following the methodology described in Section 3.2.1.3. We start by assessing the normality of the data with the Kolmogorov-Smirnov test, whose results are presented on Table 3.3. With p-values below 0.05, the results of this test show that none of the data can be considered to follow normal distributions and thus we must resort to non-parametric tests. As such, we apply the Friedman's Anova, in order to assess whether there are statistically significant differences in the results of the four approaches. The results of the Friedman's Anova (Table 3.4) show that there are statistically significant differences between the four approaches for all benchmark problems apart from the Los Altos Hills Trail. As a result, in the cases where statistically significant differences were found, we must perform pairwise comparisons between the data of the four approaches to assess which are indeed significantly different. To do so, we apply the Wilcoxon test and to account for errors of continuity, we apply the Bonferronni correction, reducing the significance value to 8.33e-03. The results of the Wilcoxon test are presented on Table 3.5 and show that G_1 produces equivalent fitness values to SGP in K_1 and SFT, performing worse than SGP in K_3, P_1 and K_{12}. The Wilcoxon test also verifies that increasing the number of GSynGP iterations often leads to worse fitness values in the symbolic regression problems, particularly in K_1, and K_{12}. Interestingly, when considering the Santa Fe Ant Trail, the results are reversed. SGP achieves equivalent results to G_1 and G_R, but is significantly

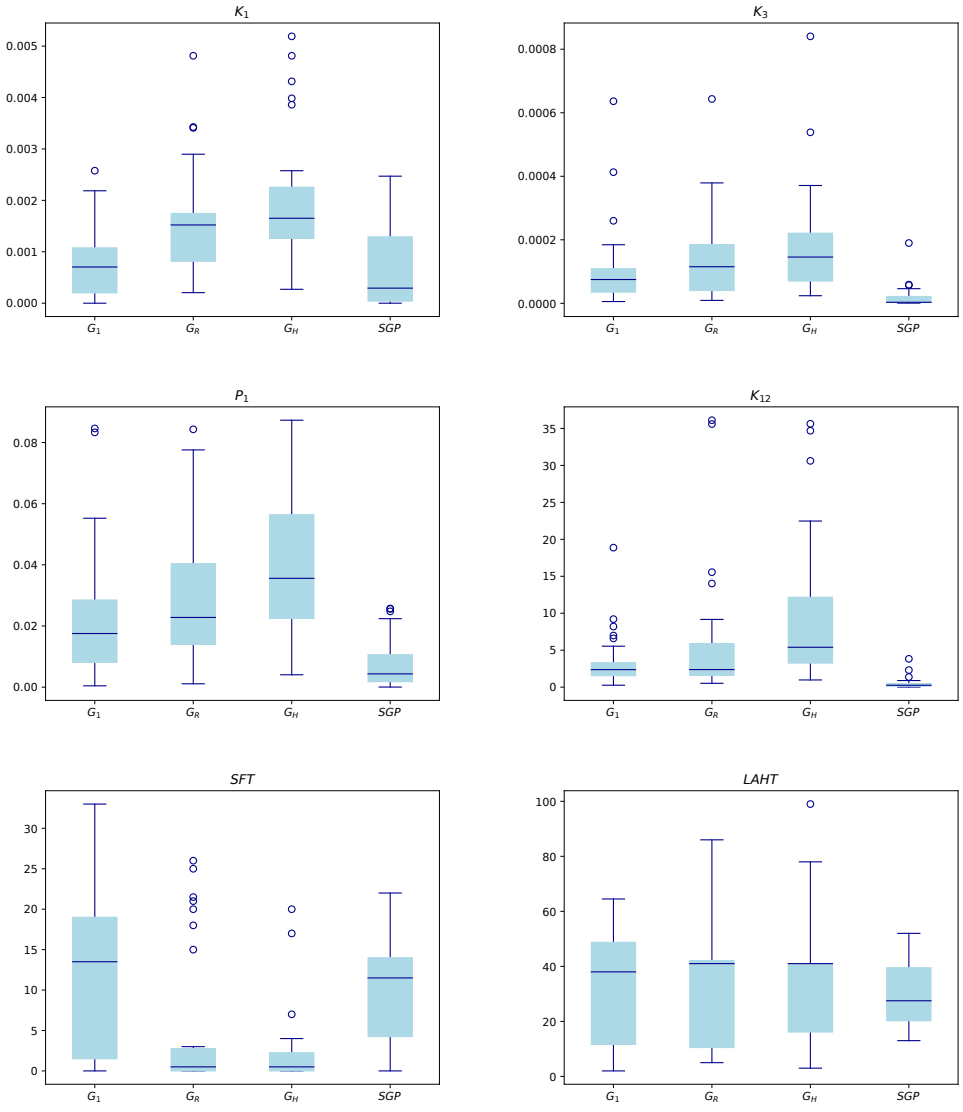


Figure 3.5: Fitness of the best individuals at the end of evolution for Koza1 (top-left), Koza3 (top-right), Paige1 (centre-left), Keijzer12 (centre-right), Santa Fe Trail (bottom-left) and the Los Altos Hills Trail (bottom-right).

Table 3.3: P-values of the Kolmogorov-Smirnov test applied to the fitness in train of the best evolved individuals.

	G_1	G_R	G_H	SGP
K_1	2.03e-06	5.52e-06	4.47e-06	1.70e-08
K_3	1.48e-05	2.45e-05	1.18e-05	2.81e-11
P_1	6.33e-06	1.12e-06	2.01e-06	1.36e-09
K_{12}	5.51e-08	1.19e-10	1.53e-08	2.56e-09
SFT	8.91e-05	6.70e-16	6.17e-12	8.71e-07
$LAHT$	2.04e-06	1.12e-07	1.00e-10	1.67e-05

Table 3.4: P-values of the Friedman's Anova applied to the fitness in train of the best evolved individuals.

K_1	K_3	P_1	K_{12}	SFT	$LAHT$
6.44e-06	5.48e-09	1.20e-08	1.77e-12	1.63e-04	**9.97e-01**

inferior to G_H. Moreover, G_1 performs significantly worse than G_R and G_H, but there are no significant differences between G_R and G_H.

Test results

This section analyses the ability of the evolved solutions for SR tasks to generalise to unseen data. While for evolving the individuals, the datasets are created by randomly sampling 20 points from the domain, in testing, a comprehensive dataset is generated for each problem. Such datasets encompass 100 points equally spaced between the lower and upper bounds of the domain. In case of multi-dimensional functions (Paige1 and Keijzer12), a mesh is created with each variable abiding by the previous method.

The boxplots of the fitness values attained by the evolved solutions with these new datasets are presented on Figure 3.6. Overall, all approaches are plagued by outliers (left column) and, after removing the outliers (right-column), it seems that all approaches perform similarly.

Table 3.5: P-values of the Wilcoxon test applied to the fitness in train of the best evolved individuals.

	G_1 - SGP	G_R - SGP	G_H - SGP	G_1 - G_R	G_1 - G_H	G_R - G_H
K_1	**4.65e-01**	4.20e-04	2.37e-05	5.47e-03	1.04e-04	**5.46e-02**
K_3	5.79e-05	2.16e-05	3.88e-06	**1.31e-01**	**3.16e-02**	**2.13e-01**
P_1	6.16e-04	4.73e-06	2.60e-06	**1.11e-01**	**9.27e-03**	**1.16e-01**
K_{12}	1.80e-05	1.02e-05	2.88e-06	**6.87e-02**	6.16e-04	**9.84e-03**
SFT	**1.94e-01**	**4.22e-02**	2.91e-04	5.33e-03	3.75e-04	**1.88e-01**
$LAHT$	-	-	-	-	-	-

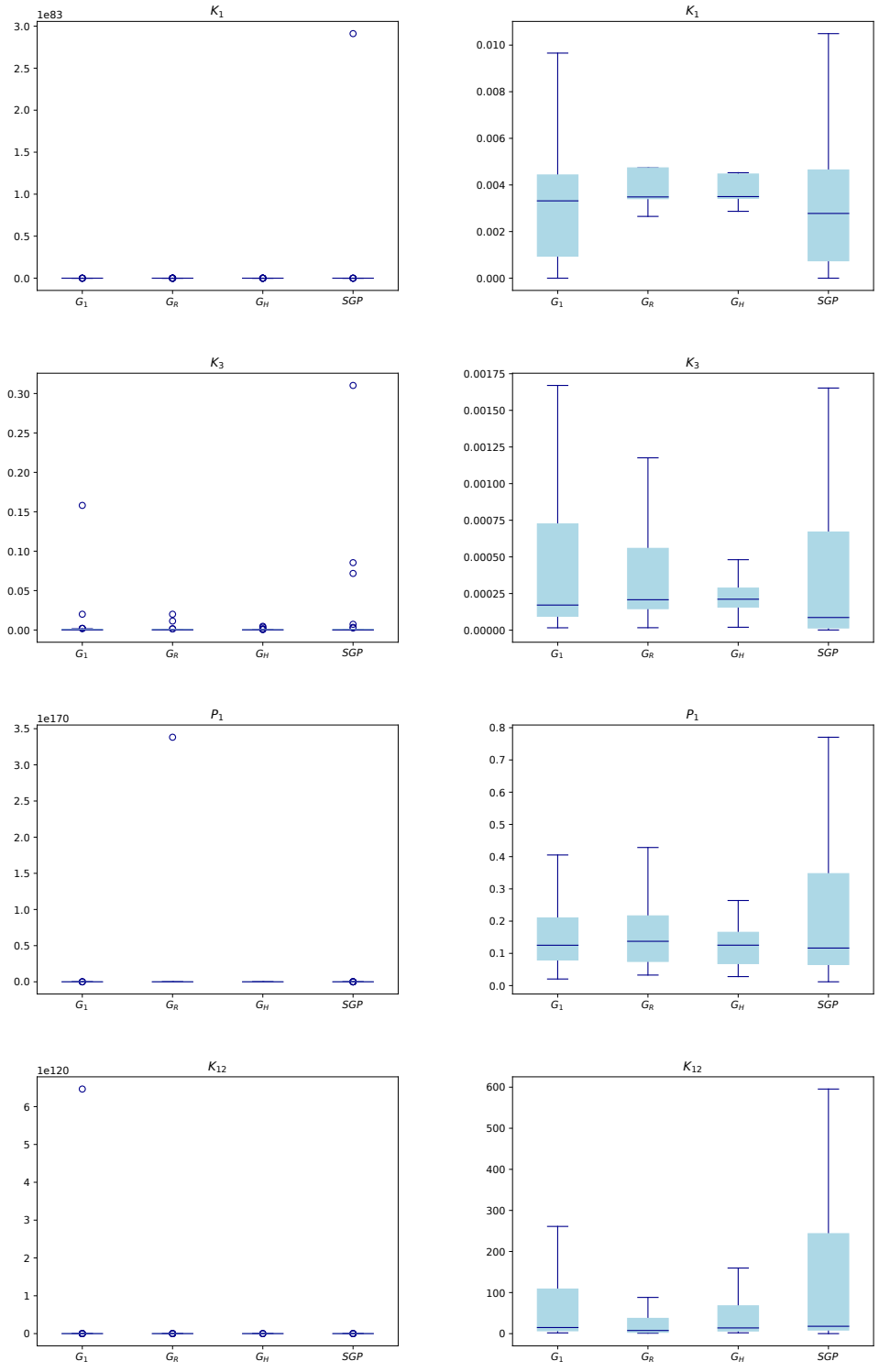


Figure 3.6: From the top to the bottom it is presented the original (left) without outliers (right) boxplots of the fitness of the best individuals in testing for Koza1, Koza3, Paige1 and Keijzer12.

Table 3.6: P-values of the Kolmogorov-Smirnov test applied to the fitness in test of the best evolved individuals.

	G_1	G_R	G_H	SGP
K_1	4.92e-22	1.27e-12	2.97e-17	4.92e-22
K_3	6.67e-17	1.55e-12	1.23e-14	3.83e-18
P_1	4.92e-22	1.85e-07	4.72e-05	3.55e-26
K_{12}	4.92e-22	1.58e-12	2.20e-08	4.92e-22

Table 3.7: P-values of the Friedman's Anova applied to the fitness in test of the best evolved individuals.

K_1	K_3	P_1	K_{12}
5.16e-01	**2.75e-01**	**6.15e-01**	**8.00e-02**

Once again, a statistical analysis is carried out to draw statistically supported conclusions. Starting with the Kolmogorov-Smirnov test (Table 3.6), its results show that we may reject the null hypothesis that the data follow normal distributions and resort to non-parametric tests. We then proceed to apply the Friedman's Anova (Table 3.7) which shows that there are no statistically significant differences between the performance of the four approaches in any of the benchmark problems, i.e., the solutions produced by the four approaches may be considered to generalise equally well to previously unseen data.

Best solutions

To conclude the analysis of the fitness of the evolved solutions, we now focus on the overall best individual evolved by each approach for each benchmark problem. These individuals are selected as being those which attain the best fitness value. The validation datasets are once again used to produce their behaviour, which is plotted along with the target function on Figures 3.7 (K_1), 3.8 (K_3), 3.9 (P_1) and 3.10 (K_{12}). As can be seen, in K_1, G_R and G_H exhibit a slight mismatch with the target function. In K_3, all individuals fail to exactly match the target function, but are still reasonably close. The same cannot be said for P_1, where all approaches struggle to match the target. In turn, in K_{12} all algorithms match the target quite well, with G_H exhibiting a slight shift.

3.2.2.2 Size of the evolved solutions

This section analyses the average size (measured in number of nodes) of the individuals in the last population of each approach, for each benchmark problem. This is an important characteristic, as smaller controllers are not only more efficient, but also more interpretable and one of the goals of this thesis is precisely the evolution of interpretable controllers. The boxplots presented on Figure 3.11 show that SGP produces individuals much larger than those of GSynGP. In fact, the median size of SGP's individuals is consistently well over 200 nodes (in

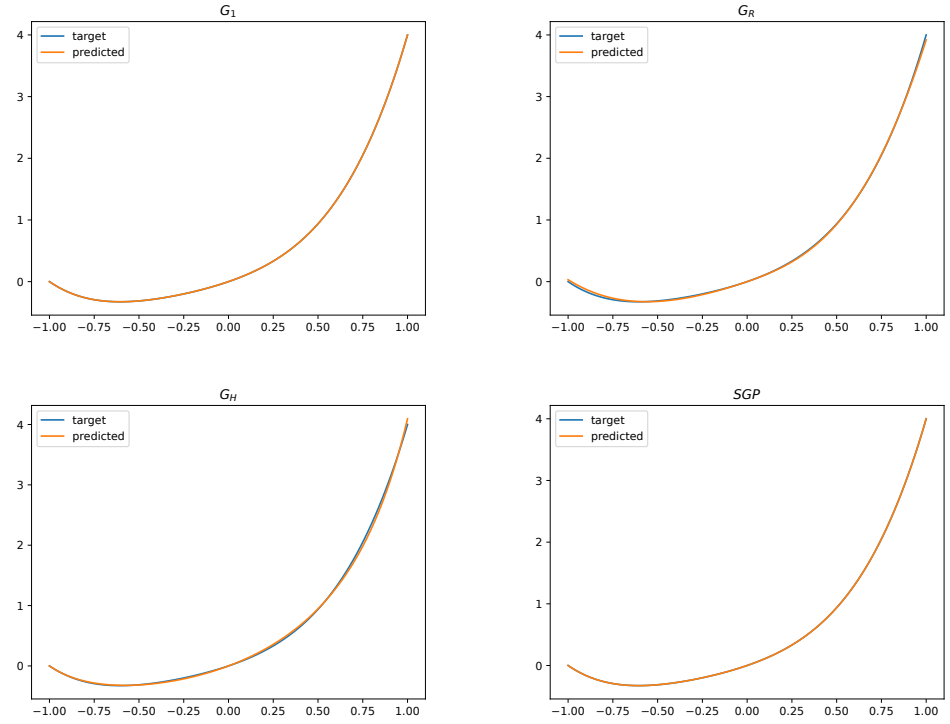


Figure 3.7: Best evolved individuals for Koza1.

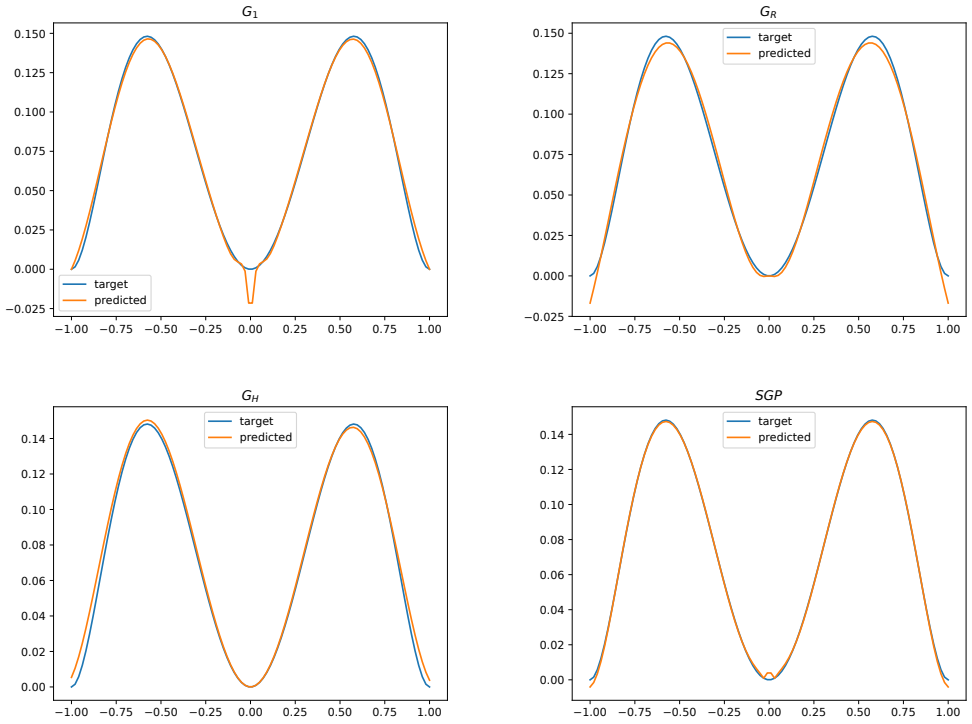


Figure 3.8: Best evolved individuals for Koza3.

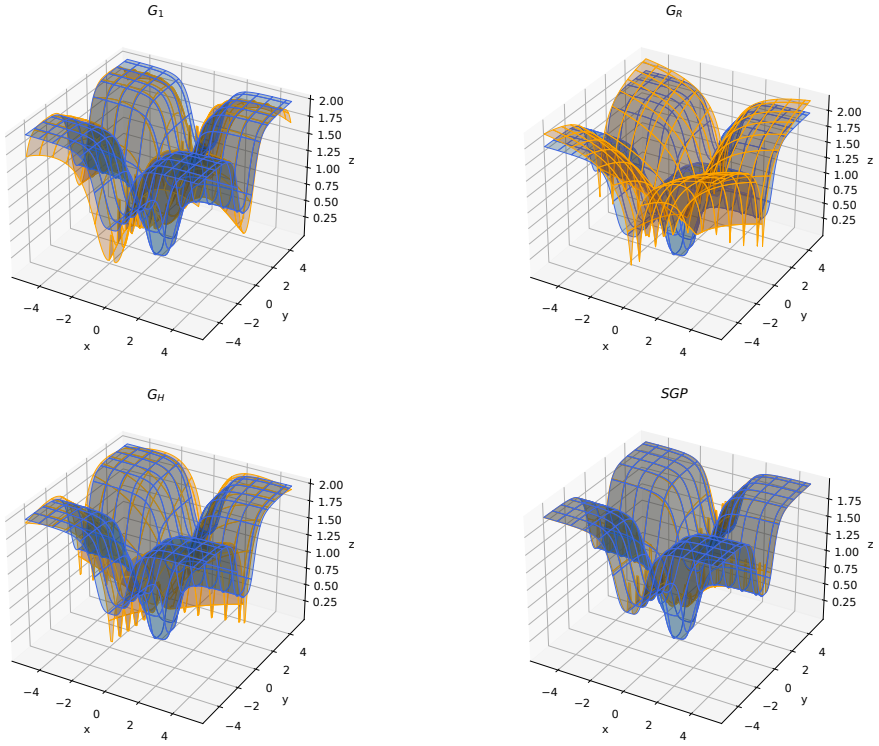


Figure 3.9: Best evolved individuals for Paige1.

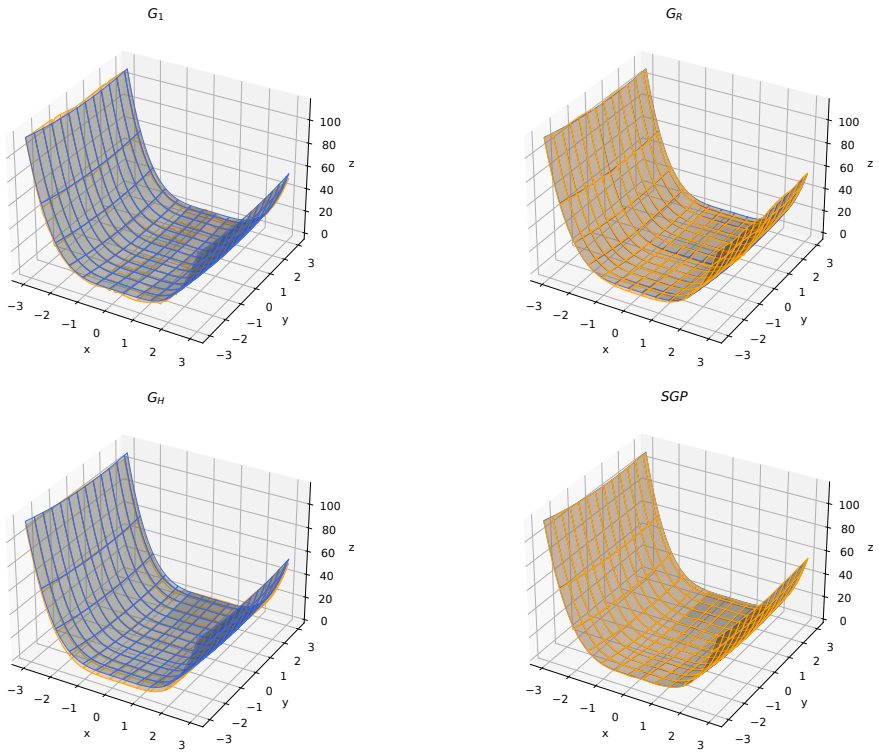


Figure 3.10: Best evolved individuals for Keijzer12.

Table 3.8: P-values of the Kolmogorov-Smirnov test applied to the mean size of the individuals in the final populations of each algorithm.

	G_1	G_R	G_H	SGP
K_1	4.48e-06	8.34e-05	3.50e-05	9.30e-06
K_3	8.69e-05	7.42e-14	2.45e-04	4.09e-04
P_1	1.00e-04	5.54e-09	8.41e-09	8.15e-06
K_{12}	3.39e-04	1.71e-08	3.79e-07	1.90e-05
SFT	2.75e-04	2.21e-07	1.41e-04	8.00e-05
$LAHT$	3.14e-07	1.96e-05	2.21e-07	4.47e-05

Table 3.9: P-values of the Friedman's Anova applied to the mean size of the individuals in the final populations of each algorithm.

K_1	K_3	P_1	K_{12}	SFT	$LAHT$
7.11e-18	3.89e-17	1.30e-16	1.73e-17	2.60e-18	5.29e-18

some cases over 300 nodes), whereas those of G_1 are close to 100 nodes. It can also be seen that increasing the amount of GSynGP iterations leads to smaller individuals, with G_R and G_H producing much smaller controllers than G_1, with median sizes below 50 nodes.

Following the aforementioned methodology, the Kolmogorov-Smirnov test is once again applied to assess the normality of the data (Table 3.8). Its results show that all instances are below the significance value of 0.05, implying that none of the data can be considered to follow normal distributions and forcing us to resort to non-parametric tests. The Friedman's Anova is then applied to assess whether there are differences between the mean sizes of the evolved individuals for each benchmark problem. As can be seen in Table 3.9, there are statistically significant differences in all benchmark problems, and thus we proceed to perform pairwise comparisons with the Wilcoxon test. Once again, the Bonferroni correction is applied to adjust the significance value to 8.33e-03. The results of the Wilcoxon test are presented on Table 3.10 and show that almost all comparisons produce statistically significant differences, meaning that SGP produces the largest individuals and that increasing GSynGP's iterations leads to significantly smaller individuals. The sole exception is when comparing G_R and G_H in K_1, which produce individuals with equivalent sizes.

3.2.2.3 Number of iterations

In this section we investigate whether the reason for G_R and G_H seeming to perform more similarly than G_1 and G_R is related to their number of iterations being closer. Figure 3.12 presents boxplots of the mean GSynGP iterations performed at the final generation. As can be seen, G_H makes more iterations than G_R in all benchmark problems, with the gap being larger in K_1 and in the ant problems than in the other three. Also, it is interesting to note that the median iterations of G_R range between two and three, i.e., double and triple of

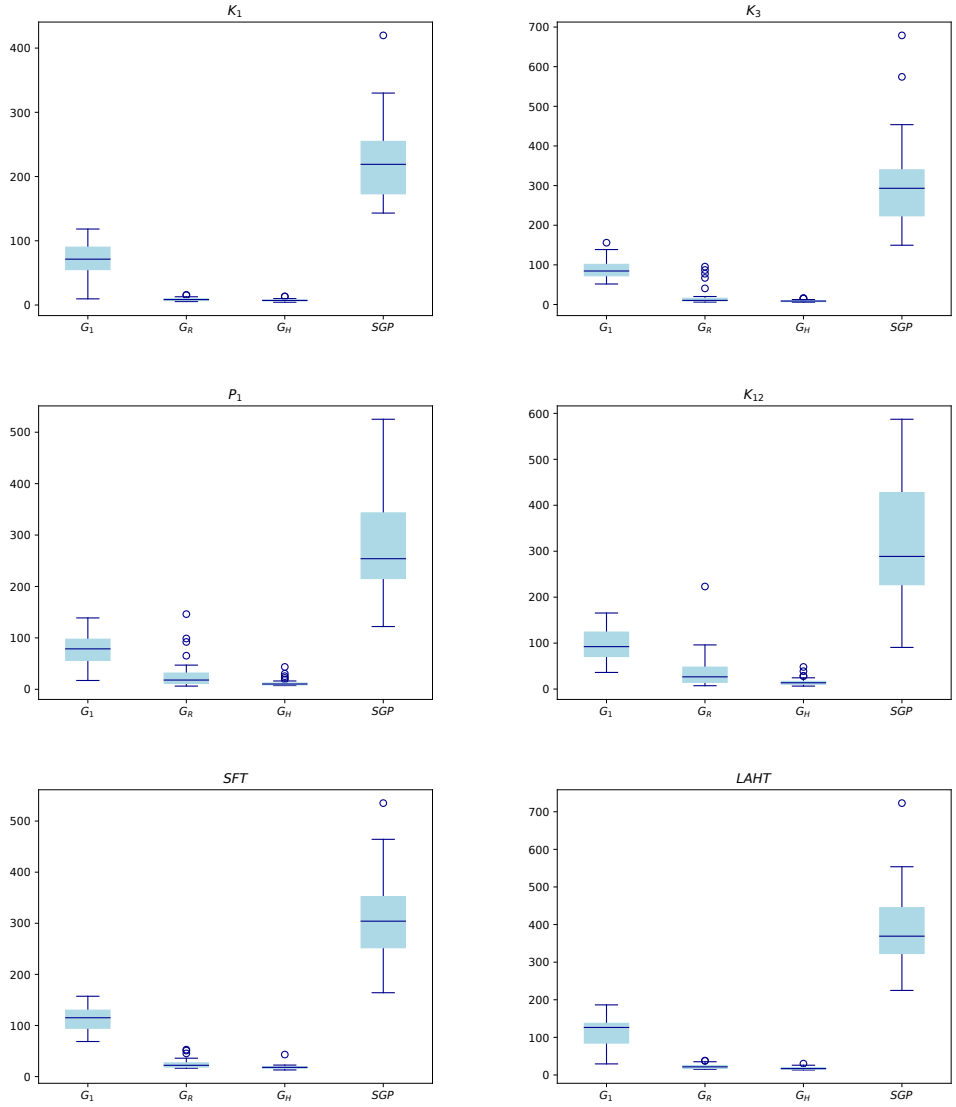


Figure 3.11: Mean size of the individuals in the last population for Koza1 (top-left), Koza3 (top-right), Paige1 (centre-left), Keijzer12 (centre-right), Santa Fe Trail (bottom-left) and the Los Altos Hills Trail (bottom-right).

Table 3.10: P-values of the Wilcoxon test applied to the mean size of the individuals in the final populations of each algorithm.

	G_1 - SGP	G_R - SGP	G_H - SGP	G_1 - G_R	G_1 - G_H	G_R - G_H
K_1	1.73e-06	1.73e-06	1.73e-06	1.73e-06	1.73e-06	**1.48e-02**
K_3	1.92e-06	1.73e-06	1.73e-06	2.88e-06	1.73e-06	7.27e-03
P_1	1.73e-06	1.73e-06	1.73e-06	2.84e-05	2.13e-06	3.16e-03
K_{12}	1.73e-06	1.73e-06	1.73e-06	5.79e-05	1.73e-06	2.83e-04
SFT	1.73e-06	1.73e-06	1.73e-06	1.73e-06	1.73e-06	1.60e-04
$LAHT$	1.73e-06	1.73e-06	1.73e-06	1.73e-06	1.73e-06	1.59e-03

Table 3.11: P-values of the Kolmogorov-Smirnov test applied to the mean normalised genotypic diversity of the final populations of each algorithm.

	G_1	G_R	G_H	SGP
K_1	1.98e-04	1.68e-04	2.80e-05	5.89e-06
K_3	3.22e-06	2.94e-09	2.78e-06	1.05e-06
P_1	1.68e-05	2.06e-05	3.89e-06	2.63e-06
K_{12}	1.74e-04	1.98e-06	1.51e-04	7.39e-05
SFT	2.13e-06	2.27e-06	3.53e-04	7.85e-04
$LAHT$	2.05e-04	1.65e-04	5.16e-06	1.83e-04

the iterations of G_1. In turn, the proportion between the iterations of G_H and G_R is always smaller, which may be the reason why these two variants perform more similar than G_1 and G_R.

3.2.2.4 Population diversity

We now assess the ability of each algorithm to maintain the population diversity over the evolution. This is an important feature not only for properly exploring the search space and maximising the chances of finding the global optima, but also for quickly re-adapting the population in case of dynamic environments. We assess the diversity of the algorithms both at the phenotypic and semantic (behavioural) level. Due to computational constraints, the diversities are measured by computing the average distance of all individuals to the best of the population, and we focus solely on the last population. This approach should still provide a good indicator, as it provides insight into whether the populations have converged to the best individual or whether they still contain a high degree of diversity, enabling the algorithm to continue exploring the search space.

Genotypic diversity

The genotypic distance from one individual to another consists on the number of its genes that are not contained in the Longest Common Subsequence between both individuals. As a result, the distance between two individuals is not necessarily symmetrical.

The mean genotypic diversity in the last population of each algorithm for each benchmark problem are plotted on Figure 3.13. At first glance it would seem that SGP is much better at maintaining the population diversity. However, these plots must be taken with a grain of salt, as the individuals evolved by SGP are much larger, and consequently much more prone to having higher quantities of differing genes. As a result, we normalize the distance between two individuals by their size and plot those results on Figure 3.14. As can be seen from this figure, once the genotypic diversity is normalised, SGP's ability to maintain diversity is similar to that of G_1, particularly in the symbolic regression problems. Interestingly, for GSynGP, the population diversity increases with the amount of iterations performed.

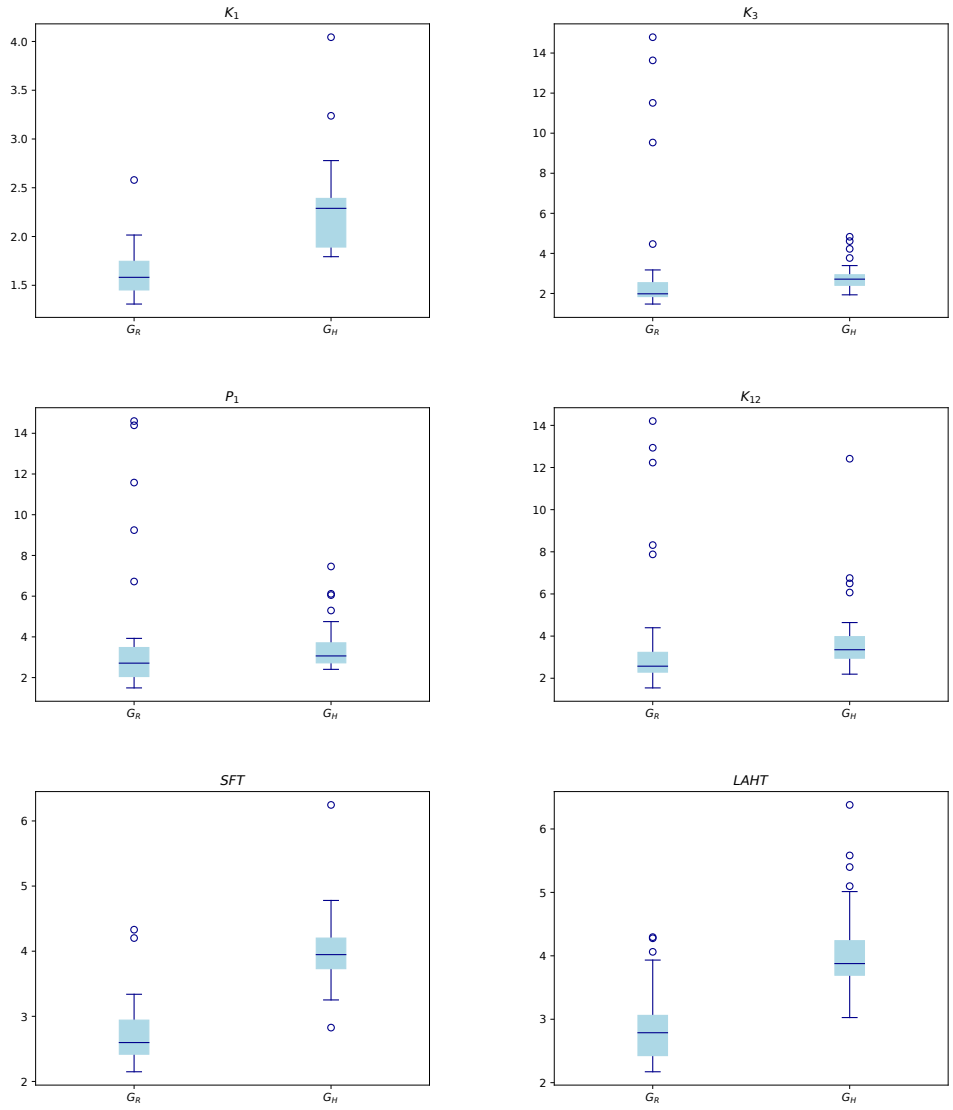


Figure 3.12: Mean number of GSynGP iterations for Koza1 (top-left), Koza3 (top-right), Paige1 (centre-left), Keijzer12 (centre-right), Santa Fe Trail (bottom-left) and the Los Altos Hills Trail (bottom-right).

Table 3.12: P-values of the Friedman's Anova applied to the mean normalised genotypic diversity of the final populations of each algorithm.

K_1	K_3	P_1	K_{12}	SFT	$LAHT$
1.62e-15	4.35e-15	2.42e-13	1.57e-11	5.34e-07	1.48e-13

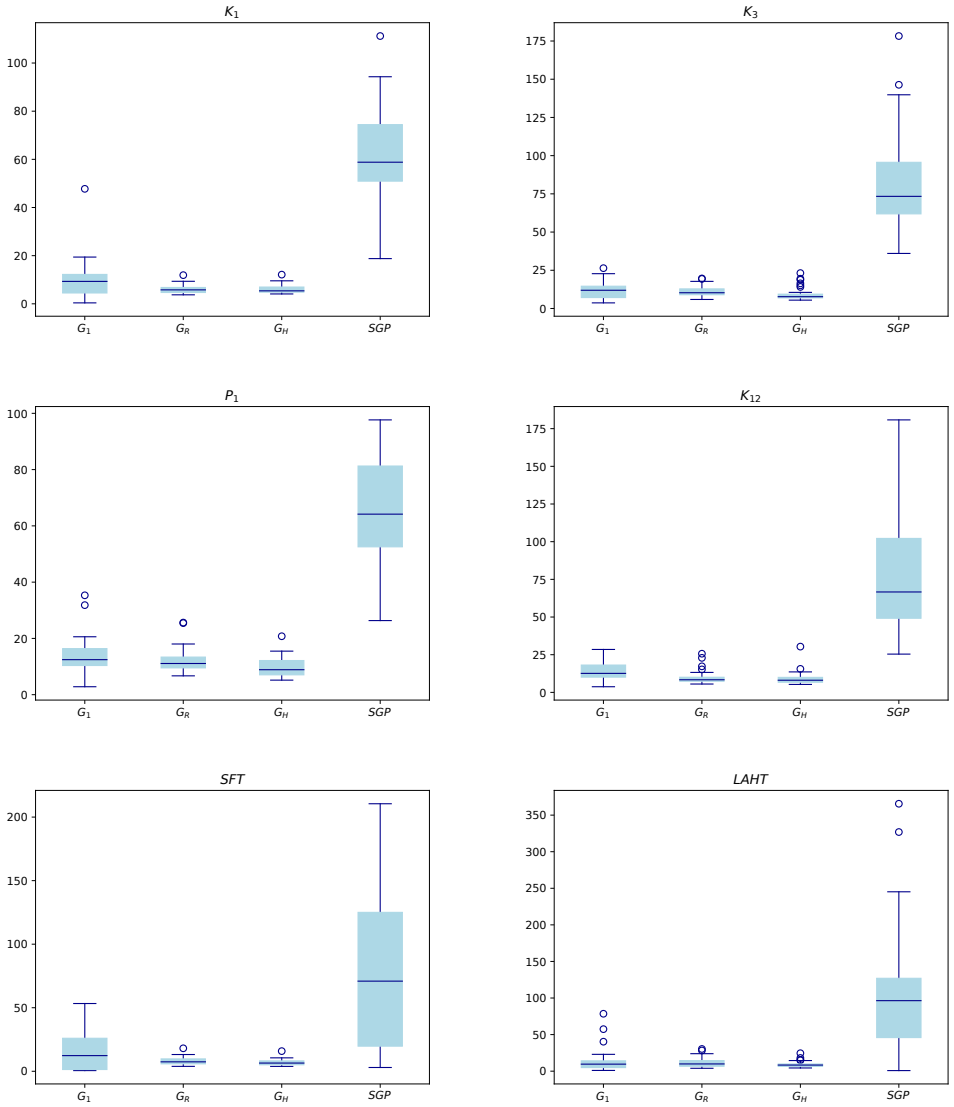


Figure 3.13: Mean genotypic distance for Koza1 (top-left), Koza3 (top-right), Paige1 (centre-left), Keijzer12 (centre-right), Santa Fe Trail (bottom-left) and the Los Altos Hills Trail (bottom-right).

Table 3.13: P-values of the Wilcoxon test applied to the mean normalised genotypic diversity of the final populations of each algorithm.

	G_1 - SGP	G_R - SGP	G_H - SGP	G_1 - G_R	G_1 - G_H	G_R - G_H
K_1	**5.72e-01**	1.73e-06	1.73e-06	1.73e-06	1.92e-06	2.96e-03
K_3	**4.91e-01**	2.13e-06	1.73e-06	1.73e-06	1.73e-06	**3.71e-01**
P_1	**5.72e-01**	3.88e-06	1.73e-06	1.49e-05	1.73e-06	**7.52e-02**
K_{12}	**8.77e-01**	5.29e-04	1.73e-06	1.60e-04	2.13e-06	4.39e-03
SFT	**1.85e-02**	**2.07e-02**	8.94e-04	3.41e-05	5.75e-06	4.90e-04
$LAHT$	3.59e-04	5.79e-05	1.36e-05	1.73e-06	1.73e-06	**5.45e-02**

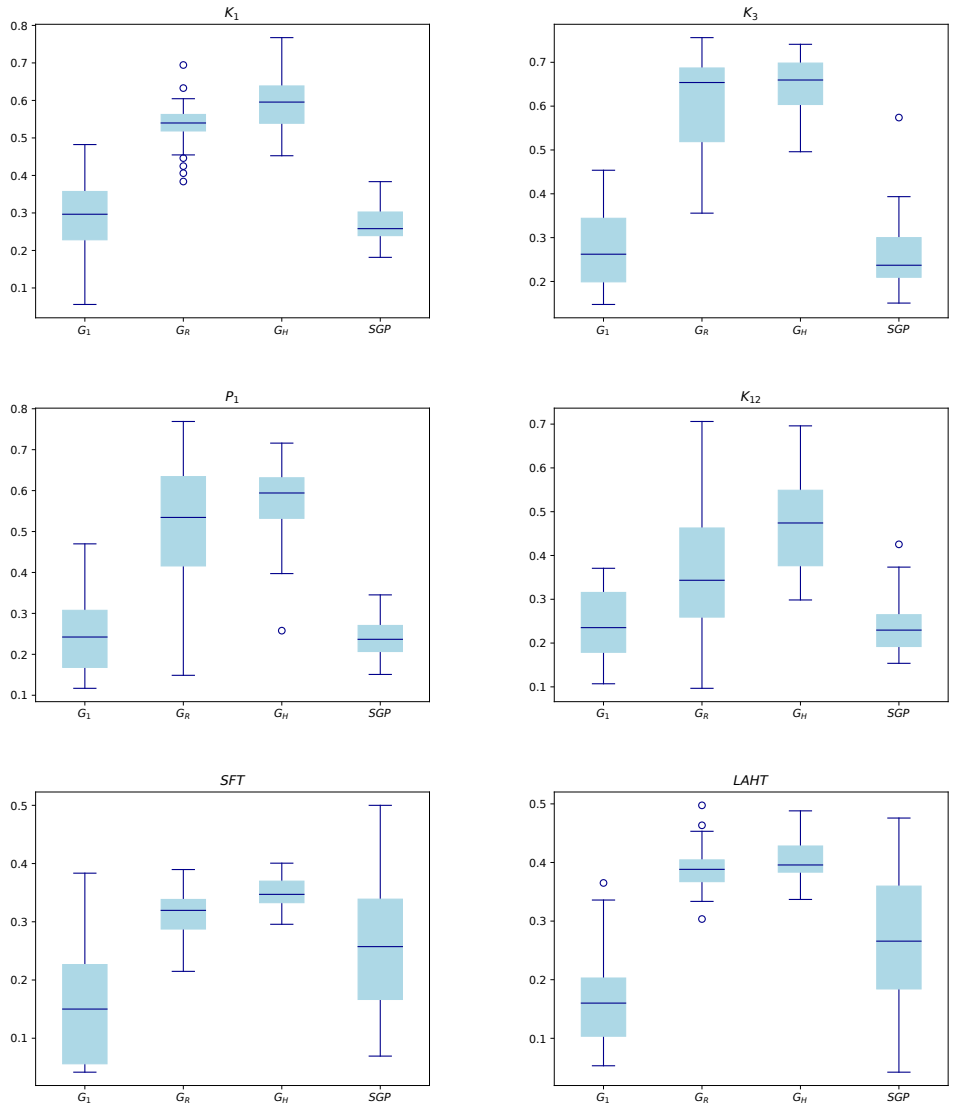


Figure 3.14: Mean normalised genotypic diversity for Koza1 (top-left), Koza3 (top-right), Paige1 (centre-left), Keijzer12 (centre-right), Santa Fe Trail (bottom-left) and the Los Altos Hills Trail (bottom-right).

Once again, statistical hypothesis tests are performed to draw more robust conclusions. Following the aforementioned methodology, the Kolmogorov-Smirnov test is first applied to assess the normality of the data. Its p-values (Table 3.11) are all below the significance value, implying that none of the data can be considered to follow normal distributions. As a result, we must resort to non-parametric tests and we carry on with the Friedman's Anova to assess whether there are statistically significant differences between the ability of the four algorithms to maintain population diversity. This test yields p-values (Table 3.12) below the significance value, indicating the existence of statistically significant differences between the algorithms. As a result, we proceed to apply the Wilcoxon test for performing pairwise comparisons. Once again, the Bonferroni correction is applied to adjust the significance value to 8.33e-03. The results of this test are presented on Table 3.13 and show that G_1 and SGP produce equivalent levels of genotypic diversity in all problems apart from LAHT. Using more iterations, GSynGP consistently produces significantly higher genotypic diversity than SGP. The sole exception is when comparing G_R and SGP in the SFT, where they produce equivalent diversity values. Finally, the results show that the increasing the iterations of GSynGP leads to significantly higher diversity values. The few exceptions ocurr when comparing G_R and G_H in K_3, P_1 and LAHT, where their diversity values cannot be considered to be significantly different.

Behavioural diversity

Finally, the behavioural diversity is assessed. Having analysed the genotypic diversity, the behavioural diversity analysis is justified by the low locality of GP, i.e., individuals with similar genotypes may produce quite distinct behaviours. In case of SR problems, the behaviour of one individual is the output it provides for a given input vector. In case of the artificial ant problems, the behaviour is the trajectory taken, represented by the sequence of coordinates of the visited cells. In both cases, the distance between two behaviours is computed as the Euclidean distance. Similarly to the genotypic diversity, due to computational constraints the behavioural diversity of each algorithm is computed simply by comparing its best individual with all the others.

The behavioural diversity of all algorithms and benchmark problems are plotted on Figure 3.15. For the symbolic regression problems, the existence of outliers with much higher magnitude make the boxplots of little use and so we present a set of descriptive statistics on Table 3.14. As can be seen, in the symbolic regression problems, G_1 typically produces similar diversity values to SGP, while G_R and G_H produce higher values. The exception is K_{12}, where all GSynGP variants produce much higher behavioural diversity than SGP and that diversity increases with the amount of iterations performed. Moreover, in the ant problems the results seem reversed, with G_1 and SGP producing more diverse behaviours than G_R and G_H.

In order to draw more robust conclusions we proceed to the statistical analysis following the same methodology as before. Starting with the Kolmogorov-

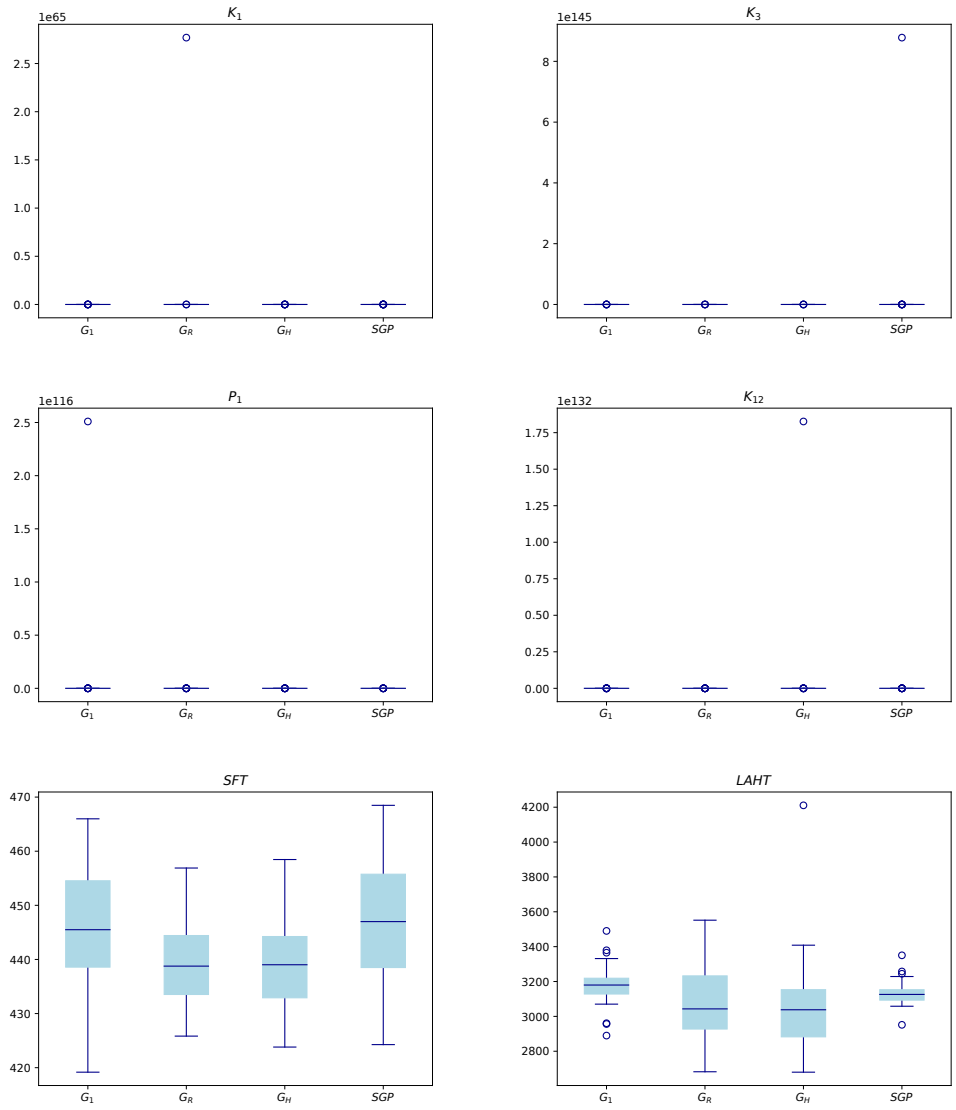


Figure 3.15: Mean behavioural distance for Koza1 (top-left), Koza3 (top-right), Paige1 (centre-left), Keijzer12 (centre-right), Santa Fe Trail (bottom-left) and the Los Altos Hills Trail (bottom-right).

Table 3.14: Descriptive statistics of the mean behavioural distance of the individuals in the last population of each algorithm.

		G_1	G_R	G_H	SGP
K_1	Min	8.55e-01	4.10e+00	3.82e+00	6.14e-01
	Q1	1.26e+00	4.73e+00	5.32e+00	1.19e+00
	Q2	1.90e+00	5.13e+00	5.82e+00	1.46e+00
	Q3	2.93e+00	6.52e+00	1.14e+01	3.81e+00
	Max	1.52e+03	2.77e+65	9.31e+05	7.61e+16
K_3	Min	4.08e-01	6.04e-01	1.25e+00	3.10e-01
	Q1	5.51e-01	1.18e+00	1.68e+00	4.35e-01
	Q2	8.62e-01	1.37e+00	2.03e+00	6.02e-01
	Q3	1.38e+00	1.76e+00	2.23e+00	8.24e-01
	Max	9.28e+115	2.08e+01	4.73e+03	8.78e+145
P_1	Min	1.85e+00	3.02e+00	4.43e+00	1.41e+00
	Q1	3.08e+00	7.87e+00	1.50e+01	2.35e+00
	Q2	5.73e+00	1.98e+01	1.94e+06	4.04e+00
	Q3	5.10e+01	1.26e+07	2.05e+20	5.65e+01
	Max	2.51e+116	4.25e+59	1.07e+65	6.16e+26
K_{12}	Min	1.72e+01	9.36e+01	2.22e+04	9.81e+00
	Q1	3.80e+01	1.90e+06	2.16e+08	2.25e+01
	Q2	5.97e+04	1.61e+16	9.03e+18	5.03e+01
	Q3	2.75e+21	8.21e+37	5.69e+26	2.46e+05
	Max	2.61e+54	5.03e+114	1.83e+132	6.55e+22
SFT	Min	4.19e+02	4.26e+02	4.24e+02	4.24e+02
	Q1	4.39e+02	4.34e+02	4.33e+02	4.39e+02
	Q2	4.45e+02	4.39e+02	4.39e+02	4.47e+02
	Q3	4.55e+02	4.44e+02	4.44e+02	4.56e+02
	Max	4.66e+02	4.57e+02	4.58e+02	4.68e+02
$LAHT$	Min	2.89e+03	2.68e+03	2.68e+03	2.95e+03
	Q1	3.13e+03	2.93e+03	2.88e+03	3.09e+03
	Q2	3.18e+03	3.04e+03	3.04e+03	3.13e+03
	Q3	3.22e+03	3.23e+03	3.15e+03	3.15e+03
	Max	3.49e+03	3.55e+03	4.21e+03	3.35e+03

Table 3.15: P-values of the Kolmogorov-Smirnov test applied to the mean behavioural diversity of the final populations of each algorithm.

	G_1	G_R	G_H	SGP
K_1	4.79e-19	4.92e-22	5.19e-22	1.47e-25
K_3	4.92e-22	1.24e-07	9.28e-19	4.92e-22
P_1	4.92e-22	4.92e-22	5.27e-22	4.94e-22
K_{12}	6.18e-22	4.92e-22	4.92e-22	1.13e-19
SFT	1.47e-04	1.80e-06	1.90e-04	6.09e-05
$LAHT$	6.65e-03	7.92e-05	3.71e-05	1.36e-04

Table 3.16: P-values of the Friedman's Anova applied to the mean behavioural diversity of the final populations of each algorithm.

K_1	K_3	P_1	K_{12}	SFT	$LAHT$
1.61e-06	9.29e-09	4.95e-06	8.84e-07	3.04e-02	1.22e-02

Smirnov test (Table 3.15), we verify that none of the data can be considered to follow normal distributions and thus we must resort to non-parametric tests. We proceed to apply the Friedman's Anova (Table 3.16), verifying that there are statistically significant differences between the algorithms in all benchmark problems. As a result, we resort to the Wilcoxon test to perform pairwise comparisons, using the Bonferroni correction to adjust the significance value to 8.33e-03. The results of this test are presented on Table 3.17 and show that using the adjusted significance value, none of the comparisons in the ant problems can be considered to contain statistically significant differences. Regarding the symbolic regression problems, the Wilcoxon test shows that G_1 always produces equivalent behavioural diversity values to SGP. Also, the three variants of GSynGP tend to produce equivalent values of behavioural diversity, with the exceptions being when comparing G_1 to G_R and G_H in K_1 and when comparing G_1 to G_H in P_1. In those cases, using more iterations leads to significantly higher behavioural diversity. Also, the results show that G_R produces significantly more diverse behaviours than SGP in K_{12} and so does G_H in K_3, P_1 and K_{12}.

Table 3.17: P-values of the Wilcoxon test applied to the mean behavioural diversity of the final populations of each algorithm.

	G_1 - SGP	G_R - SGP	G_H - SGP	G_1 - G_R	G_1 - G_H	G_R - G_H
K_1	**4.65e-01**	**4.49e-02**	**1.41e-01**	3.16e-03	2.96e-03	**1.06e-01**
K_3	**1.47e-01**	**1.11e-02**	3.59e-04	**4.28e-02**	**1.17e-02**	**9.63e-04**
P_1	**2.21e-01**	**2.85e-02**	1.25e-04	**2.45e-01**	3.16e-03	**7.19e-02**
K_{12}	**5.45e-02**	2.22e-04	5.31e-05	**4.95e-02**	**1.99e-01**	**8.45e-01**
SFT	**7.66e-01**	**1.25e-02**	**9.27e-03**	**3.16e-02**	**1.75e-02**	**8.45e-01**
$LAHT$	**4.07e-02**	**1.11e-01**	**4.07e-02**	**9.84e-03**	**1.32e-02**	**6.88e-01**

3.3 Discussion

The results presented in the previous section report a thorough analysis of the novel geometric crossover method versus the standard GP algorithm, covering fitness in both train and test, size of the individuals and population diversity both at the genotypic and behavioural (semantic) levels. The analysis was carried out in popular benchmark problems from the domains of symbolic regression and path planning. The statistically validated results show that in the symbolic regression problems, GSynGP tends to produce worse fitness values in train than SGP. The sole exception occurs in K_1, where G_1 produces equivalent fitness values to those of SGP. Also in K_1, increasing the amount of GSynGP iterations leads to worse results, while in the remaining problems it tends to produce no significant differences. However, when considering the fitness in test, all approaches produce equivalent results. Regarding the ant problems, there were no statistically significant differences in LAHT and in SFT, both G_1 and G_R produce equivalent fitness values to SGP but G_H produced managed to outperform the standard algorithm. Regarding the size of the individuals, the results show that GSynGP always produces significantly smaller solutions than SGP and that increasing the amount of GSynGP iterations typically leads to further reduction of the size of the solutions. Focusing on the genotypic diversity, it is interesting to see that G_1 tends to produce equivalent values to SGP and that the diversity increases with the amount of iterations of GSynGP. However, the increase in genotypic diversity translates to performance gains in the SFT and performance losses in the symbolic regression problems. This may be due to the nature of the problems themselves. As a single operation in GSynGP may result in a big rearrangement of the tree, this may have devastating impact in a symbolic regression solution. In turn, considering that the ant problems only have two functions and three terminal symbols, the effects of these operations may result in more useful solutions. This is also supported by the behavioural diversity values. In the ant problems increasing the amount of iterations leads to slightly lower diversities (but without statistical significance), whereas in the symbolic regression domain, G_H produces significantly more diverse behaviours than G_1 in two benchmark problems, which in turn always produces equivalent behaviours to SGP.

As a result, a good compromise would be to use G_1 in place of SGP, as it can produce solutions with equivalent performance and much smaller size, whilst maintaining equivalent levels of population diversity, both at the genotypic and behavioural levels. On the other hand, if one prioritizes small solutions rather than fitness, G_R or G_H would be preferred, at a possible cost of performance.

Chapter 4

Single-robot approaches for odour source localisation

Contents

The existing robotic approaches for odour source localisation may be divided based on whether they attempt to reach the location of the source or whether they aim to estimate its position. In this chapter, we present three novel evolutionary single-robot approaches for odour source localisation:

- The evolution of tree-based strategies for reaching the odour source, which is presented together with the process of designing fitness functions for this task (Section 4.2);
- The automatic parametrisation of Infotaxis, a popular approach for estimating the position of the chemical source (Section 4.3);
- The combination of Infotaxis with elementary behaviours and perceptions into tree-based controllers, resulting in strategies that both seek and estimate the source location (Section 4.4).

Due to the nature of the odour source localisation task, it would not be feasible to evolve the controllers through Embodied Evolution [Watson et al., 2002, Bredeche et al., 2018], as it would require releasing a chemical substance in the laboratory for an extended time period (hours or days), which would make the environment become saturated and force us to interrupt the experiments. Moreover, such long operation would also wear out the robots which, to increase the feasibility, would have to possess self-charging capabilities. Moreover, the available wind-tunnel is only capable of generating stable chemical plumes, which would restrict the scope of environments tested. As a result, before presenting the proposed approaches, we present the developed simulator.

4.1 Simulator

The first step in preparing the experiments consisted in reviewing the existing simulators in search for one that modelled the airflow and chemical dispersion phenomena with enough realism while being sufficiently fast to ensure the feasibility of the experiments. Unfortunately, the available simulators either do not model gas dispersion and air-flow, or do so with such detail that become too slow for being used in learning or evolutionary robotics experiments. For that reason, we resorted to implementing a simulator from the ground up following the guidelines of [Mouret and Chatzilygeroudis, 2017] to meet the aforementioned requirements.

The developed simulator models the world in 2D and uses simplified kinematics models for reducing the computational complexity. The focus is on properly modelling the air flow and chemical dispersion, for which Farrell et al.'s models [Farrell et al., 2002] are used, as described in Sections 2.1.1.1 and 2.1.1.2. To speed-up the simulations, the chemical dispersion and air flow are modelled a priori and played back on each simulation. The resulting simulator is a good compromise between accurately modelling the real-world and execution speed, being more than 7500 times faster than real time (averaged over 10000 single-robot evaluations). This speed-up makes the developed simulator adequate for

learning and evolutionary robotics experiments, where the evaluations of the candidate solutions are typically the most time consuming part of the process. The proposed simulator models the world as rectangular arenas, which may be empty or contain obstacles. Moreover, each environment may contain one, or several chemical sources, whose locations are drawn randomly from a specified region. Moreover, the start location of each robot is also drawn randomly from a pre-defined region. All of the parameters used in this simulator can be easily reconfigured to create distinct environments.

The simulated robots are circular-shaped differential-driven units. Each robot is equipped with the necessary sensors for locating odour sources, i.e., a laser range finder (LRF) for obstacle avoidance, an anemometer and a gas sensor. The gas sensor and anemometer were already described, respectively in Sections 2.1.1.2 and 2.1.1.1. The laser range finder is composed by a set of equally spaced beams centred on the front of the robot that measure the distance to nearby objects. The number and range of the beams as well as the field-of-view of the sensor are user-defined parameters. On each simulation step, the robots move with a given linear and angular velocity. Effects of friction, acceleration and uncertainties of the actuators are neglected. The robots are modelled with a 8 cm radius and a maximum velocity of 0.5 m/s.

4.1.1 Environments

Three environments are devised based on a square arena containing no obstacles and a single odour source. The choice for including no obstacles was made due to the goal of evolving search strategies for locating chemical sources in large outdoor spaces, where the influence of a few obstacles is not very significant. Different airflow and source characteristics were set for each environment, creating an increasing level of difficulty. The stability of the wind is controlled by adjusting the resolution of the grid used to compute its velocity over the environment, as well as by the standard deviation of the Gaussian noise (W_v) used to emulate turbulence. Thirty instances of each environment are created with the same parameters (Table 4.1) but with different seeds for the random generator, creating a different instance for each independent trial with the same underlying characteristics. The position of the chemical source is sampled from its admissible region for each environment instance and kept fixed for all evaluations in the same trial. Unless noted otherwise, a search is considered successful if the robot reaches a position closer than 0.5 m from the chemical source, within a 600 s time limit. In the following subsection, the specifics of each environment shall be described.

4.1.1.1 Stable

The first environment (S. Env.) contains a stable plume with no intermittency. It is depicted on the top-left of Figure 4.1 and is characterised by an initial wind speed (W_s) of 0.5 m/s, W_v of 0.001 and each wind grid cell measuring 2.8 x 2.8 m^2. Also, the filament emission rate is set to 1 Hz.

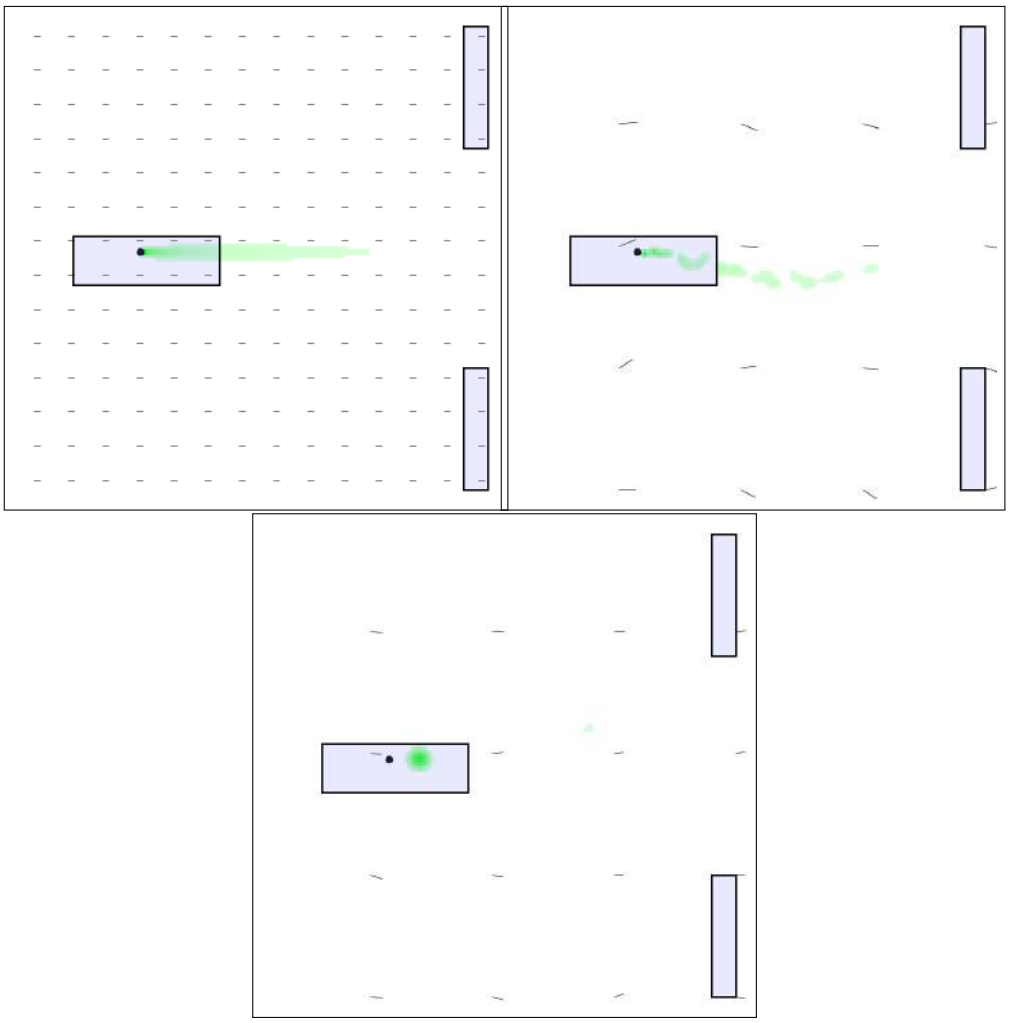


Figure 4.1: Time average of air-flow (grey) and gas dispersion (green) of an instance of the stable (top-left), meandering (top-right) and intermittent-meandering (bottom) environments. The rectangle surrounding the chemical source (black circle) denotes the region from which its location is sampled, whereas the rightmost rectangles represents the start regions for the robot at the odd (bottom) and even (top) numbered evaluations of the corner scenario.

Table 4.1: Environmental Parameters

Parameter	S. Env.	M. Env.	I. M. Env.
Filament emission rate(F_r)	1.0 Hz	1.6 Hz	0.05 Hz
Initial filament radius($f_r(0)$)		0.0316 m	
Filament growth rate (γ)	0.02 m^2/s	0.02 m^2/s	0.05 m^2/s
Chemical Emission rate ($\bar{Q}$)	2.47 mg/s	2.47 mg/s	1.23 mg/s
Detection threshold (D_t)		10 ng/m^3	
Saturation threshold (S_t)		500 ng/m^3	
Initial wind speed (W_s)	0.5 m/s	1.5 m/s	1.0 m/s
Turbulence std. dev. (W_v)	0.001	0.1	0.1
Wind grid spacing	2.8 m	10 m	10 m
Diffusivity constant (K_x)		8	
Arena size		40 m x 40 m	
Source region		$x \sim U(8, 14)$ m, $y \sim U(19.2, 21.2)$ m	
Simulation step		0.5 s	
Simulation time		600 s	

4.1.1.2 Meandering

The second environment (M. Env.) is depicted on the top-right of Figure 4.1 and contains a meandering plume with some intermittency. Its filament emission rate and initial wind speed are increased to respectively 1.6 Hz and 1.5 m/s. The turbulence is set higher at 0.1 and each cell of the grid measures 10 x 10 m^2, meaning that each wind vector influences the filaments for much longer than in the stable environment.

4.1.1.3 Intermittent-Meandering

The third environment (I. M. Env.) contains a meandering plume with high intermittency and is depicted on the bottom of Figure 4.1. It shares all wind-related parameters with the previous environment apart from the initial wind speed which is now set at 1.0 m/s. Its filament emission rate is set to 0.05 Hz and the filament growth rate is set to 0.05 m^2/s, meaning that the filaments will be less frequent and will dissipate faster.

4.1.1.4 Start positions

To study the influence of the robots' start positions, three scenarios are devised for each environment, as depicted on Figure 4.2:

- Corner: where the robots depart from one of the downwind corners of the arena (lower corner on odd numbered evaluations and upper corner on the even numbered evaluations). This region has a width of 2 m on the x-axis and 10 m on the y-axis.
- Border: where the robots depart from the downwind border of the arena. This region has a width of 2 m on the x-axis and 38 m on the y-axis.

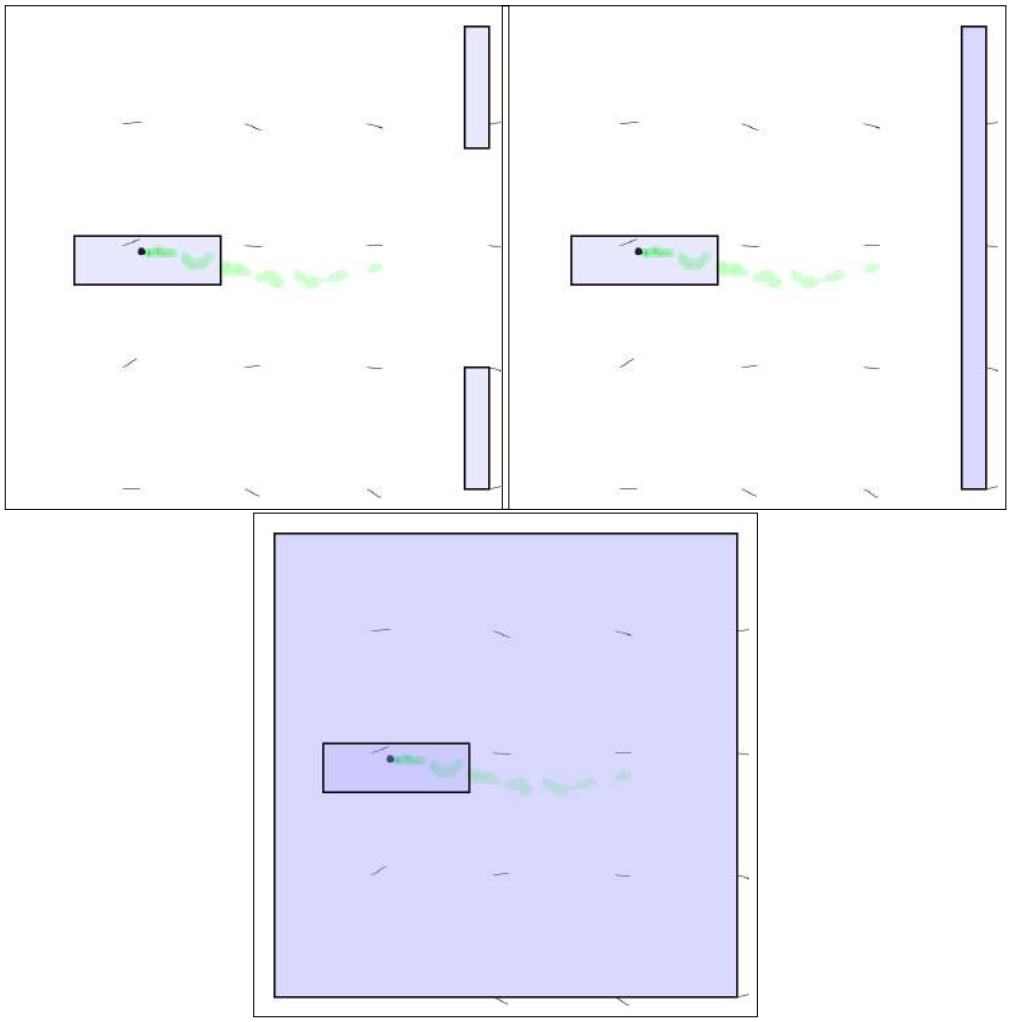


Figure 4.2: Robots' start regions for the meandering environment: corner (top-left), border (top-right) and scattered (bottom).

Table 4.2: Parameters of the EAs

Parameter	Value
Number of generations	100
Size of the population	500
Size of the elite	3
Number of elitist immigrants	15
Number of random immigrants	35
Crossover rate	0.7
Size of the tournament	2

- Scattered: where the robots depart from anywhere in the environment. This region has a width of 38 m on the x-axis and 38 m on the y-axis.

4.2 Evolving tree-based search strategies with Geometric Syntactic Genetic Programming

In this study, Geometric Syntactic Genetic Programming shall be used to evolve tree-based robotic search strategies from bio-inspired behaviours and perceptions. The choice of evolving tree-based controllers is motivated by the desire of producing human-readable search strategies which not only enable the experimenters to draw insights regarding the robot's decision making process but also have an increased robustness to the reality gap by enabling fine-tuning their parameters. Moreover, the use of Genetic Programming is due to the previously reported success in evolving robotic controllers for various tasks. Based on the results of the previous chapter, we shall resort to GSynGP to evolve compact search strategies with improved efficiency and readability over those that could be obtained with the standard GP algorithm.

GSynGP iteratively evolves a population of tree-based search strategies, which in this work are initially generated randomly with a maximum depth of 6 through the ramped-half-and-half method. On each generation, the parent individuals are selected through tournament and recombined through crossover with a 70 % probability. Also, again based on the results from the previous chapter, a single iteration of the geometric crossover operator is used. The extended version of GSynGP is used, evolving trees with multiple symbols per node (e.g., a behaviour and its parameters). As a result, when mutation is applied (30 % probability), a node is randomly selected and either its main symbol or one of its parameters may be changed with respectively 25 %, 75 % probability. The parameters may be mutated in two different manners: if the parameter takes a non-numeric value, its value is simply replaced. Otherwise, it is equally likely that the value is replaced or suffers a Gaussian mutation with zero mean and standard deviation equal to half of the parameters' value (the minimum value for the standard deviation is 0.1). Note that there is no clamping, i.e., the mutation is allowed to drive the values outside of their predefined domain. In order to maintain the population diversity, 50 (10 % of the population size) immigrants are created on

each generation, 35 (7 % of the population size) of which are random, being the remaining 15 (3 % of the population size) elitist. At the end of the generation, a new population is created through elitist selection. The values chosen for these parameters were found through preliminary experimentation and are presented on Table 4.2.

As previously mentioned, GSynGP uses bio-inspired building blocks to evolve its search strategies, with the function set containing:

- *SO* - which informs whether the robot is currently sensing odour;
- *HSO(t)* - which informs whether the robot has sensed odour in the last t seconds, $t \in [1, 500]$ s;
- *Progn* - which executes both subtrees in sequence;

Apart from *Progn*, if the symbols in the function set evaluate to true, the left sub-tree is executed, otherwise the right sub-tree is executed. In turn, the terminal set is composed by:

- *wanderUpwind()* - moves the robot 0.5 m in the upwind direction with a random offset between -45° and 45°;
- *wanderCrosswind()* - moves the robot 0.5 m in the crosswind direction with a random offset between -45° and 45°;
- *wanderDownwind()* - moves the robot 0.5 m in the downwind direction with a random offset between -45° and 45°;
- *moveRandom(d)* - moves the robot d m in the direction it is facing with a random offset between -45° and 45° and $d \in \{0.25, 0.5, 1.0\}$ m;
- *moveUpwind(d)* - moves the robot d m upwind, with $d \in \{0.25, 0.5, 1.0\}$ m;
- *spiral(dis_{inc}, iters, term)* - makes a rectilinear spiral, composed by 6 segments. The first segment measures 0.5 m and the following are increased by dis_{inc} ($dis_{inc} \in \{0, 0.125, 0.25, 0.5, 0.75\}$ m). The spiral is repeated for *iters* iterations (*iters* $\in \{1,2,3\}$) and is halted if a termination criteria *term* is verified (*term* $\in \{C, SO(), PL(t), HSO(t)\}$, where C refers to iteration completed, i.e., the motion terminates normally, *SO* returns true if the robot is currently sensing odour, *PL(t)* returns true if odour has not been sensed for longer than t seconds and *HSO(t)* returns true if odour has been sensed in the last t seconds, with $t \in \{5, 10, 20, 30\}$);

4.2.1 Designing fitness functions for odour source localisation

Up to now, almost all components of the evolutionary system have been defined, being only missing the fitness function. Considering that our goal is to evolve simple and interpretable robotic controllers for locating a chemical source as fast and as reliably as possible, a first approach could be to evaluate the individuals' ability to do just that, i.e., to devise an evaluation function (F) that minimises both the distance of the robot to the chemical source at the end of the trial (d),

as well as the duration of the search (t) (Equation 4.1).

$$F = d + t \tag{4.1}$$

According to Nelson et al.'s taxonomy (Section 2.1.4.4), this is an aggregate fitness function, which despite being straightforward, will often lead to undesired results. Consider the scenario depicted in Figure 4.3, where the robot departs from the bottom-left corner. A single chemical source (brown circle) is emitting odour (green dots), which is carried by the wind towards the rightmost border of the environment. The dotted line represents the trajectory that produces the optimal fitness according to this function, i.e., the trajectory that leads to the chemical source in the least amount of time. When evaluating each candidate solution only once, it is possible (depending also on the function and terminal sets) that the EA finds a controller that overfits the environment, and simply moves in the direction of the chemical source by guessing. Such controller would indeed be useless, as it would not work if the position of the chemical source or the initial pose of the robot varied.

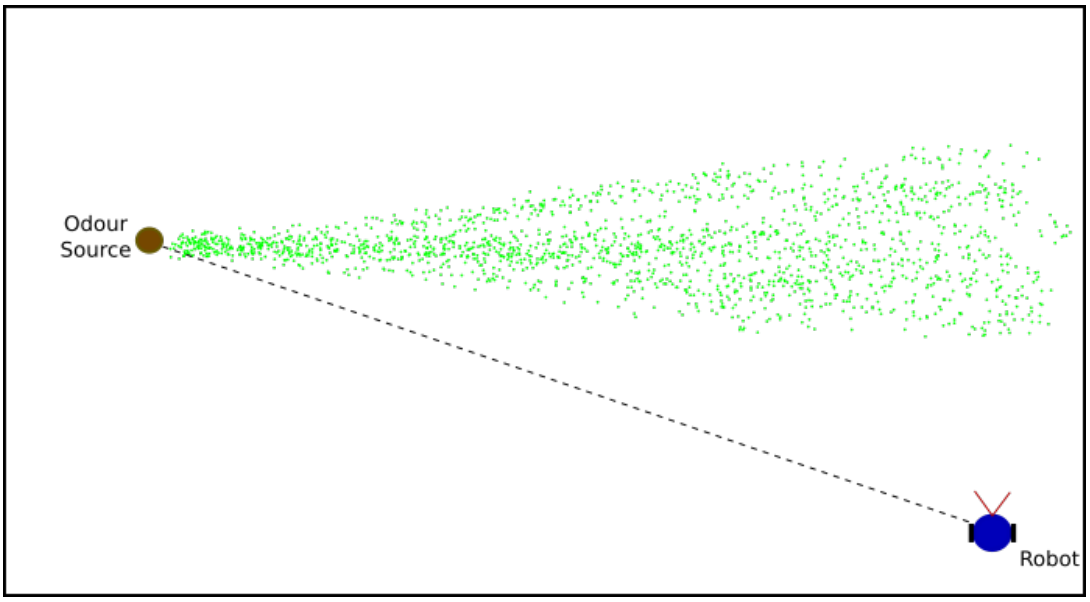


Figure 4.3: Optimal trajectory according to an aggregate fitness function.

There are some terms to refer to this problem. Perhaps the clearest one is the AI misalignment, or the misalignment of goals [Koch et al., 2021]. This term refers to the misalignment between the evaluation function that the experimenter intended and the one that was actually written. In this case, the experimenter wanted to evolve a strategy that was able to *locate* the chemical sources in various scenarios with equivalent wind and chemical dispersion characteristics as the evaluation one. However, the EA evolved a strategy that was able to solve the target problem optimally: i.e., it was able to *reach* the chemical source as fast as possible in the given scenario.

A popular approach to cope with the misalignment of objectives caused by reward functions is to perform various evaluations, each with the robot starting

at a different pose. However, this causes a serious increase of the evolution time. A different approach is reward shaping [Grzes, 2017], which consists on adding additional objectives to increase the density of the evaluation function, assigning more rewards during the trial. The drawback of this approach is that it may bias the evolution, i.e., it may restrict the EAs' ability to discover new ways of solving the task.

4.2.1.1 Influence of the number of evaluations

As previously said, there are two main ways to cope with noisy evaluations in ER: either evaluate each individual multiple times or increase the complexity the evaluation function in an attempt to better analyse the behaviour of the individual. In this section, we take the first option, studying the influence of the number of evaluations. We use a variation of the evaluation function presented earlier to assign the fitness values:

$$F_{1,Ne} = \frac{1}{N} \cdot \sum_{i=1}^{N} (\frac{d}{D} + \frac{t}{T}) \tag{4.2}$$

where N is the number of evaluations performed, and D and T are used to normalize the two terms of the function and respectively represent the maximum possible distance to the chemical source and the maximum evaluation time, which is set to 600 s with a 0.5 s time step. The fitness of an individual is mean of the values attained on all evaluations. To make the experimentation feasible, four values were selected for the number of evaluations: one ($F_{1,1e}$), three ($F_{1,3e}$), five ($F_{1,5e}$) and ten ($F_{1,10e}$). Thirty independent trials were carried out for each number of evaluations, in each combination of environment and start region.

Figure 4.4 presents the boxplots of the success rates attained by the best strategies produced by each evaluation method in the validation step, which consists on taking the thirty best strategies produced by each method (one for each run) and re-evaluating each one of them on the thirty instances of each environment, as decribed in Section 4.1.1. The results show that generally the success rate increases with the amount of evaluations performed, with the biggest gains being made when moving from one to three evaluations.

In order to be able to draw more robust conclusions, we proceed to perform a more thorough statistical analysis, as described in Section 3.2.1.3. Considering a 95 % confidence interval, the results of the Kolmogorov-Smirnov test (Table 4.3) are well below the significance value of 0.05, thus not allowing us to use parametric tests. As a result, the Friedman's Anova (Table 4.4) is applied, showing that there are statistically significant differences between the evaluation methods in all environments.

The Wilcoxon test is then applied to perform pairwise comparisons between the evaluation methods (Table 4.5). The Bonferroni correction is used to adjust the significance value to 8.33e-3. As can be seen, increasing the number of evaluations often leads to significantly higher success rates, with the only case

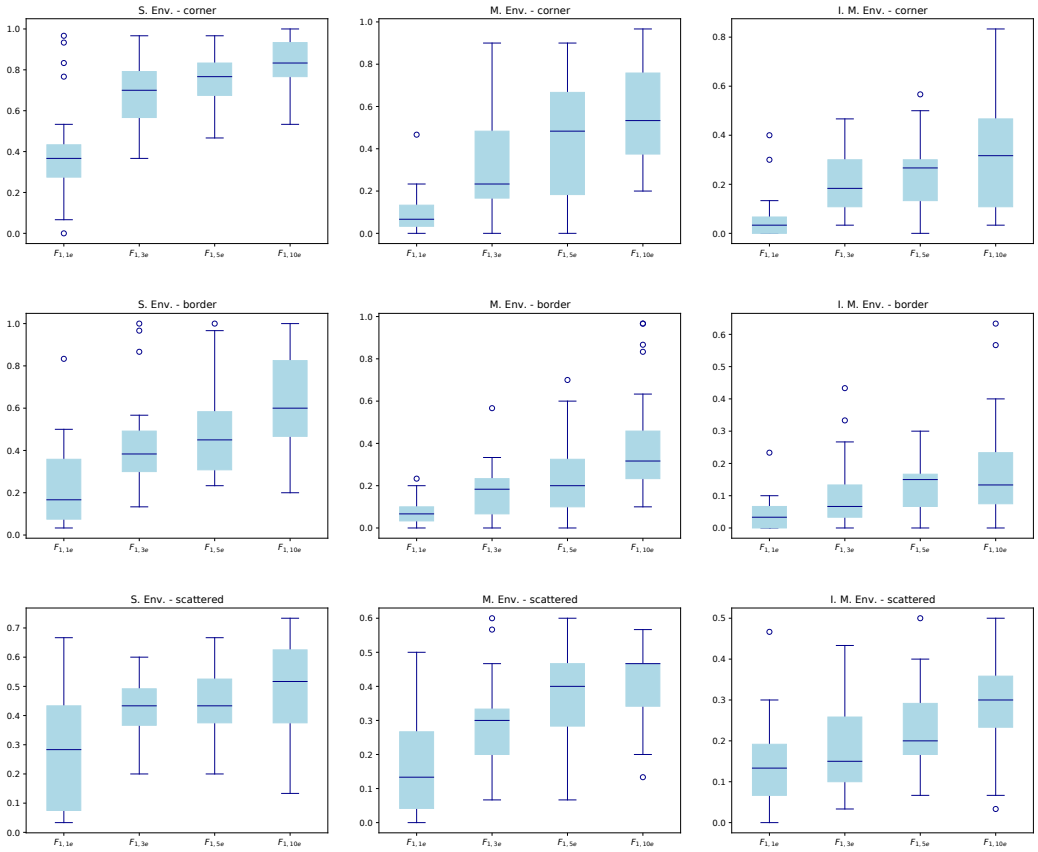


Figure 4.4: Boxplots of the success rates attained in validation in the stable (left column), meandering (centre column) and intermittent-meandering (right column) environments. The top row shows the results of the corner scenario, the middle row shows the results of the border scenario and the bottom row shows the results of the scattered scenario.

Table 4.3: Significance values of the Kolmogorov-Smirnov test applied to the success rates attained in validation.

Start region	Environment	$F_{1,1e}$	$F_{1,3e}$	$F_{1,5e}$	$F_{1,10e}$
Corner	S. Env.	9.85e-04	1.58e-04	9.77e-05	2.66e-04
Corner	M. Env.	4.42e-08	7.18e-07	1.44e-05	1.95e-04
Corner	I. M. Env.	2.81e-07	9.43e-07	7.20e-05	1.30e-05
Border	S. Env.	1.37e-07	1.45e-05	3.43e-06	2.11e-05
Border	M. Env.	1.23e-05	1.10e-05	3.84e-05	7.21e-08
Border	I. M. Env.	5.89e-06	4.75e-09	4.39e-07	1.70e-07
Scattered	S. Env.	1.79e-06	1.58e-03	3.98e-05	2.37e-06
Scattered	M. Env.	2.77e-07	2.55e-06	1.55e-06	2.61e-09
Scattered	I. M. Env.	1.25e-05	4.17e-07	3.54e-06	1.31e-04

Table 4.4: P-values of the Friedman's Anova applied to the validation success rates of strategies evolved with various amounts of evaluations

Start region	S. Env.	M. Env.	I. M. Env.
Corner	1.83e-08	4.13e-10	4.91e-10
Border	2.23e-10	7.66e-10	2.54e-05
Scattered	5.77e-04	2.09e-07	2.70e-05

Table 4.5: P-values of the Wilcoxon test applied to the validation success rates of strategies evolved with various amounts of evaluations

Start region	Functions	S. Env.	M. Env.	I. M. Env.
Corner	$F_{1,3e}$ - $F_{1,1e}$	4.84e-05	5.22e-05	2.56e-05
Corner	$F_{1,5e}$ - $F_{1,1e}$	1.12e-05	4.05e-05	2.57e-05
Corner	$F_{1,10e}$-$F_{1,1e}$	8.71e-06	1.72e-06	1.01e-05
Corner	$F_{1,3e}$ - $F_{1,5e}$	**5.91e-02**	**3.41e-02**	**1.70e-01**
Corner	$F_{1,3e}$ - $F_{1,10e}$	1.41e-03	2.47e-05	4.90e-03
Corner	$F_{1,5e}$-$F_{1,10e}$	**3.97e-02**	**2.46e-02**	**1.33e-02**
Border	$F_{1,3e}$ - $F_{1,1e}$	9.91e-04	8.28e-04	**1.24e-02**
Border	$F_{1,5e}$ - $F_{1,1e}$	4.61e-05	1.56e-04	4.90e-05
Border	$F_{1,10e}$-$F_{1,1e}$	3.49e-06	2.54e-06	2.54e-05
Border	$F_{1,3e}$ - $F_{1,5e}$	**3.26e-01**	**1.74e-01**	**9.15e-02**
Border	$F_{1,3e}$ - $F_{1,10e}$	2.05e-04	3.11e-05	**1.62e-02**
Border	$F_{1,5e}$-$F_{1,10e}$	**1.18e-02**	1.64e-03	**1.44e-01**
Scattered	$F_{1,3e}$ - $F_{1,1e}$	1.27e-03	**1.78e-02**	**9.73e-02**
Scattered	$F_{1,5e}$ - $F_{1,1e}$	8.98e-04	9.37e-05	4.70e-03
Scattered	$F_{1,10e}$-$F_{1,1e}$	6.72e-04	1.63e-05	7.82e-05
Scattered	$F_{1,3e}$ - $F_{1,5e}$	**8.19e-01**	**7.17e-02**	**6.41e-02**
Scattered	$F_{1,3e}$ - $F_{1,10e}$	**1.68e-01**	6.29e-04	1.74e-04
Scattered	$F_{1,5e}$-$F_{1,10e}$	**2.74e-01**	**1.51e-01**	**2.24e-02**

where there are consistently no significant differences is when comparing five and three evaluations in all environments and start regions. Also, there is only one instance (border setting of the meandering environment) where it is significantly better to use ten evaluations rather than five. There are even two cases (border start region in the I. M. Env. and scattered start region in the S. Env.) where there are no statistically significant performance gains in using ten evaluations rather than only three. Moreover, in three of the most difficult settings (i.e., I.M. Env. with the border and scattered start regions and the M. Env. with the scattered start region) there are also no statistically significant gains in using three evaluations rather than a single one.

To sum up, making more evaluations often leads to significantly higher success rates. However, the evolution time increases linearly with the amount of evaluations and most performance gains are made by increasing the evaluations from one to three. As a result, in the following section, we shall compare the influence of different fitness functions when making only one and three evaluations of each candidate solution.

4.2.1.2 Reward Shaping

Having assessed the robustness gains of performing multiple evaluations, in this section we investigate another simple method of coping with noisy and sparse reward functions: reward shaping [Macedo et al., 2021a]. Our goal is to produce an evaluation function that provides a meaningful quality measure from a single evaluation, leading to significant reductions of the evolution time. Furthermore, we are interested in including the minimum amount of prior knowledge, in an attempt to reduce the amount of bias introduced into the evolution.

We start with the simplest evaluation function (Equation 4.2), which was already used in the previous section and evaluates an individual solely through its ability to get close to the chemical source and by the time spent during the search. Intuitively, if the robot spends most of its time in contact with the plume, it should be able to track it better. As a result, a second evaluation function is proposed, which encourages the robot to minimise the time spent without sensing odour:

$$F_{2,Ne} = \frac{1}{N} \cdot \sum_{i=1}^{N} (\frac{d}{D} + \frac{t}{T} + \frac{t_p}{t}) \tag{4.3}$$

where t_p is the time spent without sensing odour. The newly introduced component should not only encourage the robot to stay in contact with the plume, but also to find it as quickly as possible. Note that, while this component introduces some prior knowledge (the concept that the agent should stay within the plume to be able to track it) it does not specify how the robot should proceed to find or keep in touch with the plume. According to the followed taxonomy, $F_{2,Ne}$ can be considered to be a tailored fitness function. Finally, two additional components of prior knowledge are introduced, creating a behavioural fitness function that should make the evolution easier. The first of these components models the concept that, in environments with strong airflows, the odour spreads mainly through advection. As a result, if the robot moves upwind when sensing odour,

it should move closer to the chemical source. This component is computed as follows:

$$u = 1 - \sum_{i=1}^{N_{so}} \left(\frac{d_i \cdot \max(\cos(\theta_{u,i}), 0)}{v \cdot \Delta t} \right)$$

where N_{so} is the amount of steps sensing odour, d_i is the distance travelled in step i, $\theta_{u,i}$ is the upwind direction in step i in the robot's local coordinates, v is the robot's linear speed and Δt is the duration of the control step. The second component models the concept that when the robot loses contact with the plume, it is typically because: (1) the robot moved too much crosswind; (2) the meandering effect of the plume made it move away from the robot; or (3) the robot moved past the chemical source but without reaching the goal region (i.e., a distance close enough to the source so that it can be considered to have been found). In the presence of chemical plumes with little intermittency, the robot should be more successful in re-encountering odour if it searches in the crosswind and downwind directions. Moreover, much like the previous component, such behaviours are present in OSL strategies inspired by natural behaviours. This component is computed as follows:

$$l = 1 - \sum_{i=1}^{N_{nso}} \left(\frac{d_i \cdot (\max(\cos(\theta_{x,i}), 0) + \max(\cos(\theta_{d,i}), 0))}{2v \cdot \Delta t} \right)$$

where N_{nso} is the amount of control steps where the robot does not sense odour and $\theta_{x,i}$ and $\theta_{d,i}$ are respectively the crosswind and downwind directions in step i in the robot's local coordinates. The resulting fitness function to be used in this work is computed by Equation 4.4.

$$F_{3,Ne} = \frac{1}{N} \cdot \sum_{i=1}^{N} \left(\frac{d}{D} + \frac{t}{T} + \frac{t_p}{t} + u + l \right) \tag{4.4}$$

Two variants are created for each of the designed fitness function, differing on whether one or three evaluations are used and thirty independent runs of GSynGP are made for each combination of evaluation function, environment and start region. For each independent run, a new instance of the environment is created, differing in the initial pose of the robot, as described in Section 4.1.1. In order to assess the generalisation ability induced by the evaluation functions, a validation step is performed, consisting on taking the resulting search strategy from each run and re-evaluating it in the 30 instances of the corresponding environment.

Success rates

Figure 4.5 presents the success rates (S_r) attained in validation with each evaluation function. When using a single evaluation, increasing the complexity of the function often leads to higher success rates. However, exceptions do exists, such as when comparing $F_{2,1e}$ and $F_{3,1e}$ in the stable environment with scattered start region. Still, the major performance gains seem to come from using the mean value of three evaluations, where the three functions become more similar

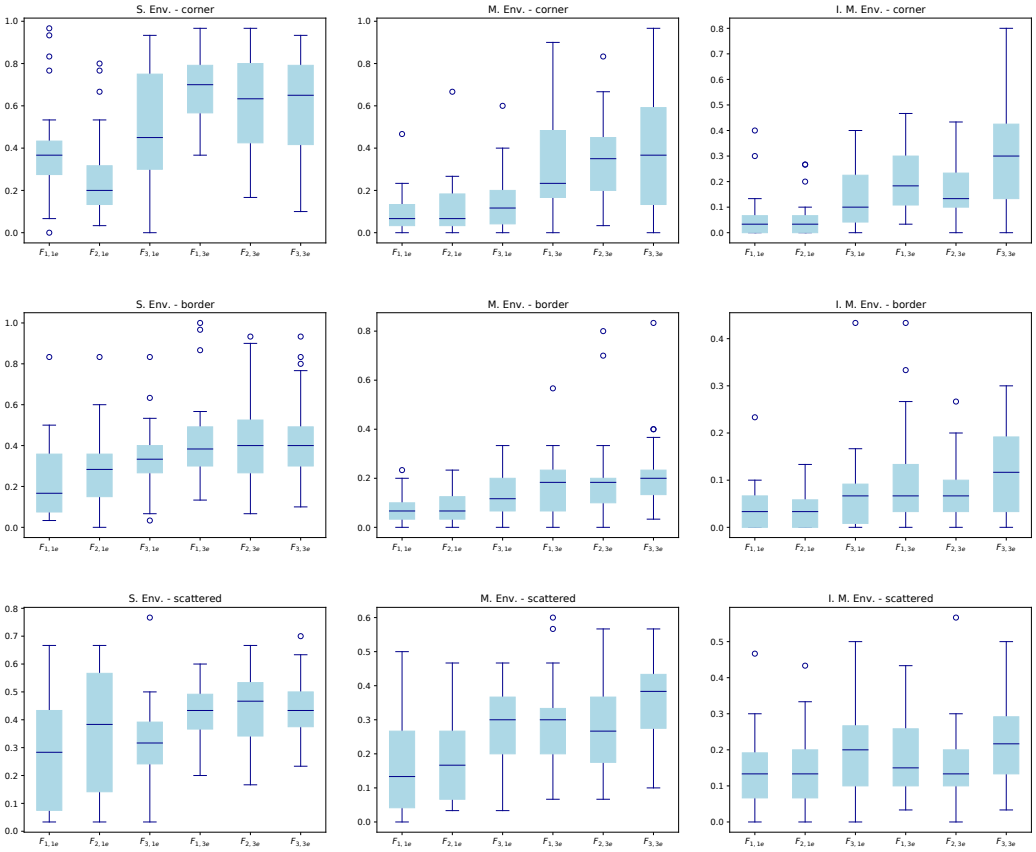


Figure 4.5: Boxplots of the success rates attained in validation in the stable (left column), meandering (centre column) and intermittent-meandering (right column) environments. The top row shows the results of the corner scenario, the middle row shows the results of the border scenario and the bottom row shows the results of the scattered scenario.

to each other.

In order to draw more robust conclusions, an inferential statistical analysis is once again conducted, following the methodology described in Section 3.2.1.3. The Kolmogorov-Smirnov test (Table 4.6) shows that none of the data sets can be considered to follow normal distributions and, as a result, non-parametric tests must be applied. For that reason, the Friedman's Anova was applied for each combination of start region and environment, as presented on Table 4.7. Its results show that there are statistically significant differences between the success rates of the evaluation methods in all environments and thus we must apply the Wilcoxon test to perform pairwise comparisons. The Bonferroni correction is used to adjust the significance value to 3.33e-03. As presented on Table 4.8, F_2 attains equivalent success rates to F_1 in all scenarios, regardless of the number of evaluations performed. Moreover, $F_{3,1e}$ produces significantly higher success rates than $F_{2,1e}$ in four scenarios, but only manages to significantly outperform $F_{1,1e}$ once. When using three evaluations, the performance of the controllers evolved with $F_{2,3e}$ improves substantially, being surpassed by those evolved with $F_{3,3e}$ on only two settings. Analysing the influence of the number of evaluations

Table 4.6: P-values of the Kolmogorov-Smirnov test applied to the success rates

St. region	Env.	$F_{1,1e}$	$F_{2,1e}$	$F_{3,1e}$	$F_{1,3e}$	$F_{2,3e}$	$F_{3,3e}$
Corner	S.	9.85e-04	2.42e-07	1.44e-05	1.58e-04	3.32e-05	1.88e-06
Corner	M.	4.42e-08	7.10e-08	8.98e-07	7.18e-07	1.77e-04	7.70e-05
Corner	I. M.	2.81e-07	3.38e-13	5.34e-09	9.43e-07	2.88e-07	5.43e-05
Border	S.	1.37e-07	7.03e-04	2.58e-05	1.45e-05	9.44e-04	1.10e-04
Border	M.	1.23e-05	1.05e-08	3.43e-07	1.10e-05	1.36e-04	6.93e-06
Border	I. M.	5.89e-06	4.00e-04	2.64e-06	4.75e-09	1.25e-05	1.75e-06
Scattered	S.	1.79e-06	8.03e-07	1.37e-04	1.58e-03	2.94e-05	4.79e-06
Scattered	M.	2.77e-07	1.59e-05	2.85e-05	2.55e-06	3.00e-05	1.51e-05
Scattered	I. M.	1.25e-05	5.58e-06	7.76e-04	4.17e-07	1.70e-06	1.93e-05

Table 4.7: P-values of the Friedman's Anova applied to the success rates

St. region	S. Env.	M. Env.	I. M. Env.
Corner	1.46e-08	7.47e-07	3.31e-15
Border	3.10e-05	1.36e-07	3.79e-04
Scattered	1.73e-05	7.34-08	1.54e-03

on each function, one may conclude that the performance differences reduce as the complexity of the function increases, as the single evaluation variants of $F_{1,1e}$, $F_{2,1e}$ and $F_{3,1e}$ produce significantly worse success rates than their three-evaluation counterparts in respectively six, four and three scenarios.

Duration of the successful searches

Having assessed the success rates produced with each evaluation method, this section analyses the influence of the evaluation method in the duration of the successful runs. Ideally, the evaluation method should maximise the success rate whilst minimising the duration of the search. Figure 4.6 presents the boxplots of the mean duration of the successful evaluations in validation. As can be seen, in the stable environment $F_{2,1e}$ seems to take the longest, followed by $F_{2,3e}$, which may be due to the function putting a stronger emphasis than the others in minimising the time spent without sensing odour, i.e., using F_2, the robot is strongly encouraged to remain in contact with the plume in detriment of the duration of the search. However, as the environments become more difficult, the differences between the various approaches become less obvious.

In order to be able to draw more robust conclusions, statistical hypothesis tests are once again applied. Starting with the Kolmogorov-Smirnov test (Table 4.9) we can consider that none of the data sets follow normal distributions and, thus, a non-parametric group test, i.e., the Friedman's Anova, must be applied. Its results, presented on Table 4.10, show that there are statistically significant differences between the speed of the approaches in all scenarios of the stable and meandering environments. As a result, the Wilcoxon test is applied to perform

Table 4.8: P-values of the Wilcoxon Test applied to the success rates

St. region	Functions	S. Env.	M. Env	I. M. Env.
Corner	$F_{1,1e} - F_{2,1e}$	**1.37e-02**	**7.80e-01**	**8.06e-01**
Corner	$F_{1,1e} - F_{3,1e}$	**1.38e-01**	**1.52e-01**	**6.73e-03**
Corner	$F_{1,1e} - F_{1,3e}$	4.84e-05	5.22e-05	2.56e-05
Corner	$F_{1,1e} - F_{2,3e}$	**5.31e-03**	1.29e-05	8.07e-05
Corner	$F_{1,1e} - F_{3,3e}$	**4.76e-03**	2.92e-04	1.83e-05
Corner	$F_{2,1e} - F_{3,1e}$	2.49e-03	**1.21e-01**	1.71e-03
Corner	$F_{2,1e} - F_{1,3e}$	6.30e-06	1.57e-03	1.57e-05
Corner	$F_{2,1e} - F_{2,3e}$	1.96e-05	6.59e-05	6.66e-06
Corner	$F_{2,1e} - F_{3,3e}$	1.22e-05	8.09e-05	1.67e-05
Corner	$F_{3,1e} - F_{1,3e}$	**6.64e-03**	**1.14e-02**	**1.30e-02**
Corner	$F_{3,1e} - F_{2,3e}$	**6.60e-02**	1.15e-03	**1.70e-01**
Corner	$F_{3,1e} - F_{3,3e}$	**1.12e-01**	2.02e-03	3.57e-04
Corner	$F_{1,3e} - F_{2,3e}$	**3.60e-01**	**2.26e-01**	**1.00e-01**
Corner	$F_{1,3e} - F_{3,3e}$	**1.68e-01**	**3.60e-01**	**3.51e-02**
Corner	$F_{2,3e} - F_{3,3e}$	**5.92e-01**	**7.51e-01**	1.64e-03
Border	$F_{1,1e} - F_{2,1e}$	**3.80e-01**	**6.36e-01**	**8.40e-01**
Border	$F_{1,1e} - F_{3,1e}$	**4.25e-03**	1.91e-03	**1.08e-01**
Border	$F_{1,1e} - F_{1,3e}$	9.91e-04	8.28e-04	**1.24e-02**
Border	$F_{1,1e} - F_{2,3e}$	**3.68e-03**	4.73e-05	**2.78e-02**
Border	$F_{1,1e} - F_{3,3e}$	7.93e-04	9.76e-06	1.28e-03
Border	$F_{2,1e} - F_{3,1e}$	**1.08e-02**	8.57e-04	**5.19e-02**
Border	$F_{2,1e} - F_{1,3e}$	2.54e-03	2.10e-03	**7.41e-03**
Border	$F_{2,1e} - F_{2,3e}$	**1.95e-02**	1.75e-05	**9.22e-03**
Border	$F_{2,1e} - F_{3,3e}$	1.98e-03	1.75e-04	2.60e-04
Border	$F_{3,1e} - F_{1,3e}$	**5.00e-02**	**3.80e-01**	**2.33e-01**
Border	$F_{3,1e} - F_{2,3e}$	**2.59e-01**	**8.32e-02**	**6.06e-01**
Border	$F_{3,1e} - F_{3,3e}$	**6.14e-02**	**2.23e-02**	**1.44e-02**
Border	$F_{1,3e} - F_{2,3e}$	**9.57e-01**	**1.98e-01**	**4.03e-01**
Border	$F_{1,3e} - F_{3,3e}$	**3.25e-01**	**5.00e-02**	**1.86e-01**
Border	$F_{2,3e} - F_{3,3e}$	**9.46e-01**	**1.21e-01**	**5.80e-02**
Scattered	$F_{1,1e} - F_{2,1e}$	**1.48e-02**	**7.34e-01**	**6.26e-01**
Scattered	$F_{1,1e} - F_{3,1e}$	**6.24e-01**	**1.52e-02**	**6.70e-03**
Scattered	$F_{1,1e} - F_{1,3e}$	1.27e-03	**1.78e-02**	**9.73e-02**
Scattered	$F_{1,1e} - F_{2,3e}$	4.21e-04	**9.90e-03**	**3.58e-01**
Scattered	$F_{1,1e} - F_{3,3e}$	1.31e-03	2.33e-05	1.78e-03
Scattered	$F_{2,1e} - F_{3,1e}$	**2.64e-01**	**5.64e-03**	4.60e-04
Scattered	$F_{2,1e} - F_{1,3e}$	**1.25e-01**	3.00e-03	**6.42e-02**
Scattered	$F_{2,1e} - F_{2,3e}$	**2.87e-02**	**1.07e-02**	**3.07e-01**
Scattered	$F_{2,1e} - F_{3,3e}$	**9.76e-02**	2.10e-05	2.19e-03
Scattered	$F_{3,1e} - F_{1,3e}$	4.37e-04	**9.45e-01**	**4.00e-01**
Scattered	$F_{3,1e} - F_{2,3e}$	5.56e-04	**9.31e-01**	**1.11e-01**
Scattered	$F_{3,1e} - F_{3,3e}$	2.49e-04	**5.04e-03**	**4.56e-01**
Scattered	$F_{1,3e} - F_{2,3e}$	**8.12e-01**	**9.18e-01**	**5.59e-01**
Scattered	$F_{1,3e} - F_{3,3e}$	**8.69e-01**	3.24e-03	**1.96e-01**
Scattered	$F_{2,3e} - F_{3,3e}$	**8.82e-01**	1.01e-03	**2.69e-02**

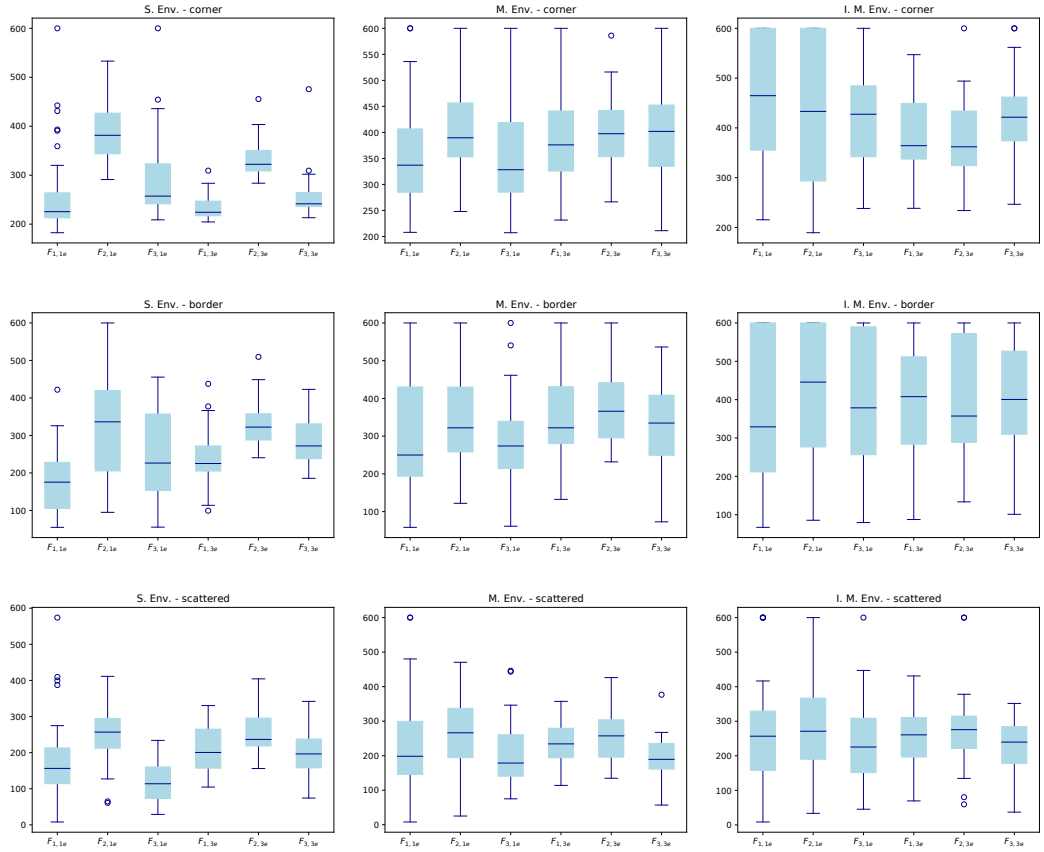


Figure 4.6: Boxplots of the duration of the successful validation runs of the best strategies in the stable (left column), meandering (centre column) and intermittent-meandering (right column) environments. The top row shows the results of the corner scenario, the middle row shows the results of the border scenario and the bottom row shows the results of the scattered scenario.

Table 4.9: P-values of the Kolmogorov-Smirnov test applied to the durations of successful evaluations

St. region	Env.	$F_{1,1e}$	$F_{2,1e}$	$F_{3,1e}$	$F_{1,3e}$	$F_{2,3e}$	$F_{3,3e}$
Corner	S.	2.80e-08	4.07e-04	3.29e-08	2.05e-07	3.06e-07	3.27e-07
Corner	M.	3.69e-05	1.01e-05	2.68e-07	3.10e-04	2.32e-06	2.40e-05
Corner	I. M.	1.65e-06	3.92e-05	5.15e-05	2.78e-06	3.15e-05	4.32e-04
Border	S.	1.32e-05	2.09e-06	4.68e-04	1.82e-05	9.11e-06	2.35e-05
Border	M.	1.13e-07	2.90e-05	5.83e-03	6.08e-06	1.02e-04	5.21e-04
Border	I. M.	5.01e-07	2.71e-06	7.67e-05	1.47e-04	5.68e-07	1.05e-04
Scattered	S.	5.41e-05	4.77e-04	1.04e-04	1.36e-04	4.71e-07	4.59e-04
Scattered	M.	5.27e-04	1.98e-05	3.07e-06	8.66e-05	1.23e-04	2.13e-03
Scattered	I. M.	1.98e-03	4.19e-05	1.29e-04	2.81e-05	1.74e-03	2.96e-05

Table 4.10: P-values of the Friedman's Anova applied to the durations of successful evaluations

St. region	S. Env.	M. Env.	I. M. Env.
Corner	5.55e-16	6.23e-03	**1.46e-01**
Border	1.38e-11	3.47e-02	**9.70e-01**
Scattered	2.53e-09	3.79e-03	**1.53e-01**

pairwise comparisons in those scenarios and the Bonferroni correction is used to adjust the significance value to 3.33e-03. The results of the Wilcoxon test (Table 4.11) show that the only statistically significant differences of the meandering environment occur in the scattered scenario, where $F_{3,3e}$ is faster than $F_{1,3e}$ and $F_{2,3e}$. In the stable environment, the controllers evolved with $F_{2,1e}$ are significantly slower than those evolved with $F_{1,1e}$ and $F_{3,1e}$ regardless of the start region. Moreover, when departing from the border start region, $F_{1,1e}$ produces faster searchers than $F_{3,1e}$. Similarly to the success rate, the speed of the controllers evolved with F_2 increases with the use of three evaluations, particularly in the stable environment, where with the corner start region it is significantly better than $F_{2,1}$ and with the scattered start region it ceases to be significantly different from $F_{1,3e}$. The same trend does not apply to the other fitness functions as the Wilcoxon test found that increasing the number of evaluations produces no statistically differences in F_1 and that $F_{3,3e}$ is significantly slower than $F_{3,1e}$ in one scenario.

Behavioural Diversity

The goal of this study is to devise an evaluation function that maximises the success rate of the search strategies, while minimising the amount of evaluations needed as well as the bias introduced into the evolutionary process. Due to the low locality of GP, we measure the amount of bias introduced into the evolution not through the genotypic diversity, but rather through the behavioural diversity of the best strategies, which in turn is measured through the diversity in the trajectories made during the validation. Figure 4.7 presents the boxplots of the mean behavioural diversity of each search strategy, measured through the euclidean distance between the trajectories performed during validation. Thus, the higher the euclidean distance, the more diverse are the behaviours. Considering a single evaluation, F_2 often produces the most diverse behaviours, being generally followed by F_1 and finally by F_3. This is counterintuive as one would expect that the function with the least amount of prior knowledge would produce the most diverse behaviours. However, it is possible that using F_1 the evolution gets drawn to a local optima region of the search space where the behaviours are all quite similar. When comparing with the three evaluation variants, the general trend points to a reduction of the behavioural diversity versus the single evaluation counterparts, which meets our expectations.

In order to draw more robust conclusions, a more thorough statistical analysis is performed. Table 4.12 presents the results of the Kolmogorov-Smirnov test

Table 4.11: Wilcoxon Test applied to the durations of successful evaluations

St. region	Functions	S. Env.	M. Env	I. M. Env.
Corner	$F_{1,1} - F_{2,1e}$	8.92e-05	**2.67e-02**	-
Corner	$F_{1,1} - F_{3,1e}$	**1.02e-01**	**3.09e-01**	-
Corner	$F_{1,1} - F_{1,3e}$	**4.65e-01**	**1.66e-02**	-
Corner	$F_{1,1} - F_{2,3e}$	8.31e-04	**4.49e-02**	-
Corner	$F_{1,1} - F_{3,3e}$	**6.44e-01**	**5.71e-02**	-
Corner	$F_{2,1} - F_{3,1e}$	8.19e-05	**2.43e-02**	-
Corner	$F_{2,1} - F_{1,3e}$	1.73e-06	**3.15e-01**	-
Corner	$F_{2,1} - F_{2,3e}$	1.38e-03	**4.28e-01**	-
Corner	$F_{2,1} - F_{3,3e}$	6.34e-06	**3.49e-01**	-
Corner	$F_{3,1} - F_{1,3e}$	3.32e-04	**1.78e-01**	-
Corner	$F_{3,1} - F_{2,3e}$	1.04e-03	**1.53e-01**	-
Corner	$F_{3,1} - F_{3,3e}$	**5.19e-02**	**3.49e-01**	-
Corner	$F_{1,3} - F_{2,3e}$	1.92e-06	**8.29e-01**	-
Corner	$F_{1,3} - F_{3,3e}$	**4.99e-03**	**7.04e-01**	-
Corner	$F_{2,3} - F_{3,3e}$	1.24e-05	**9.10e-01**	-
Border	$F_{1,1} - F_{2,1e}$	1.73e-06	**1.70e-01**	-
Border	$F_{1,1} - F_{3,1e}$	9.71e-05	**5.38e-01**	-
Border	$F_{1,1} - F_{1,3e}$	**7.73e-03**	**4.69e-01**	-
Border	$F_{1,1} - F_{2,3e}$	2.60e-06	**1.16e-01**	-
Border	$F_{1,1} - F_{3,3e}$	7.51e-05	**6.44e-01**	-
Border	$F_{2,1} - F_{3,1e}$	2.22e-04	**4.99e-03**	-
Border	$F_{2,1} - F_{1,3e}$	**1.48e-02**	**9.05e-01**	-
Border	$F_{2,1} - F_{2,3e}$	**8.94e-01**	**1.41e-01**	-
Border	$F_{2,1} - F_{3,3e}$	**2.13e-01**	**5.04e-01**	-
Border	$F_{3,1} - F_{1,3e}$	**8.45e-01**	**7.86e-02**	-
Border	$F_{3,1} - F_{2,3e}$	1.20e-03	**3.85e-03**	-
Border	$F_{3,1} - F_{3,3e}$	**1.06e-01**	**3.68e-02**	-
Border	$F_{1,3} - F_{2,3e}$	2.16e-05	**6.56e-02**	-
Border	$F_{1,3} - F_{3,3e}$	2.96e-03	**8.61e-01**	-
Border	$F_{2,3} - F_{3,3e}$	**7.73e-03**	**5.45e-02**	-
Scattered	$F_{1,1} - F_{2,1e}$	8.31e-04	**2.56e-02**	-
Scattered	$F_{1,1} - F_{3,1e}$	**2.70e-02**	**6.00e-01**	-
Scattered	$F_{1,1} - F_{1,3e}$	**2.45e-01**	**5.30e-01**	-
Scattered	$F_{1,1} - F_{2,3e}$	2.77e-03	**1.11e-01**	-
Scattered	$F_{1,1} - F_{3,3e}$	**3.18e-01**	**4.78e-01**	-
Scattered	$F_{2,1} - F_{3,1e}$	1.36e-05	**3.00e-02**	-
Scattered	$F_{2,1} - F_{1,3e}$	**9.37e-02**	**1.99e-01**	-
Scattered	$F_{2,1} - F_{2,3e}$	**7.04e-01**	**6.58e-01**	-
Scattered	$F_{2,1} - F_{3,3e}$	**9.27e-03**	**8.73e-03**	-
Scattered	$F_{3,1} - F_{1,3e}$	3.72e-05	**1.16e-01**	-
Scattered	$F_{3,1} - F_{2,3e}$	2.35e-06	**1.11e-02**	-
Scattered	$F_{3,1} - F_{3,3e}$	1.24e-05	**7.04e-01**	-
Scattered	$F_{1,3} - F_{2,3e}$	**6.42e-03**	**6.56e-02**	-
Scattered	$F_{1,3} - F_{3,3e}$	**2.99e-01**	2.26e-03	-
Scattered	$F_{2,3} - F_{3,3e}$	6.64e-04	7.16e-04	-

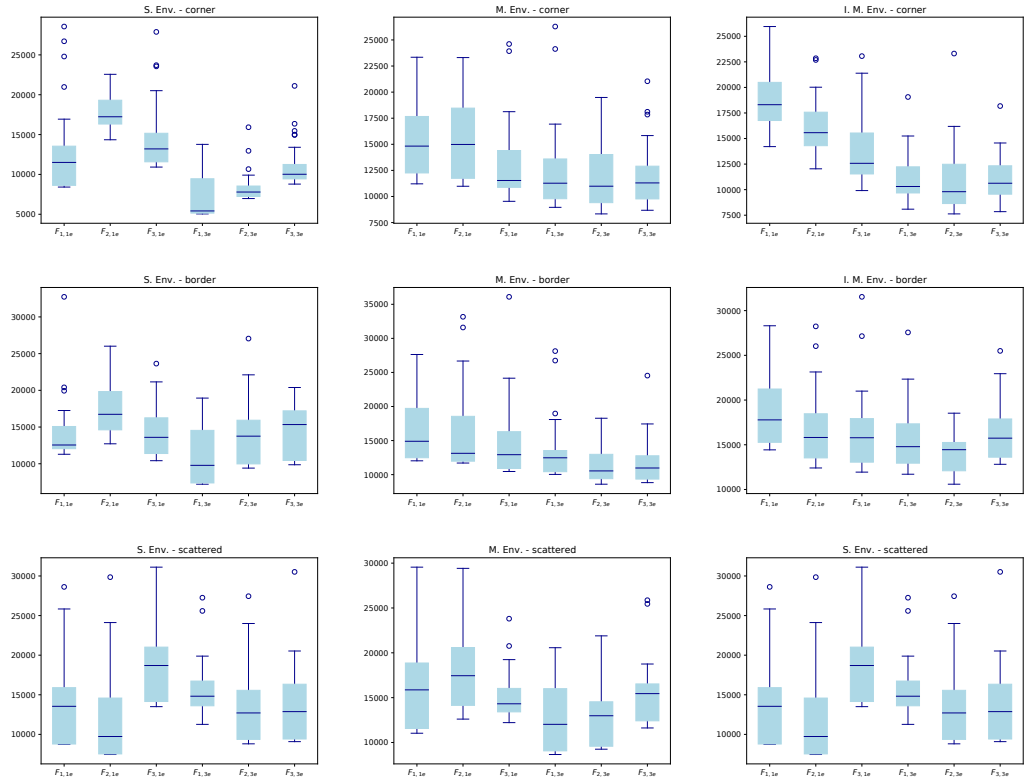


Figure 4.7: Boxplots of the behavioural diversity of the validation runs of the best strategies in the stable (left column), meandering (centre column) and intermittent-meandering (right column) environments. The top row shows the results of the corner scenario, the middle row shows the results of the border scenario and the bottom row shows the results of the scattered scenario.

applied to the various sets of data. As can be seen, at a 95 % confidence interval, none of the data sets can be considered to follow normal distributions and thus non-parametric tests must be applied. The results of the Friedman's Anova (Table 4.13) show that there are statistically significant differences between the behavioural diversity of the strategies in all environments and start regions. As a result, the Wilcoxon test is applied to perform pairwise comparisons of the various evaluation methods. The Bonferroni correction is once again applied to adjust the significance value to 3.33e-03. The results of the Wilcoxon test (Table 4.14) show that, using a single evaluation, $F_{2,1e}$ produces significantly higher levels of behavioural diversity than $F_{1,1e}$ and $F_{3,1e}$ in two scenarios each. In turn, $F_{1,1e}$ produces significantly more diverse behaviours than $F_{2,1e}$ and $F_{3,1e}$ in respectively one and three scenarios. $F_{3,1e}$ produces the least amount of behavioural diversity, only being able to outperform $F_{1,1e}$ and $F_{2,1e}$ in the scattered setting of the stable environment.

Using three evaluations the roles are reversed, with $F_{1,3e}$ and $F_{3,3e}$ producing significantly more diversity than $F_{2,3e}$ in two scenarios. Overall, $F_{1,3e}$ and $F_{3,3e}$ can be considered to perform equivalently, as each one of them is able to outperform the other in two scenarios. Increasing the number of evaluations leads to a general loss of diversity in each function, which is more prevalent in $F_{2,3e}$ and $F_{3,3e}$ than in $F_{1,3e}$.

Table 4.12: P-values of the Kolmogorov-Smirnov test applied to the behavioural diversities

St. region	Env.	$F_{1,3e}$	$F_{1,1e}$	$F_{2,1e}$	$F_{3,1e}$	$F_{2,3e}$	$F_{3,3e}$
Corner	S.	2.25e-06	1.60e-06	2.01e-08	1.35e-09	1.02e-08	5.88e-13
Corner	M.	2.29e-06	8.22e-06	2.34e-08	6.09e-07	7.15e-07	1.44e-05
Corner	I. M.	6.26e-06	3.16e-06	2.48e-07	5.76e-08	7.02e-07	1.47e-05
Border	S.	1.14e-08	4.70e-06	6.75e-07	4.43e-07	1.48e-04	7.29e-07
Border	M.	4.87e-07	1.58e-09	4.02e-07	2.69e-06	2.37e-07	3.78e-06
Border	I. M.	1.02e-06	1.47e-06	8.67e-07	2.79e-07	1.29e-04	1.05e-06
Scattered	S.	1.73e-06	1.91e-07	3.60e-04	4.27e-05	1.78e-06	4.05e-06
Scattered	M.	2.07e-06	6.27e-05	3.77e-08	6.48e-06	1.53e-05	1.67e-03
Scattered	I. M.	4.23e-04	2.02e-05	4.71e-03	6.13e-04	3.76e-06	4.95e-06

Table 4.13: P-values of the Friedman's Anova applied to the behavioural diversities

St. region	S. Env.	M. Env.	I. M. Env.
Corner	4.23e-20	6.89e-09	3.19e-17
Border	9.42e-07	2.74e-13	1.09e-08
Scattered	1.79e-08	7.21e-05	3.46e-06

Table 4.14: P-values of the Wilcoxon test applied to the behavioural diversities

St. region	Functions	S. Env.	M. Env	I. M. Env.
Corner	$F_{1,1e} - F_{2,1e}$	7.71e-04	**9.59e-01**	1.11e-03
Corner	$F_{1,1e} - F_{3,1e}$	**2.85e-02**	**9.27e-03**	7.51e-05
Corner	$F_{1,1e} - F_{1,3e}$	6.32e-05	1.96e-03	2.35e-06
Corner	$F_{1,1e} - F_{2,3e}$	1.80e-05	6.89e-05	4.29e-06
Corner	$F_{1,1e} - F_{3,3e}$	**4.17e-01**	1.89e-04	1.92e-06
Corner	$F_{2,1e} - F_{3,1e}$	2.41e-03	**6.04e-03**	**9.84e-03**
Corner	$F_{2,1e} - F_{1,3e}$	1.73e-06	9.63e-04	3.88e-06
Corner	$F_{2,1e} - F_{2,3e}$	1.73e-06	1.49e-05	2.37e-05
Corner	$F_{2,1e} - F_{3,3e}$	2.88e-06	1.48e-04	4.29e-06
Corner	$F_{3,1e} - F_{1,3e}$	3.18e-06	**5.72e-01**	8.84e-04
Corner	$F_{3,1e} - F_{2,3e}$	1.92e-06	**1.99e-01**	8.31e-04
Corner	$F_{3,1e} - F_{3,3e}$	2.61e-04	**1.25e-01**	3.88e-04
Corner	$F_{1,3e} - F_{2,3e}$	**3.68e-02**	**4.05e-01**	**4.53e-01**
Corner	$F_{1,3e} - F_{3,3e}$	1.97e-05	**4.05e-01**	**9.59e-01**
Corner	$F_{2,3e} - F_{3,3e}$	7.69e-06	**9.75e-01**	**5.86e-01**
Border	$F_{1,1e} - F_{2,1e}$	2.22e-04	**5.44e-01**	**6.84e-03**
Border	$F_{1,1e} - F_{3,1e}$	**6.14e-01**	8.94-04	1.71e-03
Border	$F_{1,1e} - F_{1,3e}$	**3.85e-03**	**5.32e-03**	2.58e-03
Border	$F_{1,1e} - F_{2,3e}$	**5.04e-01**	1.02e-05	9.32e-06
Border	$F_{1,1e} - F_{3,3e}$	**5.04e-01**	1.25e-04	**3.38e-03**
Border	$F_{2,1e} - F_{3,1e}$	7.16e-04	**1.04e-02**	**2.89e-01**
Border	$F_{2,1e} - F_{1,3e}$	1.02e-05	**3.38e-03**	**1.41e-01**
Border	$F_{2,1e} - F_{2,3e}$	**7.27e-03**	3.72e-05	1.96e-03
Border	$F_{2,1e} - F_{3,3e}$	2.58e-03	6.15e-04	**4.91e-01**
Border	$F_{3,1e} - F_{1,3e}$	2.58e-03	**1.92e-01**	**1.65e-01**
Border	$F_{3,1e} - F_{2,3e}$	**6.44e-01**	5.31e-05	**7.27e-03**
Border	$F_{3,1e} - F_{3,3e}$	**8.77e-01**	1.11e-03	**5.30e-01**
Border	$F_{1,3e} - F_{2,3e}$	**1.85e-02**	2.26e-03	**3.85e-03**
Border	$F_{1,3e} - F_{3,3e}$	**5.32e-03**	3.16e-03	**3.18e-01**
Border	$F_{2,3e} - F_{3,3e}$	**4.41e-01**	**4.91e-01**	7.71e-04
Scattered	$F_{1,1e} - F_{2,1e}$	**3.61e-03**	**6.56e-02**	**6.58e-01**
Scattered	$F_{1,1e} - F_{3,1e}$	3.32e-04	**4.78e-01**	**6.73e-01**
Scattered	$F_{1,1e} - F_{1,3e}$	**9.78e-02**	**5.67e-03**	**1.75e-02**
Scattered	$F_{1,1e} - F_{2,3e}$	**3.60e-01**	**6.84-03**	**2.45e-01**
Scattered	$F_{1,1e} - F_{3,3e}$	**6.88e-01**	**4.05e-01**	**1.48e-02**
Scattered	$F_{2,1e} - F_{3,1e}$	3.88e-06	**8.22e-03**	**9.43e-01**
Scattered	$F_{2,1e} - F_{1,3e}$	**6.84e-03**	9.71e-05	**2.18e-02**
Scattered	$F_{2,1e} - F_{2,3e}$	**5.71e-02**	2.22e-04	**1.59e-01**
Scattered	$F_{2,1e} - F_{3,3e}$	**7.52e-02**	**3.68e-02**	**3.38e-03**
Scattered	$F_{3,1e} - F_{1,3e}$	**1.04e-02**	**7.27e-03**	**8.73e-03**
Scattered	$F_{3,1e} - F_{2,3e}$	8.92e-05	**1.17e-02**	**8.97e-02**
Scattered	$F_{3,1e} - F_{3,3e}$	8.19e-05	**5.86e-01**	1.04e-03
Scattered	$F_{1,3e} - F_{2,3e}$	**4.49e-02**	**7.66e-01**	1.36e-04
Scattered	$F_{1,3e} - F_{3,3e}$	**1.40e-02**	2.11e-03	1.92e-06
Scattered	$F_{2,3e} - F_{3,3e}$	**4.78e-01**	**7.27e-03**	**3.16e-02**

4.2.2 Final remarks

This section focused on devising the evaluation function that shall be used throughout the remaining of the document. The goal was to design a function whose output provided a good indication of the quality of the search strategy, both in terms of success rate and duration of the searches, with the least amount of evaluations to minimise the computational overhead. We started by using a simple function to study the influence of the amount of evaluations and concluded that there were little gains past three evaluations. We then proceeded to investigate whether complexifying the function led to better results. We designed and compared three evaluation functions for odour source localisation, each having a single-evaluation and a three-evaluation variants. The results were highly dependent on the scenarios, with no clear patterns, but overall showed that the three evaluation-functions tend to produce significantly higher success rates than their single-evaluation counterparts, but at the cost of lower behavioural diversity. In particular, F_2 was found to produce equivalent success rates and, in some scenarios, higher levels of behavioural diversity than F_1. However, it does so at the cost of having longer searches in some settings, which could be due to the search strategies evolved with F_2 making an additional effort to keep in contact with the chemical plume than those produced by the other fitness functions. Still, the median search times are well below the time limit (note that the time limit is 600 s and in all scenarios the median search time is below 500 s) and thus are deemed acceptable. It should also be noted that the results reported in this chapter are environment-dependent. In a previous work [Macedo et al., 2021a] we showed that, in an scenario with a meandering plume with little intermittency that extends to the downwind border of the environment, $F_{2,1e}$ achieves better success rates that the remaining single-evaluation functions, being even equivalent to $F_{1,3e}$.

Throughout the rest of this thesis we shall use $F_{2,3e}$ for three main reasons: (1) one of the motivations for evolving white-box controllers is to be able to draw insights from them and so, it is interesting to use the function that creates the most diverse behaviours with the minimum amount of bias; (2) when using three evaluations, there are only two settings (out of nine) where F_3 attains higher success rates than F_2. Yet, it is a more complex function, with two behavioural components that introduce more prior knowledge into the system, which may condition the evolution of cooperative multi-robot behaviours (e.g., the controllers would receive better fitness to search the plume crosswind or downwind than moving towards neighbours sensing odour); and (3) we expect that the encouragement to keep in touch with the chemical plume will benefit the evolution in more complex scenarios, such as Evolutionary Infotaxis or the swarm approaches, where we hope that it will respectively reduce Infotaxis' tendency to keep exploring the environment after finding the plume and encourage cooperation between the robots, so that with multiple agents tracking the plume the chance of loosing it is reduced.

4.3 Evolutionary Infotaxis

Infotaxis (Section 2.3.1) is a popular method for estimating the location of a chemical source by fitting a gas distribution model to environmental measurements. Recent studies [Ruddick et al., 2018, Rodríguez et al., 2017, Martin Moraud and Martinez, 2010] have shown that one of its main drawbacks is the need for carefully selecting the parameters for its gas distribution model, as they have a great influence on its performance. As a result, most of Infotaxis' works perform the experiments exclusively in simulation, using the same exact model for mimicking the real world and for computing the probability of the odour source location. Such approaches have little realism, as rather than sampling an instantaneous chemical plume, they sample models that provide a distribution of the time-averaged odour concentration. To make matters worse, the realism of those experiments is further reduced by the common assumption that the odour detections and non-detections are independent events.

In this section we take a different path, where the robots use Infotaxis to track instantaneous chemical plumes with different characteristics. We propose two evolutionary approaches for automatically optimizing the parameters of Infotaxis' gas distribution model (D_f, R and τ from Equation 2.21) for each scenario. Both approaches shall do so with a genetic algorithm, representing each individual by a vector of three real numbers (i.e., the parameters to be optimised).

The approaches differ mainly in the fitness computation, and consequently on the purpose of evolution. The first approach, InfotaxisDB (IDB) aims to evolve the parameters that best emulate the time-averaged chemical dispersion in the environment. It is data-based i.e., it assumes complete knowledge of the environment in both time and space, to compute the mean chemical dispersion over the entire experiment. The mean plume is then used to compute the fitness of each individual, by comparing the plume produced with its parameters to the ground-truth through the Root Mean Squared Error (RMSE).

The second approach is named Evolutionary Infotaxis (EI), and aims to evolve the parameters that maximise the robot's performance in locating the chemical source. As such, the fitness of each individual (set of parameters) is computed by simulating an agent performing the actual search. These simulation-based experiments are fundamentally different to the existing ones in the literature, as they consist of using a mobile robot to sample an instantaneous, meandering plume, where the detections and non-detections of odour are correlated. Based on the findings of the previous section, $F_{2,3e}$ shall be used to evaluate the individuals, setting the fitness as the mean value of three evaluations. The goal of the EAs is to minimise this function, i.e., the lower its value, the better.

At the end of evolution, we shall compare the success rates (S_r) and the duration of the successful evaluations (T_s) of attained with the parameters of IDB, EI and by the search strategies evolved with GSynGP, which we will refer to as G.

4.3.1 Experimental results

As previously said, the individuals evolved by EI are represented by a vector of three real numbers: D_f, R and τ. In order to maximise the diversity of the initial individuals, the initial population is created using Latin Hypercubes [Eiben and Smith, 2015]. Each variable is bounded in a pre-determined domain, chosen by taking into account other works in the literature, being $D_f \in [0.01, 10]$, $R \in [0.1, 500]$ and $\tau \in [0.5, 1000]$. The individuals are recombined through arithmetical crossover, with $\alpha_{cross} = 0.5$, and mutated with a Gaussian operator, that is applied with a probability of 1 % and with each σ being one tenth of the variable's domain width. The remaining parameters are shared with GSynGP and have already been presented on Table 4.2.

Thirty independent trials were conducted for each environment and, in the case of EI, also for each start region. This section reports the experimental results, starting by presenting the models produced by the best individuals for each environment and moving on to compare their performance in locating the chemical source.

4.3.1.1 Environmental models

We start by presenting the mean chemical plumes (ground-truth) for each environment (top row of Figure 4.8). In Section 4.1.1 we stated that each independent trial has a different instance of the assigned environment. For the sake of brevity, we shall only present here the first of the thirty instances. As you can see, most of the environment has no detectable odour. However, both Infotaxis and the source seeking approaches that we propose do not use the exact chemical concentrations, but rather binary detections. For that reason, we chose to compute the mean binary chemical plume for each environment (bottom row of Figure 4.8)), i.e., the time-average of the plumes that may be detected by our sensors, and use that as the target for IDB.

The best parameters found by IDB (lowest RMSE) and by EI (overall best in validation, see next section) are presented on Table 4.15 and the corresponding models are plotted on Figure 4.9. As you can see, IDB makes better approximations of the ground-truth, as is to be expected. Still, its models contain detectable odour up to the downwind border of the arena, which may mislead the robot. In turn, EI creates models where the plume dissipates very quickly in the stable and meandering environment, which is likely to lead the robot to move faster towards the source once odour is found. Interestingly, in the corner and border settings of the intermittent-meandering environment, EI created models that resemble that of IDB, but with broader plumes.

4.3.1.2 Performance analysis

This section analysis the performance of IDB and EI, comparing it to that of tree-based controllers evolved with GSynGP through $F_{2,3e}$. The results presented are the results of the validation step, where the 30 resulting solutions from each approach are re-evaluated on the thirty instances of each environment and

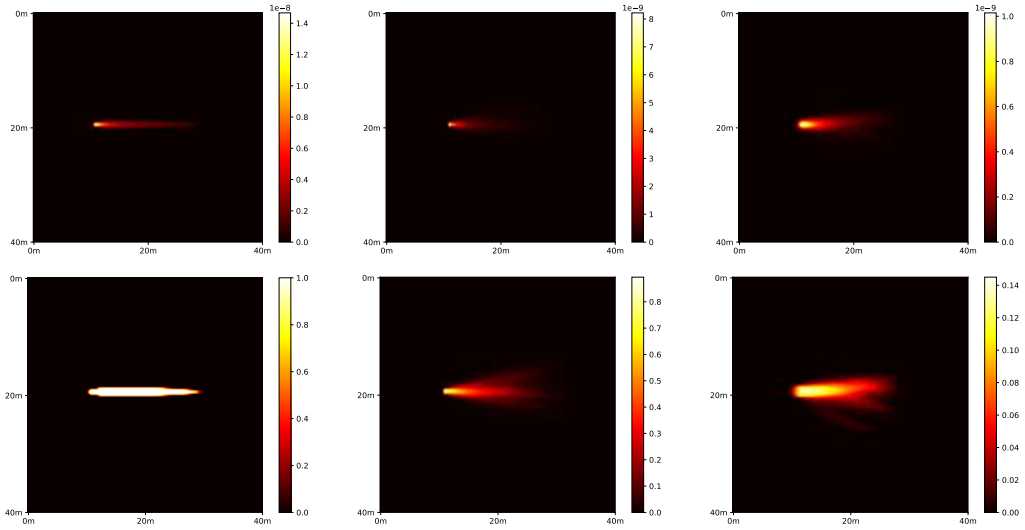


Figure 4.8: Mean chemical plumes (top) and mean binary chemical plumes (bottom) of the first instance of the stable (left), meandering (centre) and intermittent-meandering (right).

start-region combination. The success rates are plotted on Figure 4.10. As can be seen, IDB typically attains much lower success rates than the other approaches. The sole exception is in the stable environment with the scattered start region, where it attains a higher median success rate than the other approaches. Moreover, it is interesting that the tree-based controllers often attain similar success rates to EI. In order to be able to draw robust conclusions, we proceed to perform a statistical analysis. The Kolmogorov-Smirnov test (Table 4.16) shows that none of the data can be considered to follow normal distributions. As a result, the Friedman's Anova is applied (Table 4.17), showing that there are statistically significant differences between in all scenarios.

The Wilcoxon test is then applied to perform pairwise comparisons (Table 4.18), with the Bonferroni correction being used to adjust the significance value to 1.67e-02. This test shows that there are statistical significant differences in most comparisons. When departing from the corner start region, IDB is found to be consistently outperformed by EI and GSynGP. EI significantly outperforms G in the I. M. environment, while attaining equivalent success rates in the S. and M. environments. Using the border start region, IDB is outperformed by EI in all environments and by G in the M. environment. Furthermore, EI performs equivalently to G in the S. environment and outperforms it in the two other environments. Finally, when using the scattered start regions, IDB significantly outperforms G in the S. environment, while being outperformed by G in the two other environments. IDB is also outperformed by EI in the M. and I. M. environments, while performing equivalently to it in the S. environment. Furthermore, EI outperforms G in the M. environment, while performing equivalently to it in the two others.

Moving on to analyse the duration of successful evaluations (Figure 4.11), it is interesting to see that EI is typically the fastest approach, followed by G.

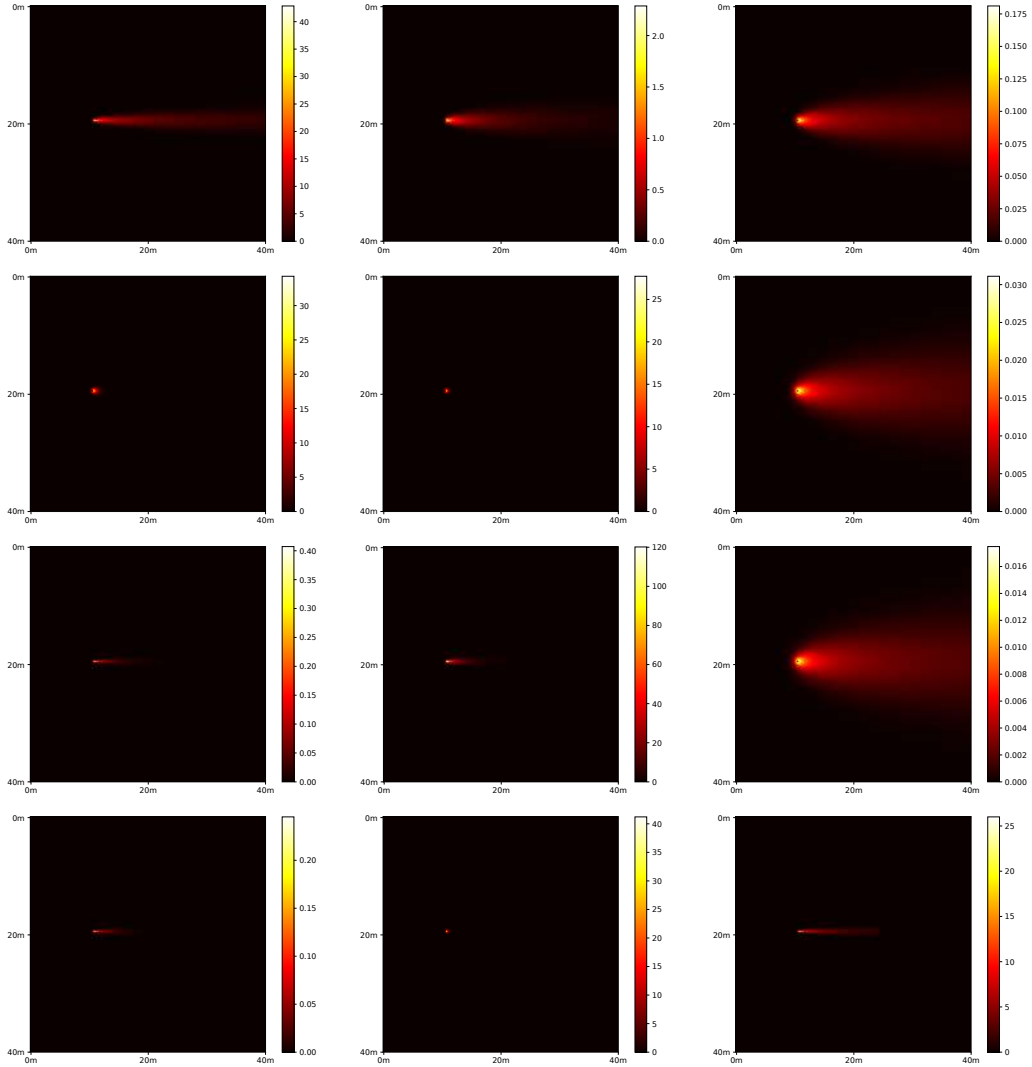


Figure 4.9: Gas distribution models parametrised with the best values found for IDB (top), EI for the corner scenario (second row) EI for the border scenario (third row) and EI for the scattered scenario (bottom row), for the first instance of the stable (left), meandering (centre) and intermittent-meandering (right).

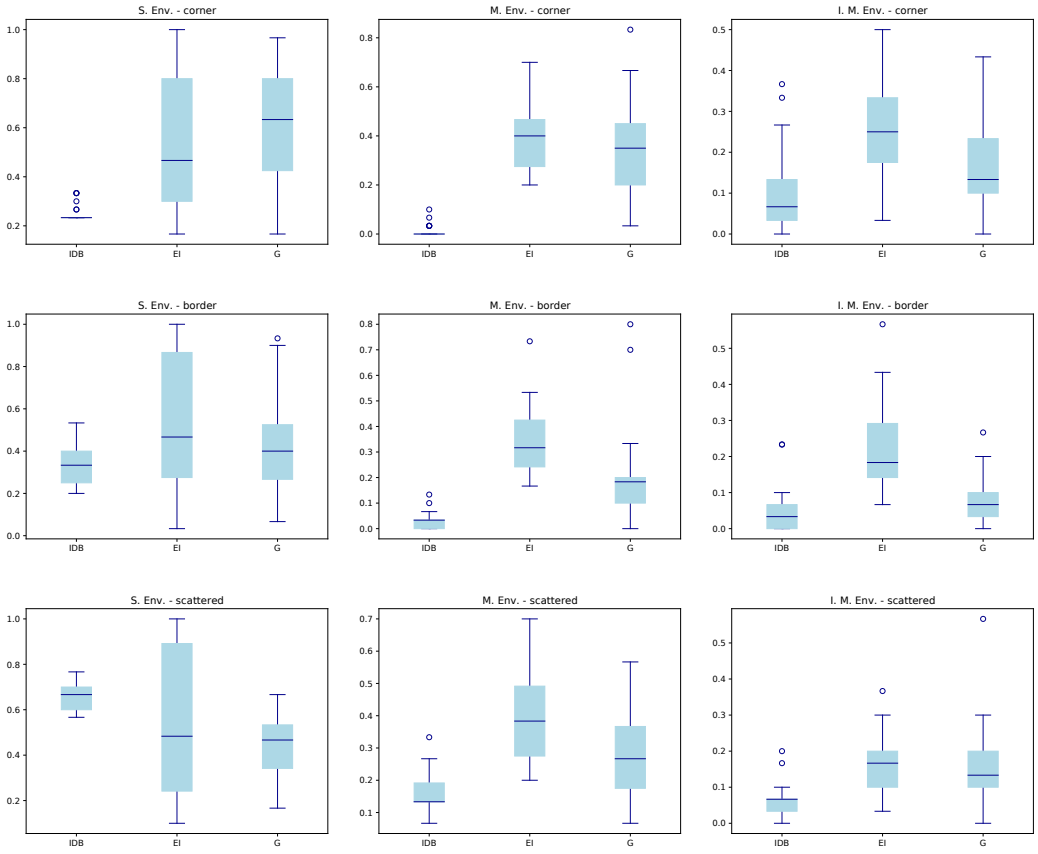


Figure 4.10: Success rates in validation for the S. (left column) M. (centre column) and I. M. environments, with the corner (top row), border (centre) and scattered (bottom) start regions.

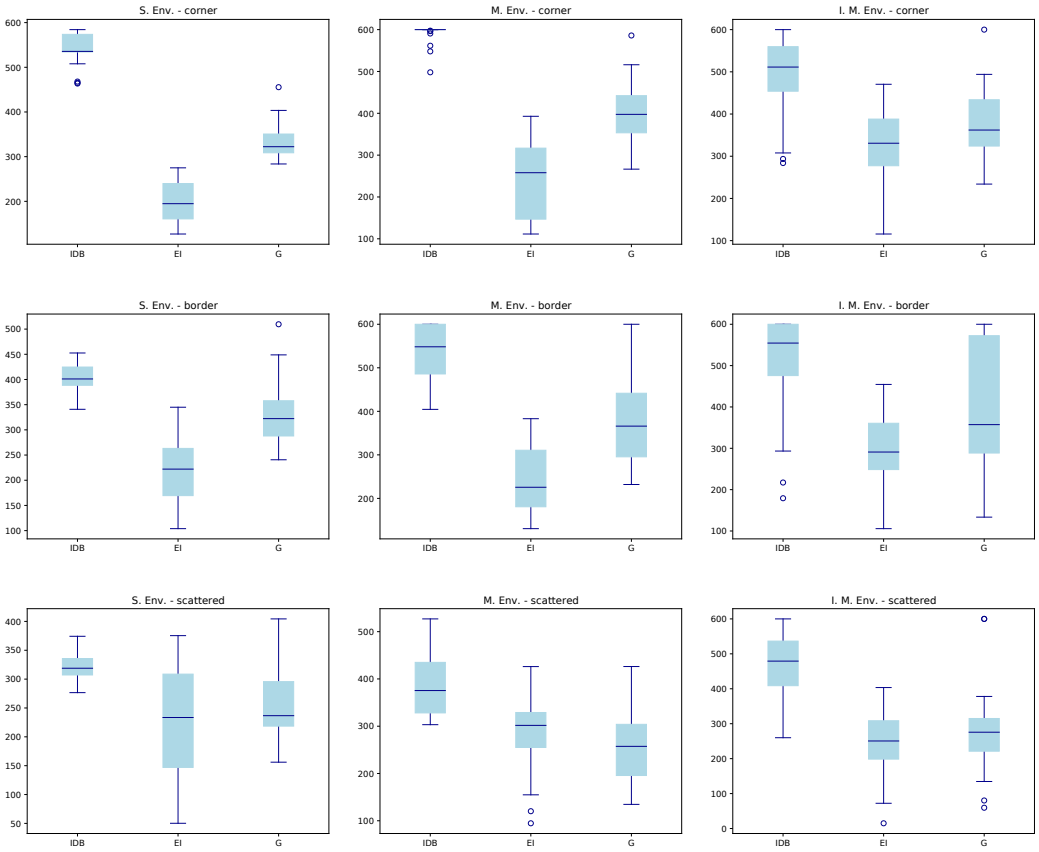


Figure 4.11: Duration of successful runs in validation for the S. (left column) M. (centre column) and I. M. environments, with the corner (top row), border (centre) and scattered (bottom) start regions.

Table 4.15: Best parameters for Infotaxis

Approach	Env.	Start region	D_f	R	τ
IDB	S.	-	6.563e-02	2.847e+01	5.002e+02
IDB	M.	-	5.577e-01	8.085e-00	6.180e+01
IDB	I. M.	-	1.054e-00	7.171e-01	2.376e+03
EI	S.	Corner	2.832e-01	9.115e+01	3.833e-00
EI	M.	Corner	6.089e-01	3.325e+02	5.180e-01
EI	I. M.	Corner	2.3404-00	2.195e-01	6.179e+02
EI	S.	Border	2.030e-02	1.552e-01	3.764e+01
EI	M.	Border	1.400e-01	2.151e+02	9.527e-00
EI	I. M.	Border	2.500e-00	1.286e-01	9.608e+02
EI	S.	Scattered	2.213e-02	1.0e-01	2.637e+01
EI	M.	Scattered	2.805e-01	3.281e+02	5.326e-01
EI	I. M.	Scattered	3.932e-02	1.884e+01	4.988e+01

Table 4.16: P-values of the Kolmogorov-Smirnov test applied to the success rates

St. region	Env.	IDB	EI	G
Corner	S.	1.02e-20	2.67e-06	3.32e-05
Corner	M.	1.44e-20	1.61e-05	1.77e-04
Corner	I. M.	1.28e-08	8.16e-05	2.88e-07
Border	S.	7.68e-05	2.38e-06	9.44e-04
Border	M.	1.23e-05	6.49e-07	1.36e-04
Border	I. M.	1.42e-07	2.23e-07	1.25e-05
Scattered	S.	1.73e-06	1.30e-05	2.94e-05
Scattered	M.	8.16e-10	2.01e-05	3.00e-05
Scattered	I. M.	1.80e-03	2.46e-05	1.70e-06

However, in the I. M. environment, there is little difference between EI and G. Applying the Kolmogorov-Smirnov test (Table 4.19) it can be seen that none of the data can be considered to follow normal distributions. As a result, the Friedman's Anova is applied (Table 4.20), showing that there are statistically significant differences in all scenarios. As a result, the Wilcoxon test is applied to perform pairwise comparisons (Table 4.21), with the Bonferroni correction being used to adjust the significance value to 1.67e-02. Its results show that most comparisons contain statistically significant differences. The exceptions are when comparing EI and G in the I. M. Env. with the corner and scattered start regions, and also in the S. environment with the scattered start region.

To assess the reason behind the different performances, we plot the trajectories made by the best individual of each approach in the first validation run (Figure 4.12). As can be seen, G tends to either perform zigzag or spiral motions to find or reacquire the plume, proceeding to move straight upwind when in the presence

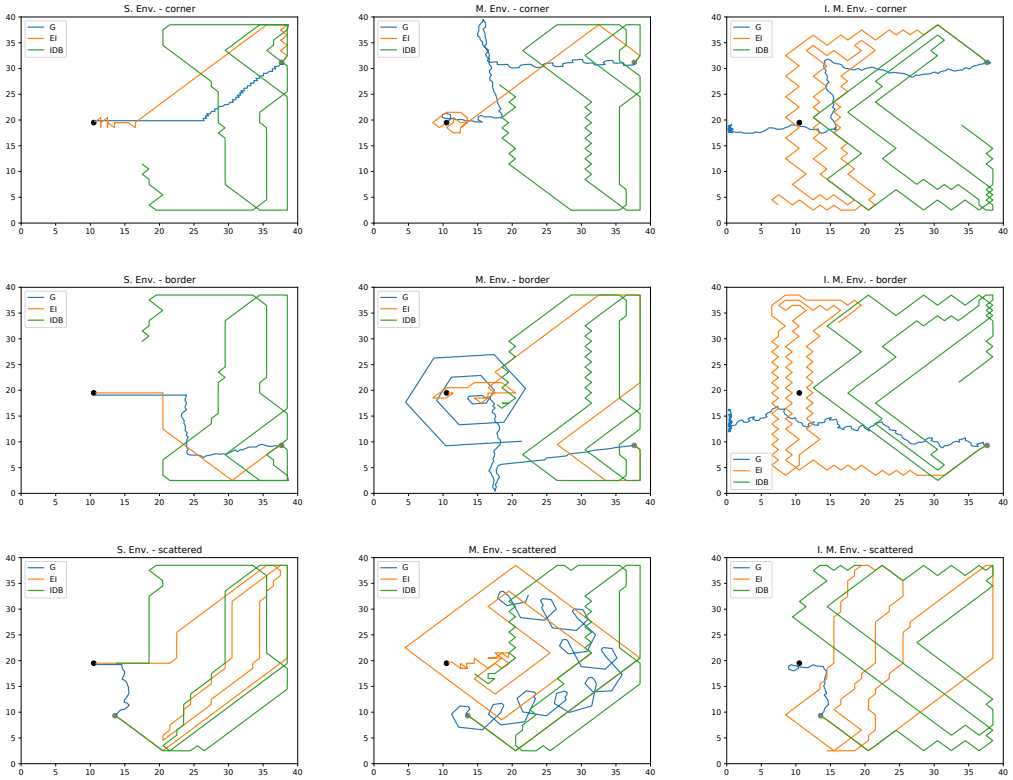


Figure 4.12: Trajectories of successful runs in validation for the S. (left column) M. (centre column) and I. M. environments, with the corner (top row), border (centre) and scattered (bottom) start regions.

Table 4.17: P-values of the Friedman's Anova applied to the success rates

St. region	S. Env.	M. Env.	I. M. Env.
Corner	1.10e-05	1.51e-10	1.45e-04
Border	3.37e-02	6.64e-12	3.76e-07
Scattered	2.21e-04	3.86e-08	3.13e-06

Table 4.18: P-values of the Wilcoxon test applied to the success rates

St. region	Functions	S. Env.	M. Env	I. M. Env.
Corner	IDB - EI	4.97e-05	1.68e-06	3.85e-04
Corner	IDB - G	8.81e-06	2.53e-06	9.84e-03
Corner	EI - G	**3.93e-01**	**3.61e-01**	4.55e-03
Border	IDB - EI	4.71e-03	1.69e-06	1.11e-05
Border	IDB - G	**1.33e-01**	2.95e-06	**7.81e-02**
Border	EI - G	**6.40e-02**	6.07e-04	5.50e-05
Scattered	IDB - EI	**5.83e-02**	1.71e-06	4.35e-05
Scattered	IDB - G	2.79e-06	9.36e-05	9.70e-05
Scattered	EI - G	**1.78e-01**	1.19e-03	**8.02e-01**

of odour. In turn, EI tends to make long straight motions, covering large portions of the environment and typically also moving upwind when sensing odour. Still, there are scenarios (S. Env. with the corner start region and M. Env. with the scattered start region) where the robot makes extra motions in other directions to refine its probability map. Finally, IDB tends to make a more thorough search of the environment, with less spacing between passes, which often causes it to run out of time quite far from the location of the source.

This results partially support the existing literature, in stating that Infotaxis' is very dependent of its parameters. However, EI shows that the parameters that better match the environment are often not the ones that lead to the best performance in locating the source. On the other hand, the controllers evolved by G attain significantly lower success rates than EI in four out of nine scenarios, and are also slower (in terms of control steps) than EI in six scenarios. However, the success rates of their controllers are often on par with each other, with the best individual of G performing better than the one of EI in some instances. Moreover, the tree-based controllers have much less computational overhead than those of Infotaxis, with each control step running approximately 7 times faster.

4.4 Genetic Programming Infotaxis

Having used EAs to evolve source seeking strategies from the ground-up (Section 4.2) and also for optimising the parameters of a probabilistic approach (Section 4.3) we hypothesised that the combination of both methods could produce bet-

Table 4.19: P-values of the Kolmogorov-Smirnov test applied to the duration of successful runs

St. region	Env.	IDB	EI	G
Corner	S.	8.57e-04	1.78e-05	3.06e-07
Corner	M.	7.81e-19	2.13e-04	2.32e-06
Corner	I. M.	5.06e-06	4.17e-04	3.15e-05
Border	S.	8.00e-05	4.20e-04	9.11e-06
Border	M.	2.17e-06	2.49e-05	1.02e-04
Border	I. M.	1.17e-12	2.24e-05	5.68e-07
Scattered	S.	8.40e-05	6.49e-06	4.71e-07
Scattered	M.	1.93e-06	1.09e-04	1.23e-04
Scattered	I. M.	9.01e-05	1.16e-04	1.74e-03

Table 4.20: P-values of the Friedman's Anova applied to the duration of successful runs

St. region	S. Env.	M. Env.	I. M. Env.
Corner	9.36e-14	3.00e-12	1.09e-06
Border	2.53e-11	3.07e-09	2.50e-05
Scattered	1.67e-06	2.42e-07	5.39e-07

ter results, i.e., that evolution could find a tree-based controller which made use of Infotaxis only under specific conditions, combining the strengths of both approaches. The motivation for this experiment is three-fold: (1) producing search strategies with overall better performance (success rate); (2) produce more efficient strategies by only using Infotaxis when it is needed; and (3) by analysing the evolved controllers, understanding in which situations it is more beneficial to use Infotaxis or simpler behaviours. As a result, in this section we propose GPInfotaxis (GPI), a method based on Geometric Syntactic Genetic Programming to evolve search strategies that combines bio-inspired building blocks (i.e., perceptions and behaviours) with Infotaxis. This is done by including an additional symbol *Infotaxis* into the terminal set of GSynGP, which commands the robot to make a single infotactic movement. In order for the infotactic behaviour to work, the probability map for the location of the odour source must be updated in the background during the robot's operation. In Infotaxis, the probability map is required to select each movement and thus must be constantly updated, leading to a high computational cost. Conversely, as GPInfotaxis only requires the probability map for the infotactic behaviour, it can be updated periodically. The update frequency is an adjustable parameter, leading to a trade-off between computational expense and performance. In this work, it is set to 0.5 Hz, i.e., every four simulation steps. Based on the findings of the previous section, the parameters of Infotaxis' gas parameter model are optimised by EI. It is important to note that all *Infotaxis* nodes use the same parameters. While it would be interesting to allow evolution to parametrise each *Infotaxis* node whithin a

Table 4.21: P-values of the Wilcoxon test applied to the duration of successful runs

St. region	Functions	S. Env.	M. Env	I. M. Env.
Corner	IDB - EI	1.73e-06	1.73e-06	6.98e-06
Corner	IDB - G	1.73e-06	1.73e-06	2.83e-04
Corner	EI - G	1.73e-06	9.32e-06	**1.75e-02**
Border	IDB - EI	1.73e-06	1.73e-06	8.47e-06
Border	IDB - G	1.15e-04	6.98e-06	8.61e-03
Border	EI - G	2.60e-06	1.89e-04	1.32e-02
Scattered	IDB - EI	6.32e-05	2.60e-05	5.22e-06
Scattered	IDB - G	2.84e-05	7.69e-06	8.19e-05
Scattered	EI - G	**2.21e-01**	1.47e-01	**3.49e-01**

GPInfotaxis individual, that would imply that each *Infotaxis* node would have its own probability map. As a result, the computational cost (both time and memory) would grow linearly with the amount of *Infotaxis* nodes included in each tree, making the approach infeasible.

4.4.1 Experimental results

Similarly to previous sections, thirty independent trials were made for each algorithm and each possible combination of environment and start region. The parameters presented in Section 4.2 were also used for these experiments, with the addition of the *Infotaxis* node to the terminal set. For each scenario, the *Infotaxis* nodes were parametrised with the corresponding best values found by EI, as presented on Table 4.15. Afterwards, a validation step took place where the resulting solutions were re-evaluated on the thirty instances of each scenario. The success rates attained in validation are plotted in Figure 4.13. As can be seen, GPI attains higher success rates than its counterparts in all settings, with its best individuals reaching at least 80 % in all scenarios except the most difficult one (I. M. Env.-Scattered).

To be able to draw more robust conclusions, we proceed to the statistical analysis, where the Kolmogorov test (Table 4.22) shows that none of the data can be considered to follow normal distributions. As a result, the Friedman's Anova is applied (Table 4.23), showing that there are statistically significant differences in all scenarios except the meandering environment with the scattered start region. We thus apply Wilcoxon (Table 4.24) test to perform pairwise comparisons in all scenarios where the Friedman's Anova found significant differences, being the Bonferroni correction used to adjust the significance value to 1.67e-02. Its results show that GPI produces significantly higher success rates than G in all scenarios, existing a single case (stable environment with the border start region) where it does not significantly outperform EI, producing equivalent success rates. Furthermore, the Wilcoxon test shows that EI significantly outperforms G in five scenarios, performing equivalently in the remaining ones.

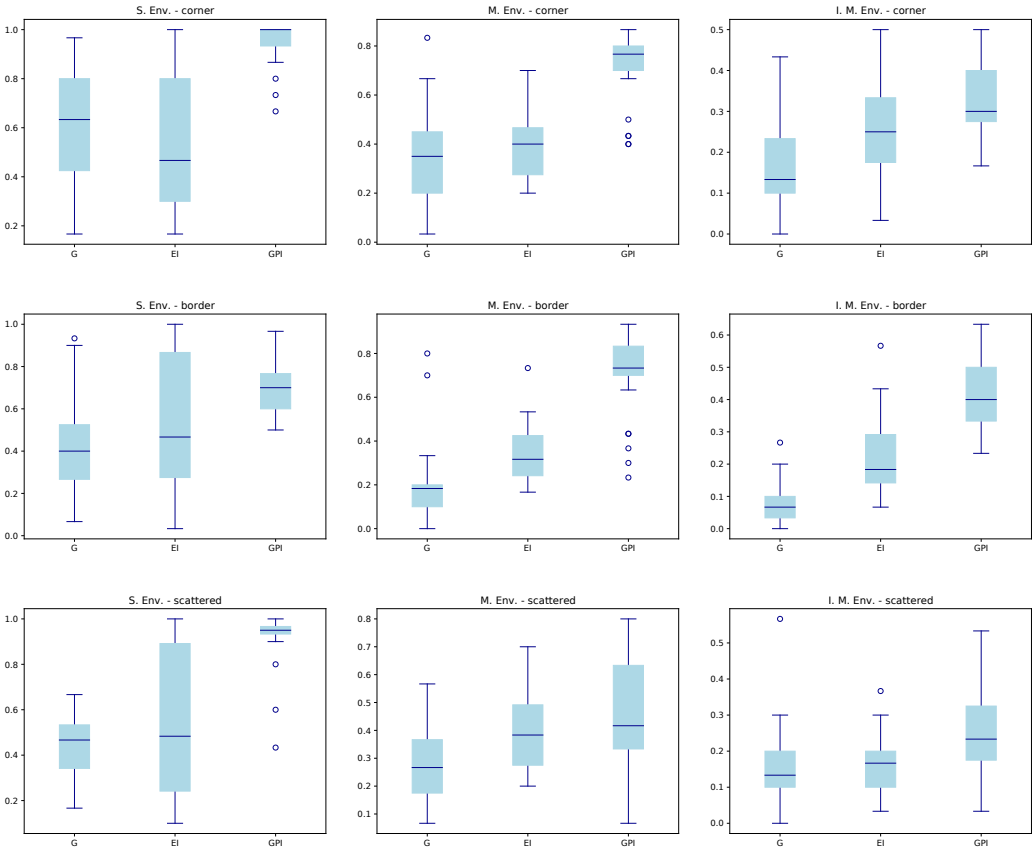


Figure 4.13: Boxplots of the success rates in validation for the S. (left column) M. (centre column) and I. M. environments, with the corner (top row), border (centre) and scattered (bottom) start regions.

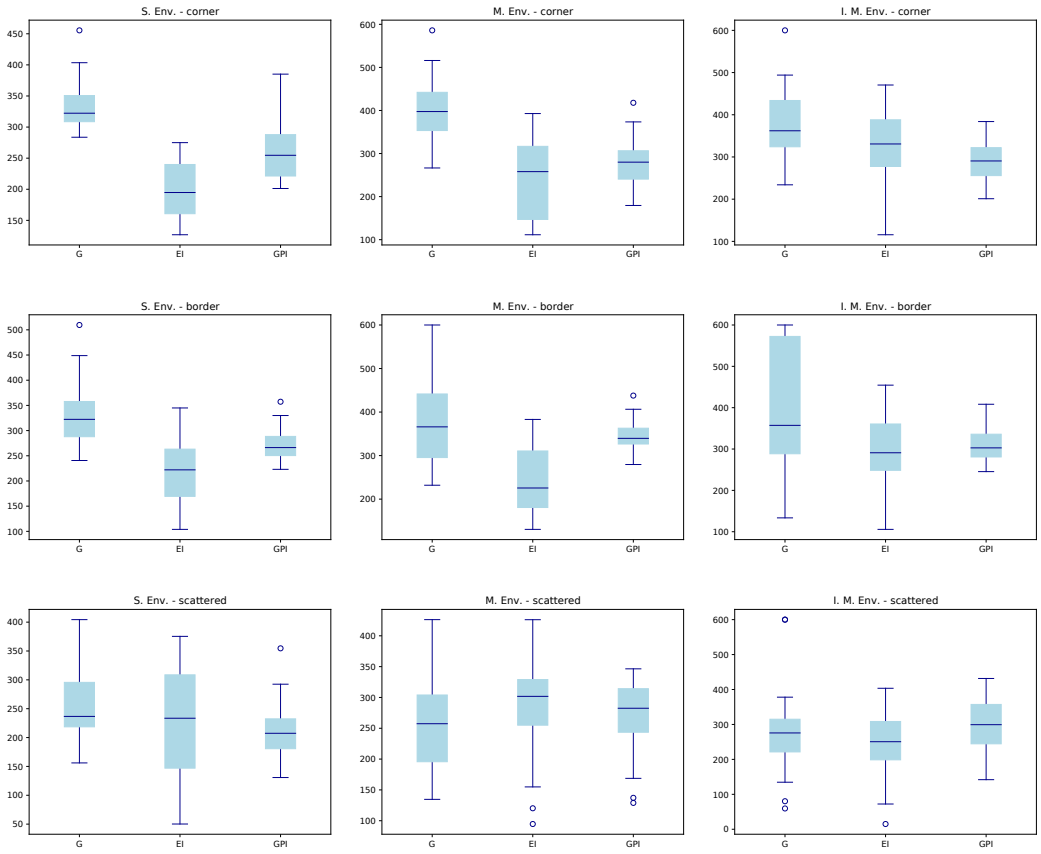


Figure 4.14: Boxplots of the durations of successful runs in validation for the S. (left column) M. (centre column) and I. M. environments, with the corner (top row), border (centre) and scattered (bottom) start regions.

Table 4.22: P-values of the Kolmogorov-Smirnov test applied to the validation success rates

St. region	Env.	G	EI	GPI
Corner	S.	3.32e-05	2.67e-06	2.23e-09
Corner	M.	1.77e-04	1.61e-05	8.53e-10
Corner	I. M.	2.88e-07	8.16e-05	1.62e-07
Border	S.	9.44e-04	2.38e-06	8.83e-05
Border	M.	1.36e-04	6.49e-07	2.96e-05
Border	I. M.	1.25e-05	2.23e-07	3.01e-06
Scattered	S.	2.94e-05	1.30e-05	1.02e-09
Scattered	M.	3.00e-05	2.01e-05	3.80e-06
Scattered	I. M.	1.70e-06	2.46e-05	7.24e-07

Table 4.23: P-values of the Friedman's Anova applied to the validation success rates

St. region	S. Env.	M. Env.	I. M. Env.
Corner	1.77e-08	7.14e-10	6.39e-05
Border	1.97e-04	2.67e-10	2.41e-11
Scattered	1.26e-07	**5.34e-02**	4.07e-03

Having assessed the success rates of each approach, we now focus on the time needed to locate the chemical source in the successful evaluations, which are plotted in Figure 4.14. As can be seen, EI produces the fastest searches in the S. and M. environments with the corner start region. However, other scenarios, the approaches seem to take more similar amounts of time. Once again, a statistical analysis is conducted to draw robust conclusions. The Kolmogorov-Smirnov test is first applied (Table 4.25) showing that none of the data can be considered to follow normal distributions. As a result, the Friedman's Anova is applied (Table 4.26), showing that there are statistically significant differences between the three approaches in six scenarios. The Wilcoxon test is then applied to perform pairwise comparisons in the scenarios identified (Table 4.27), being the Bonferroni correction used to adjust the significance value to 1.67e-02. Its results show that GPI consistently produces faster searches than G in the six remaining scenarios. However, EI is significantly faster than GPI in three scenarios, performing equivalently in the remaining ones. In turn, EI is faster than G in four scenarios, requiring equivalent amounts of time in the remaining ones to locate the chemical source.

4.4.2 Analysis of the best search strategies

In order to gain insight regarding when Infotaxis is more useful, this section analyses the best evolved controller by GPI for each scenario (Algorithms 4.1 to 4.9). Note that for the sake of clarity, some controllers had introns removed.

Table 4.24: P-values of the Wilcoxon test applied to the validation success rates

St. region	Functions	S. Env.	M. Env	I. M. Env.
Corner	G - EI	**3.93e-01**	**3.61e-01**	4.55e-03
Corner	G - GPI	5.51e-06	2.01e-06	1.40e-05
Corner	EI - GPI	4.42e-06	2.52e-06	1.67e-02
Border	G - EI	**6.40e-02**	6.07e-04	5.50e-05
Border	G - GPI	2.06e-05	1.89e-06	1.70e-06
Border	EI - GPI	**6.87e-02**	4.77e-06	2.37e-05
Scattered	G - EI	**1.78e-01**	-	**8.02e-01**
Scattered	G - GPI	1.72e-06	-	1.52e-03
Scattered	EI - GPI	5.47e-05	-	2.20e-03

Table 4.25: P-values of the Kolmogorov-Smirnov test applied to the duration of successful runs

St. region	Env.	G	EI	GPI
Corner	S.	3.06e-07	1.78e-05	5.78e-05
Corner	M.	2.32e-06	2.13e-04	1.05e-04
Corner	I. M.	3.15e-05	4.17e-04	1.71e-05
Border	S.	9.11e-06	4.20e-04	1.89e-05
Border	M.	1.02e-04	2.49e-05	1.44e-04
Border	I. M.	5.68e-07	2.24e-05	1.01e-04
Scattered	S.	4.71e-07	6.49e-06	1.76e-04
Scattered	M.	1.23e-04	1.09e-04	4.52e-04
Scattered	I. M.	1.74e-03	1.16e-04	1.01e-03

Still, the property of GSynGP holds for GPI controllers, as they original had a mean size of approximately 13.2 nodes, and after simplified their mean size is approximately of 8.56 nodes. Moreover, some controllers had no introns to start with.

At a glance, it is interesting to note that all controllers include at least one *Infotaxis* node. Moreover, the infotactic behaviours tend be applied to find or reacquire the chemical plume. The controller evolved for the stable environment with the corner start region (Algorithm 4.1) starts by checking if the robot is currently sensing odour (line 1) and, if so, moves twice upwind (respectively by 0.5 m in line 2 and by 0.237 m in line 3) followed by moving randomly for 0.65 m (line 4). If the robot is not currently sensing odour, it resorts to Infotaxis to either find or reacquire the plume (line 6).

The strategy evolved for the stable environment with the border start region (Algorithm 4.2) is the most complex of all. Still, it starts by checking if odour is currently being sensed (line 1) and, if so, moves upwind for 0.234 m (line 2). After this motion, if the robot has sensed odour in the past 5 s, it moves

Algorithm 4.1: Best strategy evolved by GPInfotaxis in the stable environment with the corner start region.

```
1  if SO() then
2      moveUpwind(d = 0.5)
3      moveUpwind(d = 0.237)
4      moveRandom(d = 0.65)
5  else
6      Infotaxis()
```

Algorithm 4.2: Best strategy evolved by GPInfotaxis in the stable environment with the border start region.

```
1  if SO() then
2      moveUpwind(d = 0.234)
3      if HSO(t = 5) then
4          moveUpwind(d = −0.105)
5      else
6          wanderUpwind()
7          if SO() then
8              spiral(dis_inc = 0.125, iters = 4, term = PL(t = 30))
9          else
10             spiral(dis_inc = 0.102, iters = 3, term = HSO(t = 30))
11 else
12     if HSO(t = 31) then
13         moveUpwind(d = 0.25)
14     else
15         moveUpwind(d = 0.837)
16 if HSO(t = 1) then
17     wanderUpwind()
18 else
19     Infotaxis()
20     if SO() then
21         wanderDownwind()
22     else
23         Infotaxis()
24         Infotaxis()
25         moveUpwind(d=0.5)
26 Infotaxis()
```

Table 4.26: P-values of the Friedman's Anova applied to the duration of successful runs

St. region	S. Env.	M. Env.	I. M. Env.
Corner	1.09e-09	1.79e-06	5.53e-04
Border	9.87e-09	1.23e-03	**1.07e-01**
Scattered	1.19e-02	**5.31e-01**	**1.07e-01**

Table 4.27: P-values of the Wilcoxon test applied to the duration of successful runs

St. region	Functions	S. Env.	M. Env	I. M. Env.
Corner	G - EI	1.73e-06	9.32e-06	**1.75e-02**
Corner	G - GPI	1.49e-05	4.73e-06	1.06e-04
Corner	EI - GPI	8.94e-04	**1.31e-01**	**1.25e-01**
Border	G - EI	2.60e-06	1.89e-04	-
Border	G - GPI	3.72e-05	1.06e-01	-
Border	EI - GPI	3.88e-04	2.16e-05	-
Scattered	G - EI	**2.21e-01**	-	-
Scattered	G - GPI	1.29e-03	-	-
Scattered	EI - GPI	**5.04e-01**	-	-

downwind (note the minus sign in the step length) for 0.105 m (line 4). If not, the robot wanders upwind (line 6) and, depending on the detection of odour, either spirals four times halting if it has lost the plume for longer than 30 s (line 8) or three times until considering that it has sensed odour in the past 30 s (line 10). If the first check for odour detections returns false, the robot moves upwind (with the step length depending on whether odour was sensed in the past 31 s lines 12 to 15). Afterwards, the robot checks if it has sensed odour in the past second and, if so, wanders upwind (lines 16 and 17). Otherwise, it makes an infotactic motion and checks if that leads it to sense odour, in which case it wanders downwind (lines 19 to 21). If not, it makes two infotactic movements followed by moving straight upwind for 0.5 m (lines 23 to 25). Note that due to the size of the simulation grid cell size being 1 m, this may result in moving 2 m according to Infotaxis and 0.5 m upwind. Finally, regardless of the choices made, an infotactic motion is made (line 26).

The strategy evolved for the scattered start region (Algorithm 4.3) is much simpler than the previous one. The robot starts by checking for current odour detections (line 1). If it is sensing odour, it wanders downwind (for the predefined step length of 0.5 m) followed by an infotactic motion (lines 2 and 3), which may lead the robot 1 m in any direction. If no odour is detected, the robot performs an infotactic motion and checks if it has sensed odour in the past second (lines 5 and 6). If so, it moves randomly (line 7), otherwise, it performs Infotaxis once again (line 9). Interestingly, this strategy performs no explicit upwind motions

Algorithm 4.3: Best strategy evolved by GPInfotaxis in the stable environment with the scattered start region.

```
1 if SO() then
2     wanderDownwind()
3     Infotaxis()
4 else
5     Infotaxis()
6     if HSO(t = 1) then
7         moveRandom(d = 0.5)
8     else
9         Infotaxis()
```

when sensing odour, resorting to Infotaxis or the random movements to move the robot closer to the source when in contact with the plume. This may simply be due to insufficient evolution time, as replacing the wanderDownwind motion by moveUpwind is bound to lead to better results. Thankfully, due to the white-box nature of the controllers, such modification could easily be made.

Algorithm 4.4: Best strategy evolved by GPInfotaxis in the meandering environment with the corner start region.

```
1 if SO() then
2     wanderUpwind()
3 else
4     Infotaxis()
```

Algorithm 4.5: Best strategy evolved by GPInfotaxis in the meandering environment with the border start region.

```
1 if SO() then
2     wanderUpwind()
3     moveUpwind(d = 1.0)
4 else
5     Infotaxis()
```

The strategies evolved for the meandering environment with the corner and border start regions (Algorithms 4.4 and 4.5) are quite similar. They both start by checking if the robot is currently sensing odour and, if so, wander upwind (lines 1 and 2), with the border controller also making a moveUpwind motion of 1 m (line 3). Otherwise, they both resort to Infotaxis (respectively line 4 and 5). In turn, the controller evolved for the meandering environment with the scattered start region (Algorithm 4.6) starts by checking if the robot has ever sensed odour (note the time threshold in line 1 which is larger than the total evaluation time) and, if so, makes a wanderCrosswind motion followed by

Algorithm 4.6: Best strategy evolved by GPInfotaxis in the meandering environment with the scattered start region.

```
1 if HSO(t = 619) then
2     wanderCrosswind()
3     Infotaxis()
4 else
5     Infotaxis()
```

Infotaxis (lines 2 and 3). Otherwise, it performs simply Infotaxis (line 5). This is an odd strategy, introducing further exploration crosswind exploration after sensing odour, but it may help to prevent premature convergence of Infotaxis' probability map, caused by odour detections far from the plume's centreline. Also, it is worth remembering that this controller attains the highest success rate of all in this scenario.

Algorithm 4.7: Best strategy evolved by GPInfotaxis in the intermittent-meandering environment with the corner start region.

```
1  if HSO(t = 644.5) then
2      if SO() then
3          wanderUpwind()
4      else
5          moveUpwind(d = 2.215)
6  else
7      Infotaxis()
8      if HSO(t = 300) then
9          if SO() then
10             wanderUpwind()
11         else
12             moveUpwind(d = 0.25)
13     else
14         Infotaxis()
```

Finally, the controllers evolved for the I. M. environment (Algorithms 4.7 to 4.9) tend to employ the HSO perception rather than the SO, to test if odour has been sensed within a period of time rather than in the current instant. This can be seen as an adaptation to the intermittency of the chemical plume, as the *SO* perception is likely to return false in most instances. In fact, only the strategy evolved for the corner scenario (Algorithm 4.7) uses the SO perception, but it is used to arbitrate between similar behaviours. Using this search strategy, the robot starts by checking if it has ever sensed odour (note that the time threshold in line 1 is higher than the total evaluation time) and, if so, checks if it is currently sensing odour (line 2). If it is, it wanders upwind (line 3), if it is not, it moves

Algorithm 4.8: Best strategy evolved by GPInfotaxis in the intermittent-meandering environment with the border start region.

```
1 if HSO(t = 5) then
2     wanderUpwind()
3 else
4     spiral(dis_inc = 0.75, iters = 3, term = PL(t = 10))
5 if HSO(t = 30) then
6     spiral(dis_inc = 0.75, iters = 5, term = HSO(t = 30))
7 else
8     Infotaxis()
```

Algorithm 4.9: Best strategy evolved by GPInfotaxis in the intermittent-meandering environment with the scattered start region.

```
1 if HSO(t = 34) then
2     moveUpwind(d = 3.229)
3 else
4     Infotaxis()
```

straight upwind for 2.215 m (line 5). If the robot has yet to sense odour, it makes an infotactic motion and checks again for recent odour detections (lines 7 and 8). If it is currently sensing odour, it wanders upwind (lines 9 and 10). Otherwise, if it has sensed odour in the past 300 s, it moves upwind for 0.25 m (line 12). If it has not sensed odour in the past 300 s, it performs another infotactic motion (line 14). It should be noted that the upwind motions once the robot has sensed odour (lines 2 to 5) may enable it to cope better with the intermittency of the plume, but may also cause it to be mislead by its meandering and move past the location of the chemical source, failing to locate it. One possible fix would be to reduce the threshold of the HSO perception in line 1, to enable the robot to resort back to Infotaxis if odour is not sensed for a long time.

The controller evolved for the border scenario 4.8 starts by checking if odour has been sensed in the last 5 s and, if so, commands the robot to move upwind (lines 1 and 2). Otherwise, it performs a spiral motion, halting if the plume has not been sensed for longer than 10 s (line 4). It then checks if odour has been sensed in the past 30 s and, if so, commands the robot to remain still (note that the termination condition of the spiral in line 6 is the same as the **If** clause in line 5), which may cause it to sense odour again, provided that it is close to the plume's centerline. Otherwise, it performs Infotaxis (line 8).

Finally, the controller evolved for scattered start region 4.9 simply checks if odour has been sensed in the past 34 s and, if so, moves upwind for 3.229 m (lines 1 and 2). Otherwise, it performs Infotaxis (line 4). This controller makes the robot make large upwind motions when sensing odour, which are bound to help it cope with the plume's intermittency. By simultaneously updating Infotaxis'

probability map in the background, the robot is also able to resort back to infotactic behaviours when the upwind strides fail to locate the chemical source and make it lose the chemical plume for longer than 34 s.

4.4.3 Final remarks

To sum up, the evolution of tree based controllers with infotactic behaviours, GPI, produces higher success rates than EI and G, while being at least as fast as G (equivalent amounts of control steps). However, it still has some of the computational overhead of Infotaxis, which despite being reduced by scarcer updates, it still makes it unfeasible to be applied in low-power microcontrollers. Analysing the best evolved controller for each scenario, it can be seen that Infotaxis is particularly useful for finding and reacquiring the chemical plume, with the bio-inspired behaviours being preferred to track it to its source.

Chapter 5

Multi-robot and swarm approaches for odour source localisation

Contents

In the previous chapter we proposed different evolutionary approaches, with increasing degrees of complexity, to produce robotic controllers for single-robot applications which culminated into the ability to solve the task with success rates ranging from 50% (in the hardest scenario) to 100% (in the easiest one). This chapter attempts to improve the performance by using multiple robots, controlled by simple tree-based controllers. Such approaches have clear advantages, such as robustness to the failure of some robots and also the ability to employ low cost robots. We aim to provide answers for the two last research questions, specifically, in Section 5.1 we investigate the effects of using many individual robots and, in Section 5.2 we assess the gains attained by cooperating. Finally, in Section 5.3 we compare the performance of the approaches evolved by GSynGP for one and many cooperative robots to those of the most complex single robot approach (GPI), to answer the question of whether is it better to have many simple robots or a single but complex one.

5.1 Influence of the number of robots

The first approach is a naive one. We simply take the best controllers evolved by GSynGP with $F_{2,3}$ for a single robot (G_1) and deploy them on groups of three (G_{1to3}), five (G_{1to5}) and ten (G_{1to10}) robots, to assess whether having many agents operating individually is enough to increase the performance. As before, the evolved controllers undergo a validation step where each is evaluated in thirty instances of the corresponding scenario. The success rates attained in validation are plotted in Figure 5.1. As can be seen, increasing the number of robots consistently leads to an increase of the success rates in all scenarios. However, even in the simplest scenario, there is a controller that using ten robots only manages a 60% success rate, so there is room for improvement. In order to draw more robust conclusions, we proceed to the statistical analysis. The Kolmogorov-Smirnov test is first applied (Table 5.1) with its results showing that none of the data can be considered to follow normal distributions. As a result, the Friedman's Anova is applied (Table 5.2), which in turn shows that there are statistically significant differences between the success rates of the approaches in all scenarios. As such, we proceed to apply the Wilcoxon test (Table 5.3), with the Bonferroni correction being used to adjust the significance value to 8.33e-03. Its results show that all comparisons are significantly different, verifying that increasing the number of robots leads to significantly higher success rates in all scenarios.

Having analysed the success rates, we now turn to the duration of the successful evaluations, which are plotted in Figure 5.2. As can be seen, the approaches do not perform very differently, but, in most scenarios, there is a slight reduction of the search times with the increase of the number of robots, that is more emphasized in the stable and meandering environments with the scattered start region. In order to be able to draw more robust conclusions, we proceed to the statistical analysis, starting with the Kolmogorovs-Smirnov test (Table 5.4). Its results show that none of the data can be considered to follow normal distributions and, as a result, we apply the Friedman's Anova to assess the

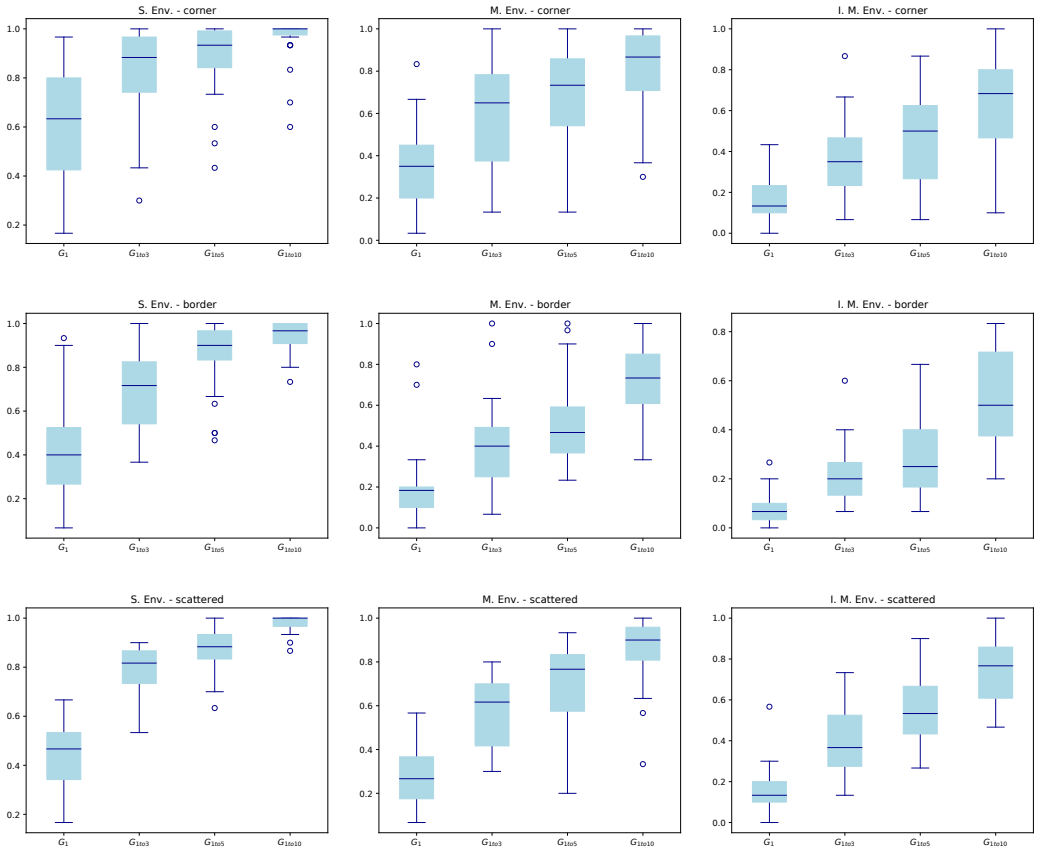


Figure 5.1: Boxplots of the success rates in validation for the S. (left column) M. (centre column) and I. M. environments, with the corner (top row), border (centre) and scattered (bottom) start regions.

Table 5.1: P-values of the Kolmogorov-Smirnov test applied to the success rates

St. region	Env.	G_1	G_{1to3}	G_{1to5}	G_{1to10}
Corner	S.	3.32e-05	4.61e-07	1.11e-08	2.23e-16
Corner	M.	1.77e-04	3.78e-06	1.42e-06	3.27e-08
Corner	I. M.	2.88e-07	1.15e-04	1.52e-04	1.65e-06
Border	S.	9.44e-04	6.34e-06	3.05e-10	7.99e-13
Border	M.	1.36e-04	9.44e-05	8.57e-07	9.36e-05
Border	I. M.	1.25e-05	1.32e-04	1.11e-06	1.05e-04
Scattered	S.	2.94e-05	2.20e-07	6.91e-06	3.53e-15
Scattered	M.	3.00e-05	1.04e-06	3.72e-08	1.59e-09
Scattered	I. M.	1.70e-06	2.46e-06	9.36e-05	5.09e-05

Table 5.2: P-values of the Friedman's Anova applied to the success rates

St. region	S. Env.	M. Env.	I. M. Env.
Corner	1.40e-17	1.65e-17	3.56e-18
Border	1.68e-17	1.00e-17	1.78e-17
Scattered	4.50e-18	1.29e-18	1.84e-18

Table 5.3: P-values of the Wilcoxon test applied to the success rates

St. region	Functions	S. Env.	M. Env	I. M. Env.
Corner	G_1 - G_{1to3}	1.70e-06	1.72e-06	4.92e-06
Corner	G_1 - G_{1to5}	1.72e-06	1.72e-06	1.73e-06
Corner	G_1 - G_{1to10}	1.72e-06	1.71e-06	1.71e-06
Corner	G_{1to3} - G_{1to5}	8.41e-04	3.18e-05	2.55e-05
Corner	G_{1to3} - G_{1to10}	7.56e-06	5.57e-06	1.72e-06
Corner	G_{1to5} - G_{1to10}	3.71e-05	1.43e-05	5.54e-06
Border	G_1 - G_{1to3}	1.85e-06	2.82e-06	3.25e-06
Border	G_1 - G_{1to5}	1.64e-06	1.71e-06	1.70e-06
Border	G_1 - G_{1to10}	1.71e-06	1.71e-06	1.71e-06
Border	G_{1to3} - G_{1to5}	8.83e-06	1.19e-04	4.54e-03
Border	G_{1to3} - G_{1to10}	3.72e-06	2.53e-06	1.73e-06
Border	G_{1to5} - G_{1to10}	5.54e-05	3.13e-06	2.53e-06
Scattered	G_1 - G_{1to3}	1.72e-06	1.72e-06	1.72e-06
Scattered	G_1 - G_{1to5}	1.67e-06	1.70e-06	1.71e-06
Scattered	G_1 - G_{1to10}	1.69e-06	1.63e-06	1.70e-06
Scattered	G_{1to3} - G_{1to5}	4.13e-05	1.74e-05	1.92e-05
Scattered	G_{1to3} - G_{1to10}	1.64e-06	1.71e-06	1.71e-06
Scattered	G_{1to5} - G_{1to10}	7.80e-06	1.64e-06	1.70e-06

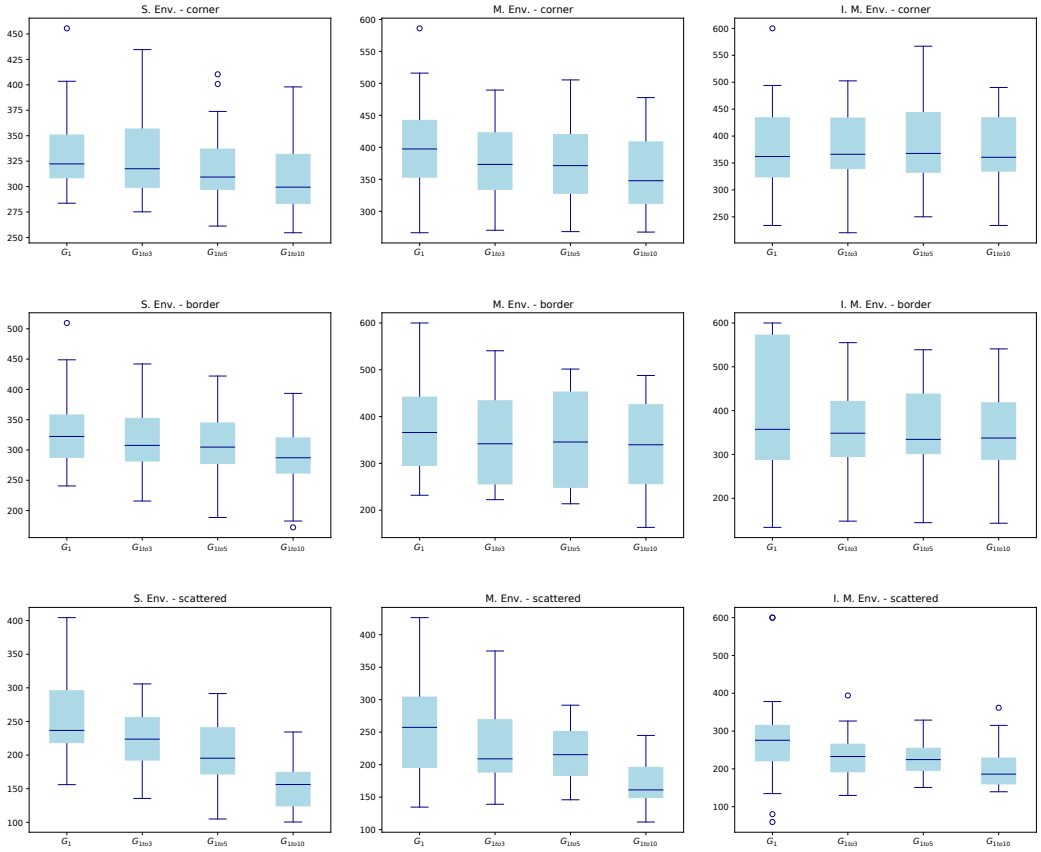


Figure 5.2: Boxplots of the durations of successful runs in validation for the S. (left column) M. (centre column) and I. M. environments, with the corner (top row), border (centre) and scattered (bottom) start regions.

existance of statistically significant differences between the search times of the four approaches in each scenario. Its results (Table 5.5) show that there are statistically significant differences in all scenarios, except the I.M. environment with the corner and border start regions. As a result, we apply the Wilcoxon test in all other scenarios (Table 5.6). Its results show that there are statistically significant differences in most comparisons, implying that, generally, using more robots leads to finding the chemical source faster. The exceptions are when using more than three robots in the meandering environment with the border start region, or in the intermittent-meandering environment with the scattered start region. Moreover, using the scattered start region, there are no differences in using one or three robots in the meandering and intermittent-meandering environments and there are also no differences when using three or five robots in the stable and meandering environments. The larger differences in the search times are to be expected, as with the use of more robots it is more likely that at least one of them starts from a location close to the plume, and thus only needs to track it to the source.

In conclusion, using increasing the number of robots used always leads to better success rates, but may not lead to faster searches in some scenarios. In the following section, we investigate whether enabling the robots to cooperate provides any improvements in success rates or search times.

Table 5.4: P-values of the Kolmogorov-Smirnov test applied to the duration of successful runs

St. region	Env.	G_1	G_{1to3}	G_{1to5}	G_{1to10}
Corner	S.	3.06e-07	6.44e-07	7.04e-07	1.09e-07
Corner	M.	2.32e-06	1.60e-04	2.08e-04	7.79e-05
Corner	I. M.	3.15e-05	3.85e-05	4.53e-06	2.95e-04
Border	S.	9.11e-06	1.26e-04	1.28e-03	2.69e-03
Border	M.	1.02e-04	1.94e-05	9.18e-05	9.21e-05
Border	I. M.	5.68e-07	1.98e-04	6.43e-05	1.51e-04
Scattered	S.	4.71e-07	3.36e-05	1.14e-04	4.79e-07
Scattered	M.	1.23e-04	1.42e-07	4.53e-04	2.28e-06
Scattered	I. M.	1.74e-03	3.80e-05	9.72e-05	3.49e-07

Table 5.5: P-values of the Friedman's Anova applied to the duration of successful runs

St. region	S. Env.	M. Env.	I. M. Env.
Corner	2.15e-11	8.69e-12	**5.37e-02**
Border	2.06e-09	2.75e-03	**6.87e-01**
Scattered	1.40e-13	7.14e-06	2.11e-03

Table 5.6: P-values of the Wilcoxon test applied to the duration of successful runs

St. region	Functions	S. Env.	M. Env	I. M. Env.
Corner	G_1 - G_{1to3}	7.52e-02	3.61e-03	-
Corner	G_1 - G_{1to5}	1.48e-04	1.49e-05	-
Corner	G_1 - G_{1to10}	5.22e-06	1.92e-06	-
Corner	G_{1to3} - G_{1to5}	3.59e-04	4.11e-03	-
Corner	G_{1to3} - G_{1to10}	3.52e-06	7.69e-06	-
Corner	G_{1to5} - G_{1to10}	1.71e-03	2.83e-04	-
Border	G_1 - G_{1to3}	1.75e-02	5.67e-03	-
Border	G_1 - G_{1to5}	1.04e-02	8.94e-04	-
Border	G_1 - G_{1to10}	2.16e-05	4.86e-05	-
Border	G_{1to3} - G_{1to5}	4.68e-03	**6.00e-01**	-
Border	G_{1to3} - G_{1to10}	4.29e-06	**2.43e-02**	-
Border	G_{1to5} - G_{1to10}	2.88e-06	**6.56e-02**	-
Scattered	G_1 - G_{1to3}	7.73e-03	**9.37e-02**	**2.29e-01**
Scattered	G_1 - G_{1to5}	4.90e-04	2.26e-03	9.27e-03
Scattered	G_1 - G_{1to10}	1.73e-06	2.37e-05	1.29e-03
Scattered	G_{1to3} - G_{1to5}	**1.04e-02**	**1.06e-01**	**4.17e-01**
Scattered	G_{1to3} - G_{1to10}	1.73e-06	1.97e-05	**5.19e-02**
Scattered	G_{1to5} - G_{1to10}	2.88e-06	2.16e-05	**4.28e-02**

5.2 The role of cooperation

In the previous section we assessed whether using more robots improved the performance of the search. In this section, we explore the benefits of cooperation. We enable GSynGP to evolve cooperating strategies by adding a perception and two behaviours respectively to its function and terminal sets:

Perceptions

- *Neighbours(n_t, r)* - which informs whether there is a neighbour of type n_t ($n_t \in$ {all, sensing odour (so), not sensing odour (nso)}) within a neighbourhood of radius r ($r \in$ [1.5, 40] meters);

Behaviours

- *moveTowards(n_t, d, r)* - moves the robot d meters (d $\in$ {0.25, 0.5, 1.0} meters) towards the centre of mass of the neighbours of type n_t ($n_t \in$ {all, sensing odour (so), not sensing odour (nso)}) within a neighbourhood of radius r ($r \in$ [1.5, 40] meters). If no robots of the specified type are within range, the robot is commanded to remain still;
- *moveAway(n_t, d, r)* - moves the robot d meters (d $\in$ {0.25, 0.5, 1.0} meters) away from the centre of mass of the neighbours of type n_t ($n_t \in$

{all, sensing odour (so), not sensing odour (nso)}) within a neighbourhood of radius r ($r \in [1.5, 40]$ meters). If no robots of the specified type are within range, the robot is commanded to remain still;

These three symbols should be able to provide interesting cooperation between the robots without being too complex. Using the *Neighbours* perception, the robots can make decisions based on the existence of neighbours of a given type in a parametrizable neighbourhood. In turn, using the *moveTowards* or *moveAway* behaviours the robots can respectively move closer or away from neighbours of a given type (e.g., a robot can move closer to neighbours sensing odour or robots not sensing odour could scatter in the environment to increase their odds in sensing the plume). Also, by combining both behaviours along with a moveUpwind behaviour, the robots could possible track a chemical plume in formation. We shall refer to these approaches as G_N, where N is the number of robots being used.

In order to further encourage the evolution of cooperation, an evaluation is now considered to be successful only when all robots reach the chemical source within the predefined time limit. Note that this only applies for evolution, as we consider a validation run to be successful as soon as one robot finds the odour source.

5.2.1 Experimental results

Table 5.7: P-values of the Kolmogorov-Smirnov test applied to the success rates

St. region	Env.	G_3	G_{1to3}	G_5	G_{1to5}	G_{10}	G_{1to10}
Corner	S.	1.18e-11	4.61e-07	1.69e-13	1.11e-08	1.50e-18	2.23e-16
Corner	M.	4.24e-06	3.78e-06	1.83e-06	1.42e-06	6.32e-07	3.27e-08
Corner	I. M.	1.69e-06	1.15e-04	2.05e-07	1.52e-04	1.63e-07	1.65e-06
Border	S.	1.73e-06	6.34e-06	3.07e-06	3.05e-10	3.57e-07	7.99e-13
Border	M.	1.19e-04	9.44e-05	5.04e-07	8.57e-07	1.12e-07	9.36e-05
Border	I. M.	5.19e-09	1.32e-04	2.50e-08	1.11e-06	2.82e-04	1.05e-04
Scattered	S.	3.64e-06	2.20e-07	6.33e-04	6.91e-06	7.88e-11	3.53e-15
Scattered	M.	1.06e-05	1.04e-06	1.80e-06	3.72e-08	3.63e-06	1.59e-09
Scattered	I. M.	3.31e-09	2.46e-06	1.21e-08	9.36e-05	2.12e-05	5.09e-05

Table 5.8: P-values of the Friedman's Anova applied to the success rates

St. region	S. Env.	M. Env.	I. M. Env.
Corner	7.41e-14	3.33e-08	4.80e-11
Border	1.24e-11	2.79e-14	1.85e-18
Scattered	5.49e-19	1.69e-13	3.63e-17

Once again, thirty independent runs are made for each combination of environment, start region and number of robots, being approaches compared through

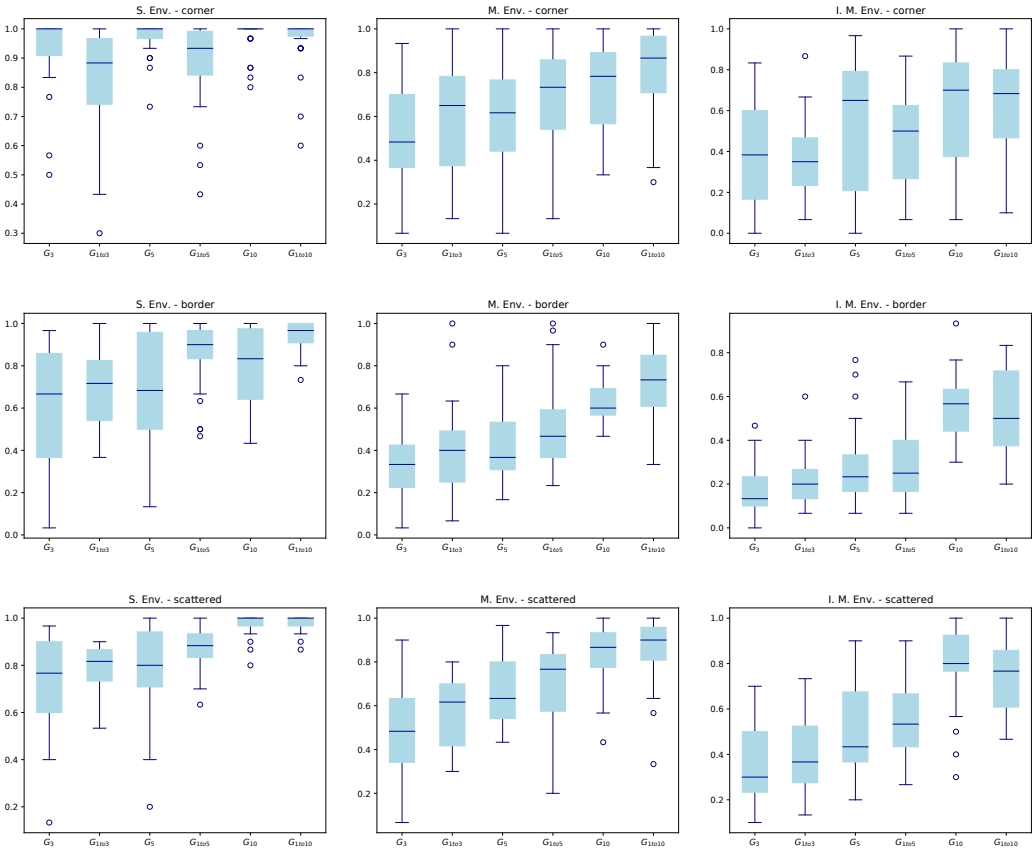


Figure 5.3: Boxplots of the validation success rates for the S. (left column) M. (centre column) and I. M. environments, with the corner (top row), border (centre) and scattered (bottom) start regions.

Table 5.9: P-values of the Wilcoxon test applied to the success rates

St. region	Functions	S. Env.	M. Env	I. M. Env.
Corner	G_3 - G_{1to3}	1.91e-03	**2.38e-01**	**3.71e-01**
Corner	G_5 - G_{1to5}	7.48e-04	**3.92e-02**	**2.34e-01**
Corner	G_{10} - G_{1to10}	**9.18e-01**	**4.23e-02**	**9.82e-01**
Border	G_3 - G_{1to3}	**5.00e-02**	**2.74e-01**	**2.47e-01**
Border	G_5 - G_{1to5}	3.45e-03	**4.20e-02**	**8.10e-01**
Border	G_{10} - G_{1to10}	6.21e-04	**3.58e-02**	**6.99e-01**
Scattered	G_3 - G_{1to3}	**1.29e-01**	**1.21e-01**	**5.00e-01**
Scattered	G_5 - G_{1to5}	**2.94e-02**	**6.73e-01**	**3.81e-01**
Scattered	G_{10} - G_{1to10}	**4.55e-01**	**7.80e-01**	**1.33e-01**

their success rates and duration of successful searches in validation. Figure 5.3 presents the boxplots of the success rates attained. As can be seen, it seems that in some scenarios the robots perform better by cooperating, while in others, they perform better by acting individually. To shed more light into these comparisons, we proceed to the statistical analysis, with the Kolmogorov-Smirnov test (Table 5.7) showing than none of the data can be considered to follow normal distributions at the chosen confidence interval. As a result, we apply the Friedman's Anova (Table 5.8), with its results showing that there are statistically significant differences between the success rates of the approaches in all scenarios. Finally, the Wilcoxon test is applied (Table 5.9) to assess the existence of statistically significant differences between the success rates of the groups of same size, with the Bonferroni correction being used to adjust the significance value to 1.67e-02. Its results show that there are only significant differences in the stable environment, with the corner and border start region. When using the corner start region, significantly higher success rates can be achieved when using cooperation in groups of three and five robots. In turn, with the border start region, significantly higher success rates can be achieved when using groups of five or ten robots acting individually, while there are no significant differences when using just three robots. Note that in both scenarios, even when there are significant differences, the best overall strategies still achieve success rates of exactly or nearly 100 %.

Table 5.10: P-values of the Kolmogorov-Smirnov test applied to the duration of successful runs

St. region	Env.	G_3	G_{1to3}	G_5	G_{1to5}	G_{10}	G_{1to10}
Corner	S.	1.51e-06	6.44e-07	2.29e-03	7.04e-07	5.89e-04	1.09e-07
Corner	M.	1.90e-04	1.60e-04	1.10e-04	2.08e-04	2.31e-04	7.79e-05
Corner	I. M.	1.16e-04	3.85e-05	1.85e-03	4.53e-06	4.44e-06	2.95e-04
Border	S.	1.36e-05	1.26e-04	1.39e-03	1.28e-03	3.84e-05	2.69e-03
Border	M.	3.66e-06	1.94e-05	2.41e-05	9.18e-05	2.53e-06	9.21e-05
Border	I. M.	7.74e-04	1.98e-04	1.33e-04	6.43e-05	4.40e-04	1.51e-04
Scattered	S.	1.01e-04	3.36e-05	6.34e-04	1.14e-04	6.26e-07	4.79e-07
Scattered	M.	1.05e-03	1.42e-07	4.35e-06	4.53e-04	3.17e-04	2.28e-06
Scattered	I. M.	2.10e-06	3.80e-05	2.05e-06	9.72e-05	1.31e-04	3.49e-07

Table 5.11: P-values of the Friedman's Anova applied to the duration of successful runs

St. region	S. Env.	M. Env.	I. M. Env.
Corner	6.29e-04	4.38e-03	7.41e-03
Border	6.21e-12	1.65e-08	1.48e-11
Scattered	3.79e-18	3.92e-15	3.06e-13

We now focus on the duration of the successful validation runs, which we plot in Figure 5.4. As can be seen, using cooperation often leads to faster searches,

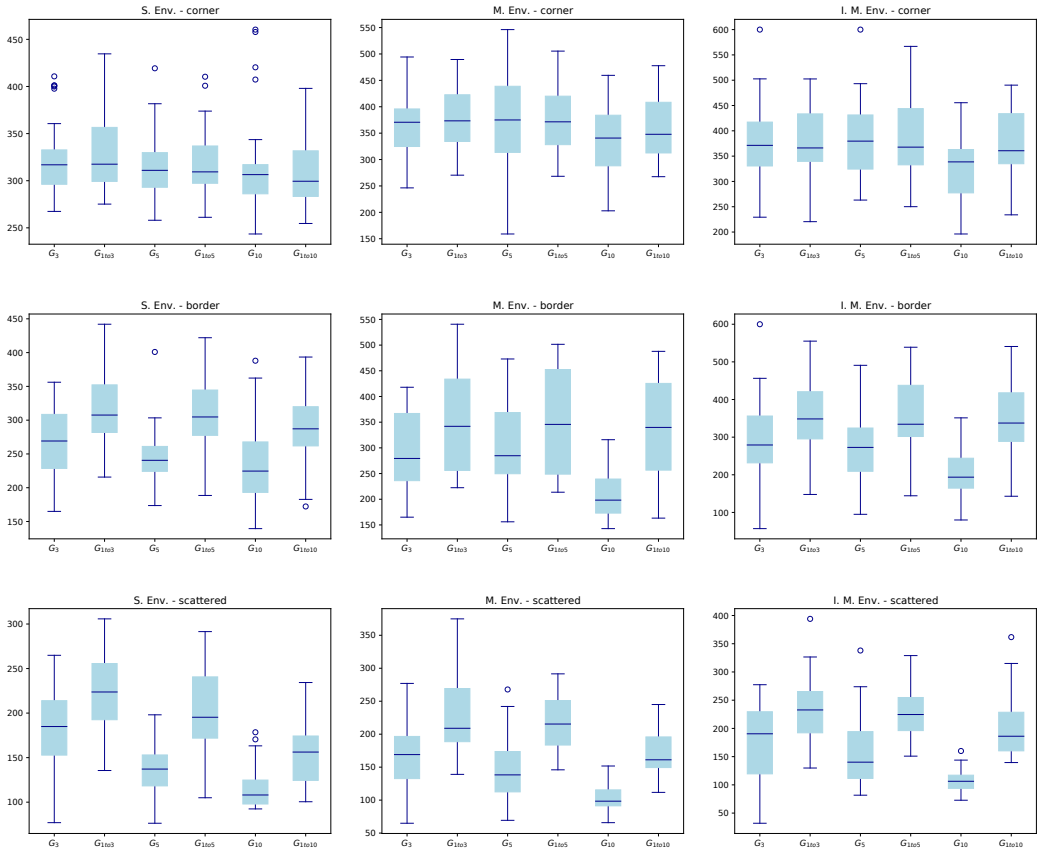


Figure 5.4: Boxplots of the duration of the successful runs in validation for the S. (left column) M. (centre column) and I. M. environments, with the corner (top row), border (centre) and scattered (bottom) start regions.

Table 5.12: P-values of the Wilcoxon test applied to the duration of successful runs

St. region	Functions	S. Env.	M. Env	I. M. Env.
Corner	G_3 - G_{1to3}	**6.44e-01**	**1.59e-01**	**8.29e-01**
Corner	G_5 - G_{1to5}	**7.66e-01**	**9.59e-01**	**8.94e-01**
Corner	G_{10} - G_{1to10}	**6.88e-01**	**3.39e-01**	5.32e-03
Border	G_3 - G_{1to3}	2.26e-03	1.66e-02	2.11e-03
Border	G_5 - G_{1to5}	8.92e-05	**4.72e-02**	1.29e-03
Border	G_{10} - G_{1to10}	8.31e-04	1.24e-05	4.73e-06
Scattered	G_3 - G_{1to3}	2.11e-03	6.64e-04	1.48e-03
Scattered	G_5 - G_{1to5}	1.02e-05	8.92e-05	3.88e-04
Scattered	G_{10} - G_{1to10}	4.20e-04	1.73e-06	1.73e-06

particularly when using the border and scattered start regions. Moving on to the statistical analysis, the Kolmogorov-Smirnov test is first applied (Table 5.10), showing that none of the data can be considered to follow normal distributions. As a result, the Friedman's Anova is applied (Table 5.11), showing that there are statistically significant differences between the search times of the various strategies in all scenarios. Finally, the Wilcoxon test is applied (Table 5.12) and the significance value is adjusted to 1.67e-02 through the Bonferroni correction. The results show that when departing from the corner start region, there are no significant differences between cooperating or acting individually. The sole exception is when using ten robots in the I. M. environment, where cooperation leads to significantly faster searches. When using the border start region, there is a single instance where there are no statistically significant differences: using five robots in the meandering environment. In all other scenarios, along with in all scenarios with the scattered start region, using cooperation consistently leads to significantly faster searches.

5.2.2 Final remarks

In conclusion, including cooperative perceptions and behaviours into the function and terminal sets of GSynGP typically does not lead to significant differences in the success rates, implying that simply using many individualistic robots is enough to solve the task. However, the use of cooperation does result in faster search strategies, often having median search times at least 60 s faster than their individualistic counter parts. Thus, it is more beneficial to make use of cooperation.

5.3 Comparison with single-robot approaches

The experiments made in the previous sections culminate into a simple, yet important question: is it better to use a single robot guided by a complex search strategy or many robots controlled by a simple search strategy? In this section, we address this question by comparing the performance of simple tree-based controllers with one to ten cooperating robots to that of GPI. We start by analysing the success rates attained in validation, which are plotted in Figure 5.5. As was previously seen, the performance of G increases with the size of the group of robots. Moreover, while there are a few scenarios where GPI attains similar success rates to those of G_{10}, and even one where it surpasses the multi-robot approach, in most scenarios it can only match the performance of smaller sized groups. In order to be able to draw more robust conclusions, we move on to the statistical analysis.

We start by applying the Kolmogorov-Smirnov test (Table 5.13), which shows that none of the data can be considered to follow normal distributions. As a result, the Friedman's Anova is applied (Table 5.14), showing that there are statistically significant differences between the various approaches in all scenarios. The Wilcoxon test is then applied (Table 5.15) to assess which comparisons contain statistically significant differences, with the significance value being adjusted

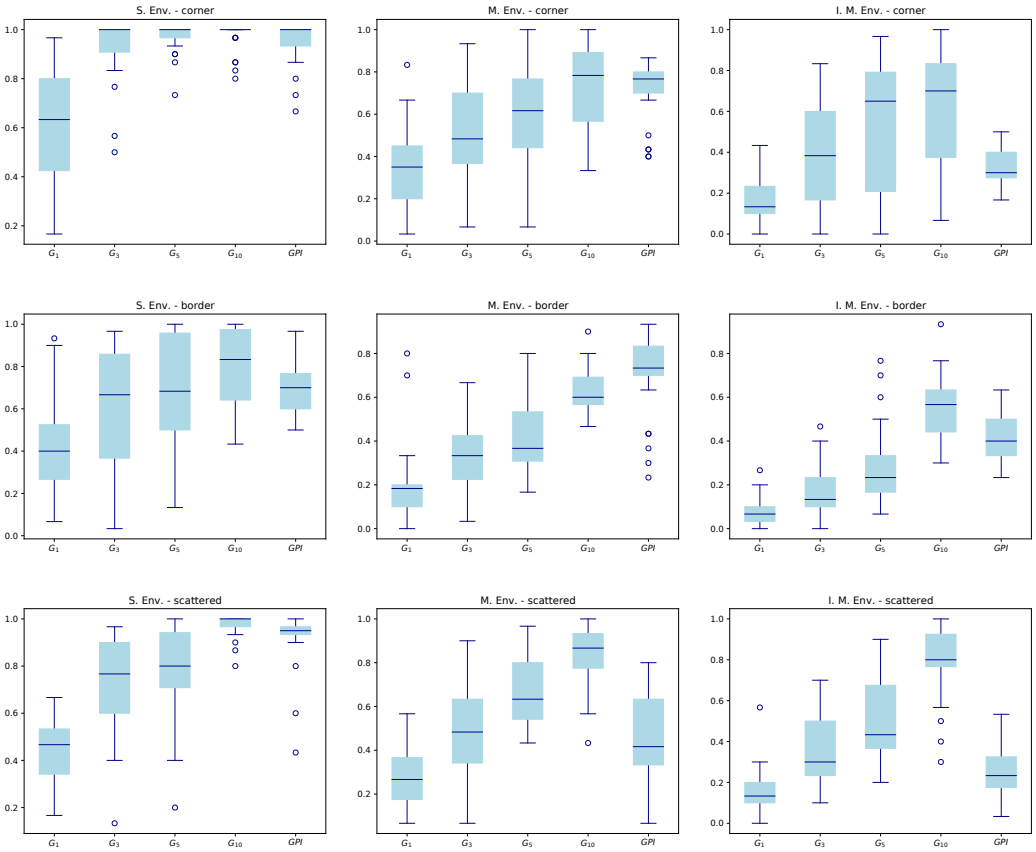


Figure 5.5: Boxplots of the validation success rates for the S. (left column) M. (centre column) and I. M. environments, with the corner (top row), border (centre) and scattered (bottom) start regions.

Table 5.13: P-values of the Kolmogorov-Smirnov test applied to the success rates

St. region	Env.	G_1	G_3	G_5	G_{10}	GPI
Corner	S.	3.32e-05	1.18e-11	1.69e-13	1.50e-18	2.23e-09
Corner	M.	1.77e-04	4.24e-06	1.83e-06	6.32e-07	8.53e-10
Corner	I. M.	2.88e-07	1.69e-06	2.05e-07	1.63e-07	1.62e-07
Border	S.	9.44e-04	1.73e-06	3.07e-06	3.57e-07	8.83e-05
Border	M.	1.36e-04	1.19e-04	5.04e-07	1.12e-07	2.96e-05
Border	I. M.	1.25e-05	5.19e-09	2.50e-08	2.82e-04	3.01e-06
Scattered	S.	2.94e-05	3.64e-06	6.33e-04	7.88e-11	1.02e-09
Scattered	M.	3.00e-05	1.06e-05	1.80e-06	3.63e-06	3.80e-06
Scattered	I. M.	1.70e-06	3.31e-09	1.21e-08	2.12e-05	7.24e-07

Table 5.14: P-values of the Friedman's Anova applied to the success rates

St. region	S. Env.	M. Env.	I. M. Env.
Corner	1.22e-14	4.42e-07	2.41e-10
Border	6.04e-07	4.96e-16	5.99e-20
Scattered	1.27e-16	1.30e-13	6.17e-17

through the Bonferroni correction to 1.25e-02. Its results show that, regardless of the start region, GPI consistently attains significantly higher success rates than G with a single robot. When departing from the corner start region, GPI significantly outperforms G_3 and G_5 in the meandering environment, but performs significantly worse than G_5 and G_{10} in the intermittent-meandering environment. When departing from the Border start region, GPi significantly outperforms G_3 and G_5 in the meandering and intermittent-meandering environments. In turn, G_{10} significantly outperforms GPI in the I. M. environment. Finally, when departing from the scattered start region, GPI significantly outperforms G_3 and G_5 in the stable environment, but performs significantly worse than G_5 and G_{10} in the meandering and intermittent-meandering environments. In all scenarios that were not mentioned, GPI performs equivalently to G. The use of multiple robots seems to be particularly useful in the intermittent-meandering environment, possibly as it is easier to cope with the intermittency of the chemical plume by tracking it cooperatively, i.e., it is more likely that at least one robot is sensing odour. Interestingly, in many scenarios, simply using three robots is enough to perform equivalently to GPI.

We carry on to assess the duration of the successful searches, which we plot in Figure 5.6. The plots show that overall, the speed of G increases with the increase of the number of robots. Regarding GPI, the plots show that is produces the fastest searches when using the corner start region, but when departing from the other start regions, it may become as slow as G_1, if not slower. Moving on to the statistical analysis, we start by applying the Kolmogorov-Smirnov test (Table 5.16) which shows that none of the data can be considered to follow normal dis-

Table 5.15: P-values of the Wilcoxon test applied to the success rates

St. region	Functions	S. Env.	M. Env	I. M. Env.
Corner	G_1 - GPI	5.51e-06	2.01e-06	1.40e-05
Corner	G_3 - GPI	**8.78e-01**	4.04e-04	**4.11e-01**
Corner	G_5 - GPI	**2.47e-01**	9.70e-03	1.91e-03
Corner	G_{10} - GPI	**1.81e-01**	**9.40e-01**	8.52e-05
Border	G_1 - GPI	2.06e-05	1.89e-06	1.70e-06
Border	G_3 - GPI	**1.65e-01**	2.58e-06	2.81e-06
Border	G_5 - GPI	**6.98e-01**	8.01e-06	5.38e-04
Border	G_{10} - GPI	**5.97e-02**	**2.18e-02**	1.80e-04
Scattered	G_1 - GPI	1.72e-06	1.53e-03	1.52e-03
Scattered	G_3 - GPI	3.39e-04	**5.02e-01**	**1.10e-02**
Scattered	G_5 - GPI	1.59e-03	6.86e-05	2.98e-05
Scattered	G_{10} - GPI	**9.86e-03**	6.12e-06	2.10e-06

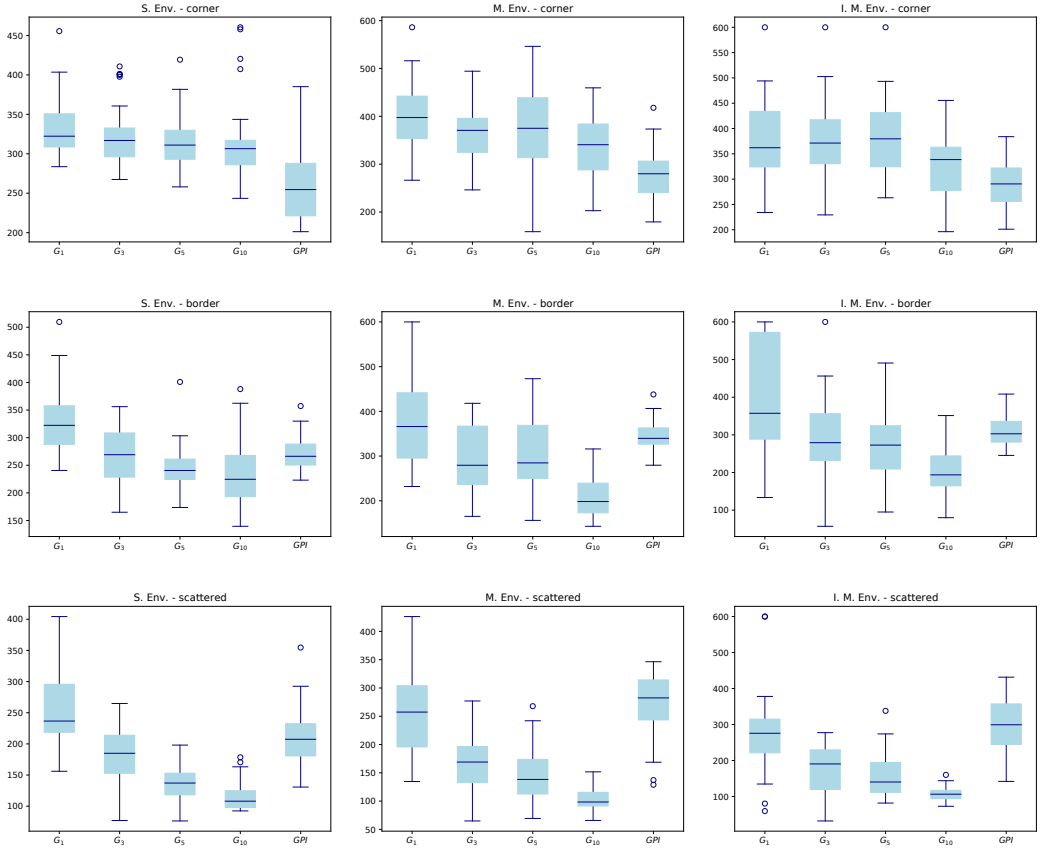


Figure 5.6: Boxplots of the duration of the successful runs for the S. (left column) M. (centre column) and I. M. environments, with the corner (top row), border (centre) and scattered (bottom) start regions.

Table 5.16: P-values of the Kolmogorov-Smirnov test applied to the duration of successful validation runs

St. region	Env.	G_1	G_3	G_5	G_{10}	GPI
Corner	S.	3.06e-07	1.51e-06	2.29e-03	5.89e-04	5.78e-05
Corner	M.	2.32e-06	1.90e-04	1.10e-04	2.31e-04	1.05e-04
Corner	I. M.	3.15e-05	1.16e-04	1.85e-03	4.44e-06	1.71e-05
Border	S.	9.11e-06	1.36e-05	1.39e-03	3.84e-05	1.89e-05
Border	M.	1.02e-04	3.66e-06	2.41e-05	2.53e-06	1.44e-04
Border	I. M.	5.68e-07	7.74e-04	1.33e-04	4.40e-04	1.01e-04
Scattered	S.	4.71e-07	1.01e-04	6.34e-04	6.26e-07	1.76e-04
Scattered	M.	1.23e-04	1.05e-03	4.35e-06	3.17e-04	4.52e-04
Scattered	I. M.	1.74e-03	2.10e-06	2.05e-06	1.31e-04	1.01e-03

Table 5.17: P-values of the Friedman's Anova applied to the duration of successful validation runs

St. region	S. Env.	M. Env.	I. M. Env.
Corner	3.36e-08	7.64e-07	7.99e-05
Border	7.97e-09	2.53e-11	2.90e-09
Scattered	1.28e-18	8.62e-16	1.33e-11

tributions. As a result, we apply the Friedman's Anova (Table 5.17), which shows that there are statistically significant differences between the duration of the searches of the various approaches in all scenarios. Finally, the Wilcoxon test is applied to perform pairwise comparisons, with the significance value being reduced through the Bonferroni correction to 1.25e-02. Its results (Table 5.18) show that using the corner start region, GPI is significantly faster than all other approaches in all environments, with the sole exceptions being G_{10} in the I. M. environment, to which it performs equivalently. The speed of GPI is greatly reduced when departing from the border start region, performing equivalently to G_1 and significantly worse than G_3 and G_5 in the meandering environment. In the stable and intermittent-meandering environments, GPI is significantly faster than G_1, performing equivalently to G_3 and G_5. Moreover, GPI is significantly slower than G_{10} in all environments with the border and scattered regions. The relative performance of GPI is even worse with the scattered start region, only being able to outperform G_1 and match the speed of G_3 in the stable environment. In the meandering and intermittent-meandering environments, it performs equivalently to G_1 and is significantly outperformed by G_3, G_5 and G_{10}. In fact, using five or ten robots produces significantly faster searches than GPI in any environment when starting from the scattered start region. It is also interesting to note that the multi-robot approaches seem to benefit from larger start regions, as they perform faster with the border and scattered start regions than with the corner, in any environment.

Table 5.18: P-values of the Wilcoxon test applied to the duration of successful validation runs

St. region	Functions	S. Env.	M. Env	I. M. Env.
Corner	G_1 - GPI	1.49e-05	4.73e-06	1.06e-04
Corner	G_3 - GPI	1.13e-05	2.84e-05	7.51e-05
Corner	G_5 - GPI	9.71e-05	1.74e-04	4.86e-05
Corner	G_{10} - GPI	5.29e-04	3.59e-04	**2.30e-02**
Border	G_1 - GPI	3.72e-05	**1.06e-01**	8.73e-03
Border	G_3 - GPI	**9.10e-01**	6.04e-03	**1.85e-01**
Border	G_5 - GPI	**1.40e-02**	9.27e-03	**2.43e-02**
Border	G_{10} - GPI	2.96e-03	1.73e-06	4.29e-06
Scattered	G_1 - GPI	1.29e-03	**2.99e-01**	**2.21e-01**
Scattered	G_3 - GPI	**2.56e-02**	1.02e-05	2.37e-05
Scattered	G_5 - GPI	1.73e-06	2.60e-06	1.36e-05
Scattered	G_{10} - GPI	1.73e-06	1.73e-06	1.73e-06

5.3.1 Final remarks

To sum up, using a group of robots with cooperative controllers evolved by GSynGP produces better results than using a single robot controller by a search strategy evolved by GPI. The exact results vary with the scenario, but in many cases using just three robots produces equivalent success rates to those of GPI. Two examples of this is when using the scattered start region in the meandering or intermittent-meandering environments. This may be due to the increased likelihood that a robot senses robot in the multi-robot scenario which, coupled with cooperation, not only enables the robots to converge to the chemical plume faster, but also improves their ability to cope with the plume's intermitency. Furthermore, the results showed that using groups of robots will also lead to faster searches and that in some cases a group of three robots is significantly faster than a single robot with a controller evolved by GPI.

Chapter 6

Wind tunnel validation

Contents

Having shown that evolutionary robotics approaches are suitable for automatically producing interpretable search strategies for odour source localisation, this chapter aims to go one step further, reporting some preliminary experiments in a wind tunnel to validate the best evolved controllers. Due to the available wind tunnel not being able to generate uniform airflows, only the controllers evolved for the stable environment will be validated, using the three start regions. Also, due to time constraints, only ten runs are made for each controller.

6.1 Experimental setup

6.1.1 Wind Tunnel

The test arena (Figure 6.1) is a 4 by 3 meters rectangular environment with 50 centimetres of height covered by plexiglass to enable viewing the robots. The walls along its length are made of plywood, whereas its upwind and downwind walls are made out of honeycomb mesh to reduce wind turbulence. The air-flow is created by an array of fans mounted on the downwind wall, which are set to operate at a constant speed, creating a laminar air-flow with a mean speed of approximately 0.3 m/s. A single odour source consisting of an ultrasonic atomiser (left side of Figure 6.2) is placed at the middle of the environment's width, 0.85 m from its upwind wall, i.e., at position (0.85, 1.5) m. It is set to release a 90 % ethanol solution vapour at a constant rate of approximately 153.51 μg/s. The arena uses the same coordinate reference as the simulator, with the origin being at the top-left corner and point (4, 3) m being the bottom-right corner. Each evaluation ends successfully if a robot gets closer than 0.3 m from the chemical source or unsuccessfully if the 300 s time limit runs out.

Due to the reduced dimensions of the arena, the start regions defined for simulation must also be adjusted. The corner start region generates positions in the rectangle defined by points (3.1, 1.9) m and (3.6, 2.7) m in the odd numbered evaluations and by points (3.1, 0.3) m and (3.6, 1.1) m in the even numbered evaluations; the border start region generates positions within the rectangle defined by points (3.1, 0.3) m and (0.3, 2.7) m; and the scattered start positions in the rectangle defined by points (0.4, 3.6) m and (0.3, 2.7) m.

6.1.2 Robot

The robot used for validating the strategies (right side of Figure 6.2) is an in-house built two-wheeled differential unit based measuring approximately 16 cm in diameter. The robot is equipped with a SGX Sensortech MiCS-5524 sensorfor measuring the gas concentration, along with a LDS-01 LiDAR and two Sharp 2YOA21F57 for measuring the distance to nearby obstacles. The choice of using a MOX sensor is due to their reduced cost, size and acceptable sensitivity and response speeds (see Section 2.1.2.2).

The low-level motion control runs on an STM32F411CEU6, whereas the interface with the sensors and the wireless communications are ensured by an

Figure 6.1: Top-view of the validation arena.

Figure 6.2: Odour source (left) and robot (right).

Espressif ESP32-DevKitC. The robot communicates wirelessly with a remote server running the Robot Operating System framework [Quigley et al., 2009]. The server receives the sensory information from the robot, executes the active controller and returns to the robot a motion command. The localisation of the robots is provided by a Marvelmind beacon system, whose measurements are fused by an Extended Kalman filter with the odometry information, providing a better estimate of the robot's pose.

To reduce costs and improve the robot's battery life, the robots are not equipped with anemometers. Instead, the wind velocity is manually set and kept fixed throughout the experiment.

6.2 Best controllers

We consider the controller with the highest success rate in the simulation-based validation as being the best for each combination of approach and scenario. In case of ties, we choose the one that maximises the success rate while minimising the duration of successful runs. The best individuals for EI and GPI were already presented in the previous chapter (respectively Table 4.3 and Algorithms 4.1 to

4.3). The best controllers evolved by GSynGP for one and three robots are presented on Algorithms 6.1 to 6.6. Note that apart from the removal of introns for improving readability, no modifications were made to the controllers.

The best strategy evolved by GSynGP for one robot departing from the corner start region ($G^*_{1,c}$[1], Algorithm 6.1) starts by checking if odour has been sensed in the last 240 s (line 1). If so, the robot checks if it is currently sensing odour (line 2), in which it remains still for a control step (note the spiral behaviour in line 3 with the same termination criteria as the **If** clause of line 2) and then moves upwind by half a meter (line 4). Otherwise, it performs a spiralling motion halting if has odour is perceived within the past 10 s of the condition verification (line 6). If odour has not been sensed in the past 240 s, the robot wanders crosswind (line 8), and checks if odour has been recently sensed (lines 9 and 10). If so, it moves upwind if the detection was very recent (line 11), or wanders downwind if it has been longer than 10 s (line 13). The first option is useful to cope with the intermittency of the plume, enabling the robot to proceed upwind. In turn, the second option is particularly useful close to the chemical source, where the plume is narrower and the robot may easily move upwind past its location without getting close enough to find it. On the other hand, if no odour was recently sensed, the robot still moves upwind, creating a diagonal motion that leads it closer to the chemical plume (through the combination of lines 8 and 15).

Algorithm 6.1: Best strategy evolved by GSynGP for one robot in the stable environment with the corner start region ($G^*_{1,c}$).

```
1  if HSO(t = 240) then
2      if SO() then
3          spiral(dis_inc = 0.75, it = 1, term = SO())
4          moveUpwind(d = 0.5)
5      else
6          spiral(dis_inc = 0.5, iters = 1, term = HSO(t = 10))
7  else
8      wanderCrosswind()
9      if HSO(t = 137) then
10         if HSO(t = 10) then
11             moveUpwind(d = 0.5)
12         else
13             wanderDownwind()
14     else
15         moveUpwind(d = 0.5)
```

The best strategy evolved by GSynGP for one robot departing from the border start region ($G^*_{1,b}$, Algorithm 6.2) starts by checking if odour has ever been

[1] In this chapter the following notation will be used: $G^*_{1,c}$ specifies the best controller produced by GSynGP with one robot for the corner start region.

sensed (note that the threshold in line 1 is higher than the maximum evaluation time) and, if so, it checks if odour has been sensed in the past 400 s (line 2). If the robot is currently sensing odour, it is commanded to move straight upwind (line 4). Otherwise, if it has sensed odour in the past 400 s, it is commanded to wander upwind (line 6). This is the first controller where there may exist large differences between its performance in validation and in the wind tunnel. Due to the evaluation time being reduced from 600 s (in simulation) to 300 s (in the wind tunnel), it is not possible for the controller to enter the **Else** clause of line 7. As such, as soon as the robot senses odour, it will only move or wander upwind. On the other hand, in simulation, if the robot has sensed odour between 500 s and 400 s ago, then it will wander downwind (line 9). This action is likely to cause it to reencounter the plume, both due to the plume's stability and to the robot only having moved, or wandered upwind. If the robot has not sensed odour for the past 719 s, which considering the maximum evaluation times is the same to say that it has not sensed odour yet, it is commanded to wander crosswind (line 13).

Algorithm 6.2: Best strategy evolved by GSynGP for one robot in the stable environment with the border start region ($G^*_{1,b}$).

```
1  if HSO(T = 719) then
2      if HSO(T = 400) then
3          if SO() then
4              moveUpwind(d = 0.5)
5          else
6              wanderUpwind()
7      else
8          if HSO(T = 500) then
9              wanderDownwind()
10         else
11             wanderUpwind()
12 else
13     wanderCrosswind()
```

Similarly to the previous two, the best strategy evolved by GSynGP for one robot departing from the scattered start region ($G^*_{1,s}$, Algorithm 6.3) also starts by checking if the robot is currently sensing odour (lines 1 to 3), making it move straight upwind if it is (line 4). Otherwise, if the robot has sensed odour within the past 39 s, it still moves upwind (line 7), but for half the step length, as a means of cautiously proceed upwind in case of plume discontinuities. If the robot has sensed odour for longer than 39 s but less than 45 s, it is commanded to wander crosswind (line 9), but if its last odour detection was between 45 s and 200 s, than it is commanded to move randomly (line 11). These two latter behaviours may be considered as plume re-encountering behaviours. If the robot has not sensed odour for longer than 200 s, it is commanded to wander crosswind,

followed by moving randomly for 25 cm (lines 13 and 14).

Algorithm 6.3: Best strategy evolved by GSynGP for one robot in the stable environment with the scattered start region ($G^*_{1,s}$).

```
1  if HSO(t = 200) then
2      if HSO(t = 45) then
3          if SO() then
4              moveUpwind(d = 0.5)
5          else
6              if HSO(t = 39) then
7                  moveUpwind(d = 0.25)
8              else
9                  wanderCrosswind()
10     else
11         moveRandom(d = 0.8)
12 else
13     wanderCrosswind()
14     moveRandom(d = 0.25)
```

Similarly to the single-robot strategies, the controllers evolved for robotic swarms command each robot to move upwind when sensing odour, differing mostly on what to do in the absence of chemical detections. The strategy evolved for the corner start region ($G^*_{3,c}$, Algorithm 6.4), exhibits no form of explicit cooperation. In the absence of chemical detections, it commands each robot to wander crosswind (line 4), followed by an upwind surge of 1 m (line 5). This is similar to what was evolved in the single-robot setting ($G^*_{1,c}$), and translates into a diagonal motion that enables the robot to search in two directions and find plumes that dissipate before the downwind border of the environment. The robot then checks if it has sensed odour in the past 30 s (line 6) and, if so, it either moves randomly if it is currently sensing odour (line 8) or makes a spiralling motion halting if odour has been sensed in the past 5 s (line 10). While at a glance the random motion can be considered to be odd, note that it is in fact a straight motions centred on the current heading of the robot with randomly drawn offset from the [-45°,45°] interval. As a result, considering that the previous motion was an upwind surge, this random motion will be equivalent to a wanderUpwind, but with a step length of 1.105 m, instead of the 0.5 m step, that is predefined for the wanderUpwind. In turn, the spiralling motion is useful for quickly re-encountering the plume, as it is in fact a circle (note the 0.0 dis_{inc}) and it is only performed if the robot is not currently sensing odour, but has detected it in the past 5 s. Moreover, this behaviour is halted as soon as it re-encounters the plume. Finally, if the robot has not sensed odour in the past 30 s, it is commanded to wander crosswind in search for the chemical plume.

The strategy evolved for the border start region ($G^*_{3,b}$, Algorithm 6.5), starts by checking if odour is currently being sensed and, if so, makes the robot move

Algorithm 6.4: Best strategy evolved by GSynGP for three robots in the stable environment with the corner start region ($G^*_{3,c}$).

```
1  if SO() then
2      moveUpwind(d = 0.675)
3  else
4      wanderCrosswind()
5      moveUpwind(d = 1.0)
6      if HSO(t = 30) then
7          if SO() then
8              moveRandom(d = 1.105)
9          else
10             spiral(dis_inc = 0.0, iters = 1, term = HSO(t = 5))
11     else
12         wanderCrosswind()
```

Algorithm 6.5: Best strategy evolved by GSynGP for three robots in the stable environment with the border start region ($G^*_{3,b}$).

```
1  if SO() then
2      moveUpwind(d = 0.4)
3  else
4      if Neighbours(n_t = so, r = 52.217) then
5          moveTowards(n_t = so, d = 2.218, r = 30.0)
6      else
7          moveAway(n_t = so, d = 0.5, r = 1.5)
8          if HSO(t = 662) then
9              if HSO(t = 240) then
10                 moveRandom(d = 0.5)
11             else
12                 moveUpwind(d = 2.361)
13         else
14             wanderCrosswind()
15             wanderCrosswind()
16             wanderCrosswind()
```

0.4 m upwind (line 2). Otherwise, it checks if there are neighbours sensing odour within a 52.217 m radius (line 4). If so, the robot is commanded to move 2.218 m towards the centroid of the neighbours sensing odour within 30 m (line 5). Otherwise, the controller specifies that the robot should move 0.5 m away from all neighbours sensing odour in a 1.5 m radius (line 7). However, as the previous condition assured that there are no such neighbours, the robot will remain still for one control step. It will then check if odour has already been sensed in this run (note the 662 s time threshold in line 8) and if so, it checks if the last detection took place within the past 240 s (line 9), in which case it will move randomly in an attempt to reacquire the plume (line 10). Otherwise, it will proceed moving upwind (line 12). Again, this point could be subject to further improvement as if the last detection took place very long ago, it would likely be better to search a large region towards downwind and crosswind, instead of keeping searching upwind. In case the plume is yet to be encountered in this evaluation (line 13) the robot performs three wanderCrosswind motions. This part of the controller could also be improved by simply removing two wanderCrosswind motions, as it is possible that the robot senses odour after the first motion and moves away from the plume by performing the other two. However, note that all behaviours that do not possess an explicit distance parameter have a step length of 0.5 m. As a result, the three motions will make the robot move, at most, 1.5 m crosswind (considering 0°offsets), which is much smaller than the 40 m arena width.

Algorithm 6.6: Best strategy evolved by GSynGP for three robots in the stable environment with the scattered start region ($G^*_{3,s}$).

```
1  if HSO(T = 221.4) then
2      if SO() then
3          moveUpwind(d = 0.25)
4      else
5          spiral(dis_inc = 0.75, iters = 4, term = SO())
6  else
7      if Neighbours(n_t = nso, r = 5) then
8          moveTowards(n_t = all, d = 0.25, r = 1.5)
9          wanderCrosswind()
10     else
11         moveTowards(n_t = all, d = 2.975, r = 40)
```

Finally, the strategy evolved for the scattered start region ($G^*_{3,s}$, Algorithm 6.6), starts by checking if odour has been sensed within the past 221.4 s and, if so, either moves the robot 0.25 m upwind (line 3) or makes a spiralling motion (line 5) depending on whether the robot is currently sensing odour. If no odour has been recently sensed, the controller checks if there are any robots in a 5 m radius that are not sensing odour (line 7) and, if so, commands the robot to move 0.25 m towards the centre point of all its neighbours within a 1.5 m radius, followed by a wanderCrosswind motion (lines 8 and 9). Otherwise, the robot is

Table 6.1: Success rates of the best controllers

Start region	G_1^*	EI^*	GPI^*	G_3^*
Corner	90 %	90 %	100 %	100 %
Border	70 %	100 %	80 %	60 %
Scattered	80 %	100 %	60 %	100 %

simply commanded to move 2.975 m towards the centre point of all neighbours within a 40 m radius (line 11). This part of the controller could be optimised by replacing line 8 with a moveAway behaviour from the neighbours that are not sensing odour, which would result in the robots scattering, followed by searching crosswind. On another note, the moveTowards behaviour of line 11 is particularly interesting as it takes advantage of the scattered positions of the robots to make them search the environment in various directions simultaneously, by simply commanding them to aggregate.

6.3 Performance comparison

The evolved search strategies are compared through their success rates (i.e., the percentage of times that the robots got closer than 0.3 m from the chemical source) and the duration of successful runs.

Analysing the success rates (Table 6.1) the strategies seem to transfer quite well from simulation to the real world, with most of them succeeding in finding the source in at least 80 % of the evaluations. The exceptions are $G_{1,b}^*$, GPI_s^* and $G_{3,b}^*$, which only managed to attain success rates of respectively 70 %, 60 % and 60 %. The similarity between the results of the various approaches can be seen as an indication of the simplicity of the wind tunnel when compared to the simulation scenario, but may also be a consequence of too few evaluations. Still, the results can be considered to be aligned with the simulations, as EI^* and G_3^* outperform G_1 in two out of three scenarios. Moreover, GPI^* attains a higher success rate than EI^* in the corner scenario. Another interesting conclusion is that, in the corner scenario, complexifying the approach always leads to performance gains, with GPI^* outperforming EI^*, which in turn outperforms G_1^*. However, this trend does not apply to the other scenarios as with the border start region EI^* is the only approach that always succeeds in finding the source and, in the scattered scenario, it also attains a 100 % success rate on par with G_3^*. These results are likely due the mismatch between the simulation environment and wind tunnel rendering the parameters of the tree-based approaches sub-optimal and consequently reducing their performance.

Figure 6.3 presents the boxplots of the durations of successful runs. As can be seen, G_1^* is the slowest approach in the corner scenario, but in the border and scattered scenarios, GPI^* has the highest median search times of all. Interestingly, EI attains the fastest and most consistent searches in the corner scenario, becoming slower in the other two scenarios. An interesting conclusion is that Infotaxis seems to transfer better than the tree-based controllers, implying that

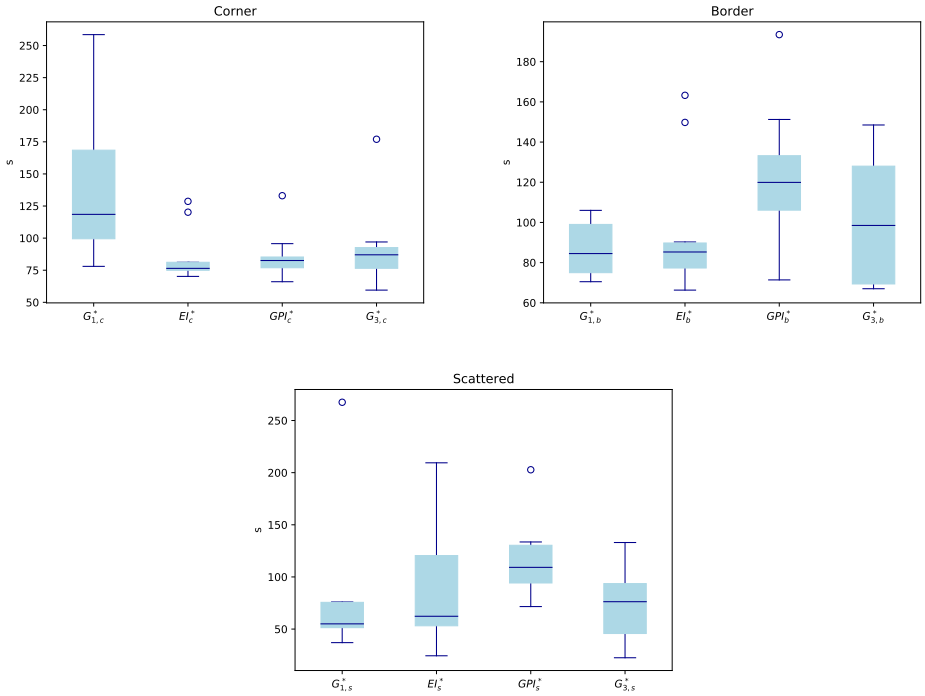


Figure 6.3: Duration of the successful runs when departing from the corner (left), border (centre) and scattered (scenarios).

the overall characteristics of airflow and chemical dispersion are similar in simulation and in the wind-tunnel, but that the reduced dimensions and evaluation time makes the tree-based controllers struggle.

6.4 Fine tuning

In this section we take advantage of the white-box nature of the tree-based controllers to make minor adjustments to the lowest performing strategies from the previous section, in the hope that they become better suited for operating in the wind tunnel. The main rational is: as the evaluation time is half of the one from simulation, the time thresholds should be halved as well. However, this rule cannot be blindly applied, as it may not give enough time to the robots to perform some behaviours. Moreover, the distance parameters should be adjusted to better reflect the constraints of the robots as well as the reduced size of the arena. Even though this is the general rule, there were some occasions where the parameters were left untouched. The modifications made to the search strategies are typeset in boldface.

The first strategy to be modified is the one evolved by GSynGP for a single robot, operating in the stable environment and departing from the corner start region ($G^t_{1,b}$, Algorithm 6.7). Its modifications consist in reducing the time thresholds in lines 1, 2 and 8, as well as reducing the step length in line 4. Note that in this instance the time thresholds were not simply halved, as through experimentation we saw that if the robot moved past the chemical source, it would spend too

Algorithm 6.7: Manually tweaked strategy evolved by GSynGP for one robot in the stable environment with the border start region ($G^t_{1,b}$).

```
1  if HSO(t = 120) then
2      if HSO(t = 30) then
3          if SO() then
4              moveUpwind(d = 0.3)
5          else
6              wanderUpwind()
7      else
8          if HSO(t = 70) then
9              wanderDownwind()
10         else
11             wanderUpwind()
12 else
13     wanderCrosswind()
```

long trying to wander upwind (line 6) and often would not have enough time to move back downwind (line 9). Also, the step length in line 4 was reduced to try to prevent the robot from moving past the location of the source.

The second search strategy to suffer modifications is the one evolved by GSynGP for three robots, operating in the stable environment and departing from the border start region ($G^t_{3,b}$, Algorithm 6.8). The modifications consist on reducing all distance related parameters to adequate values (lines 4, 5, 10 and 12), as well as halving all time thresholds (lines 8 and 9). Moreover, the type of neighbours in the moveAway behaviour was changed to consider only those not sensing odour (line 7), so that the robots can spread out in the environment to maximise their chances of detecting the plume, but without moving away from robots already sensing odour. The modification made to d in line 5 is particularly important, as it prevents a phenomenon observed during the experimentation, where the robots would move too much in the direction of their team-mates sensing odour and would cross the plume without having the chance to start tracking it.

Finally, the third controller to be modified is the one evolved by GPInfotaxis for operating in the stable environment when departing from the scattered start region (GPI^t_s, Algorithm 6.9). Only two lines of this algorithm are modified, yet the modifications are substantial. Firstly, line 2 originally commanded the robot to wander downwind, but now makes it move straight upwind for 0.5 m when sensing odour, which should drive it closer to the source. A similar rationale is applied to line 7, where the robot was originally commanded to move randomly. Considering that the robot is either sensing odour or has sensed it in the past second, this line is modified to wander upwind, so that the robot moves towards the likely location of the source while still maintaining some randomness.

Ten runs were made for each of search strategy and the success rates are presen-

Algorithm 6.8: Manually tweaked strategy evolved by GSynGP for three robots in the stable environment with the border start region ($G^t_{3,b}$).

```
1  if SO() then
2      moveUpwind(d = 0.3)
3  else
4      if Neighbours(n_t = so, r = 5.2) then
5          moveTowards(n_t = so, d = 0.3, r = 3)
6      else
7          moveAway(n_t = nso, d = 0.5, r = 1.5)
8          if HSO(t = 330) then
9              if HSO(t = 120) then
10                 moveRandom(d = 0.3)
11             else
12                 moveUpwind(d = 0.3)
13         else
14             wanderCrosswind()
15             wanderCrosswind()
16             wanderCrosswind()
```

Algorithm 6.9: Manually tweaked strategy evolved by GPInfotaxis in the stable environment with the scattered start region (GPI^t_s).

```
1  if SO() then
2      moveUpwind(d=0.5)
3      Infotaxis()
4  else
5      Infotaxis()
6      if HSO(t = 1) then
7          wanderUpwind()
8      else
9          Infotaxis()
```

Table 6.2: Success rates of the original and manually tweaked controllers

	$G_{1,b}$	GPI_s	$G_{3,b}$
Original	70 %	60 %	60 %
Tweaked	100 %	100 %	90 %

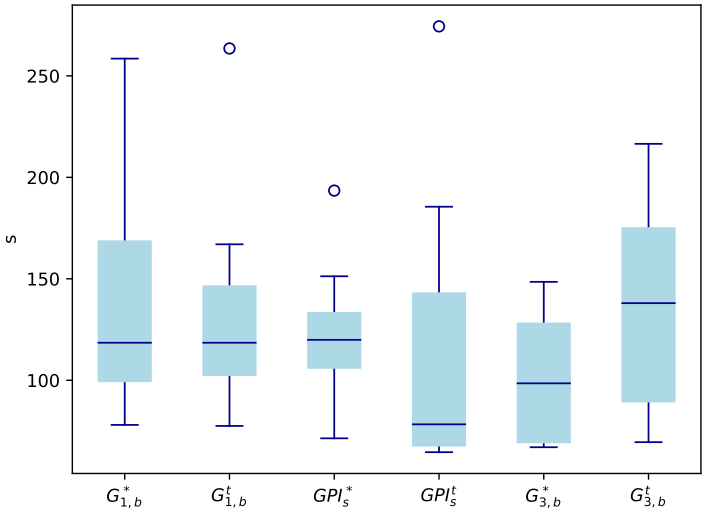


Figure 6.4: Duration of the successful runs when controlled by the manually tweaked search strategies.

ted on Table 6.2. It shows that the modifications made produced good results, with the biggest improvements being made for GPI_s, whose success rate raised from 60 % to 100 %. The other two strategies also achieved good improvements, with $G_{1,b}$ and $G_{3,b}$ raising their success rates from respectively 70 % and 60 % to 100 % and 90 %. The multi-robot approach is the only that does fails to always locate the source, which is likely due to the ratio of size between the robots, the chemical plume and the arena. While in simulation various robots could track the plume simultaneously, in the arena the robots would often interfere with each other. That interference was minimised by adjusting the parameters of $G_{3,b}$, but still could not be completely eradicated. The duration of the successful runs is plotted in Figure 6.4, showing that $G_{1,b}$ becomes more consistent with the modifications made, while GPI_s becomes faster and $G_{3,b}$ becomes slower. The increased search times of $G_{3,b}$ may be due to the reduction of the step length of line 12, which originally made the robots cross more than half of the environment with a single motion. In turn, the increased speed of GPI_s is likely related with the modifications made to direct its motions upwind.

These results support the need for evolution, as simply tweaking the evolved solutions proves to be a hard task. This is particularly true in the multi-robot setting, where the success rate could be increased but at the cost of the median search time increasing by roughly 50 %.

Chapter 7

Conclusions and Future Work

The goal of this thesis was to devise nature-inspired algorithms to automatically produce robotic controllers for locating odour sources. This goal led to the proposal of two inherently different approaches, which answer our second research question (i.e., how can one robot evolve to locate an odour source): (1) EInfotaxis, which uses a Genetic Algorithm to automatically parametrise the gas distribution model of Infotaxis; and (2) the evolution of white-box, tree-based controllers with Genetic Programming, which later culminated into GPInfotaxis, a hybrid approach that combines both bio-inspired building blocks with infotactic behaviours. The desire for the controllers to be human-readable led us to the proposal of Geometric Syntactic Genetic Programming, a novel GP algorithm which performs geometric crossover operations in the syntactic space and that showed to evolve much smaller individuals than the standard GP algorithm with no performance loss. Using GSynGP, the evolution of multi-robot cooperative controllers was also explored, answering our third research question (i.e., how can a group of robots evolve to locate chemical sources cooperatively). Finally, to answer the fourth research question, the performance of the multi-robot approach was compared to the single robot methods, showing that using many simple robots even outperforms GPInfotaxis, i.e., the most complex single-robot approach.

Two main conclusions can be drawn from this thesis:

1. Regarding general GP experiments, the results showed that GSynGP is the most suitable approach, producing smaller solutions than SGP with equivalent performance, which will not only be more interpretable, but also more efficient. Note that efficiency should not only be seen as a matter of execution speed, but also energy consumption and memory space, all of which are primary concerns if the evolved solutions are to be deployed in resource constrained devices, such as microcontrollers. Moreover, the successful application of GSynGP to the evolution of robotic controllers answers our first research question, i.e., GSynGP is a suitable algorithm to evolve human readable search strategies for odour source localisation.

2. Regarding the methods aimed at odour source localisation, one out of three approaches can be recommended:

    a) If only a single resource-limited robot is available, then GSynGP should be used to evolve a controller with only simple behaviours and actions. While this approach will have the least performance when compared to the others, in the wind tunnel it performed overall quite well, and the resulting controller will be the least computationally expensive of all. Also, if more robots become available, the results showed that simply replicating the individual controller over several robots will lead to much better performance.

    b) If a single robot with sufficient computing capability and battery range is available, then GPInfotaxis should be used to evolve the controller. Note however that using a single robot is always riskier than multiple ones, as there is no redundancy to failures.

c) Finally, if multiple robots are available, GSynGP should be used to evolve cooperative strategies, as the resulting controllers can not only outperform all single-robot approaches, but are also able to cope with the loss of agents.

The analysis of the evolved solutions also provided insights into future efforts, namely by showing that the parameters that enable Infotaxis to achieve its best performance are not necessarily those that make its gas distribution model match the environment the best. Also, GPInfotaxis' strategies showed that Infotaxis is mostly useful for finding the chemical plume, while bio-inspired behaviours are better suited to track it to its source. A selection of the best evolved controllers were tested in a controlled wind-tunnel, qualitatively validating the simulation results.

Future Work

In the future, and as a result of this thesis, various research strands can be followed:

- Apply the controllers in the real world - This thesis followed a commonly used methodology in ER, where the controllers are evolved offline, in simulation, and tested in the real world. While we presented some preliminary experiments of the best evolved controllers in a wind tunnel, it would be interesting to apply them in the real world, possibly involving different types of robots (i.e., terrestrial, aerial and aquatic). Such experiments would not only allow us to understand how they cope with different types of robotic platforms and, possibly, how could they be improved for each specific type of robot.
- Online (co-)evolution - Most ER works evolve the controllers in simulation due to it being easier, cheaper, faster and safer than evolving in the real world. However, as our preliminary experiments showed, this leads to the reality gap problem. It would be interesting to enable the robots to evolve in the real world and, if more than one robot is available, to explore the benefits of co-evolution. This poses some questions that must be addressed, such as the fitness computation, which should be made through internal fitness functions, i.e., fitness functions composed by objectives that each robot can measure without any external help or global knowledge, such as the distance travelled or the time spent sensing odour. Also, the issue of unfair evaluations must be dealt with, as in online evolution the evaluation of each controller starts from the previous evaluation last state. In turn, the co-evolution goal poses questions such as the capabilities that each robot should have (should each robot have a single controller and depend on its encounters with others to reproduce or should each robot be able to evolve its own population) and also how should they interact (e.g., should the robots perform crossover together or, rather, share individuals with their neighbours, i.e., through migration).
- Multi-robot GPInfotaxis - This thesis presented a first step into evolving

cooperative controllers for odour source localisation, showing that it can outperform GPInfotaxis, the most complex single-robot approach. However, it would be interesting to evolve multi-robot cooperative controllers with GPInfotaxis, possibly with each robot having different Infotaxis parameters. This goal poses questions regarding the existence of multiple Infotaxis parameters, that must be evolved and dealt with efficiently, but also with how should the cooperation be achieved. A simple way would be to consider that each robot performs Infotaxis individually, and achieve cooperation only through the behaviours and perceptions in the terminal and function sets. However, some researchers have been working on cooperative Infotaxis methods that should be investigated. Moreover, with such multi-robot approach would be possible to have a group of heterogeneous robots, where some have less resources and thus do not implement Infotactic behaviours, but others do.

- Improve the perceptions and behaviours - GSynGP evolves the controllers by combining the perceptions and behaviours that the experimenter included in its function and terminal sets. This can be seen as a limitation, as if necessary symbols are not included, the algorithm will never be able to evolve the optimal solution. As a result, in the future, further symbols should be included, based on natural search strategies. As an example, a flocking behaviour may be particularly useful for tracking a chemical plume with high intermittency and meandering.
- Explore the use of pheromones - In nature, many animals use pheromones to communicate. As an extension of this work, it would be interesting to use pheromones to convert the environment into a mean of indirect communication, marking regions that have been previously explored to prevent wasting resources. The pheromones laid could be different based on the chemical detection rates perceived in a given location, as to convey the degree of certainty that a region should be avoided.

Bibliography

[Allard et al., 2022] Allard, M., Smith, S. C., Chatzilygeroudis, K., and Cully, A. (2022). Hierarchical quality-diversity for online damage recovery. *arXiv preprint arXiv:2204.05726*.

[Ampatzis et al., 2008] Ampatzis, C., Tuci, E., Trianni, V., and Dorigo, M. (2008). Evolution of signaling in a multi-robot system: Categorization and communication. *Adaptive Behavior*, 16(1):5–26.

[Aoki, 1982] Aoki, I. (1982). A simulation study on the schooling mechanism in fish. *Nippon Suisan Gakkaishi*, 48(8):1081–1088.

[Balaprakash et al., 2007] Balaprakash, P., Birattari, M., and Stützle, T. (2007). Improvement strategies for the f-race algorithm: Sampling design and iterative refinement. In *International workshop on hybrid metaheuristics*, pages 108–122. Springer.

[Baldassarre et al., 2007] Baldassarre, G., Trianni, V., Bonani, M., Mondada, F., Dorigo, M., and Nolfi, S. (2007). Self-organized coordinated motion in groups of physically connected robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(1):224–239.

[Beer and Gallagher, 1992] Beer, R. D. and Gallagher, J. C. (1992). Evolving dynamical neural networks for adaptive behavior. *Adaptive behavior*, 1(1):91–122.

[Birattari et al., 2002] Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K., et al. (2002). A racing algorithm for configuring metaheuristics. In *Gecco*, volume 2.

[Bongard, 2013] Bongard, J. C. (2013). Evolutionary robotics. *Communications of the ACM*, 56(8):74–83.

[Boudardara and Gorkemli, 2018] Boudardara, F. and Gorkemli, B. (2018). Application of artificial bee colony programming to two trails of the artificial ant problem. In *2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pages 1–6. IEEE.

[Braitenberg, 1986] Braitenberg, V. (1986). *Vehicles: Experiments in synthetic psychology*. MIT press.

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

[Bredeche, 2014] Bredeche, N. (2014). Embodied evolutionary robotics with large number of robots. In *14th international conference on the synthesis and simulation of living systems (ALIFE 14)*, pages 1–2.

[Bredeche and Fontbonne, 2022] Bredeche, N. and Fontbonne, N. (2022). Social learning in swarm robotics. *Philosophical Transactions of the Royal Society B*, 377(1843):20200309.

[Bredeche et al., 2018] Bredeche, N., Haasdijk, E., and Prieto, A. (2018). Embodied evolution in collective robotics: a review. *Frontiers in Robotics and AI*, 5:12.

[Bräunl, 2006] Bräunl, T. (2006). *Embedded Robotics: mobie robot design and applications with embedded systems*. Springer.

[Buchanan et al., 2020] Buchanan, E., Le Goff, L. K., Li, W., Hart, E., Eiben, A. E., De Carlo, M., Winfield, A. F., Hale, M. F., Woolley, R., Angus, M., et al. (2020). Bootstrapping artificial evolution to design robots for autonomous fabrication. *Robotics*, 9(4):106.

[Cabrita et al., 2013] Cabrita, G., Marques, L., and Gazi, V. (2013). Virtual cancelation plume for multiple odor source localization. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 5552–5558. IEEE.

[Chung et al., 2018] Chung, S.-J., Paranjape, A. A., Dames, P., Shen, S., and Kumar, V. (2018). A survey on aerial swarm robotics. *IEEE Transactions on Robotics*, 34(4):837–855.

[Colledanchise and Ögren, 2018] Colledanchise, M. and Ögren, P. (2018). *Behavior trees in robotics and AI: An introduction*. CRC Press.

[Collins, 2022] Collins, J. T. (2022). *Simulation to reality and back: A robot's guide to crossing the reality gap*. PhD thesis, Queensland University of Technology.

[Coppola et al., 2020] Coppola, M., McGuire, K. N., De Wagter, C., and De Croon, G. C. (2020). A survey on swarming with micro air vehicles: Fundamental challenges and constraints. *Frontiers in Robotics and AI*, 7:18.

[Crespi et al., 2008] Crespi, V., Galstyan, A., and Lerman, K. (2008). Top-down vs bottom-up methodologies in multi-agent system design. *Autonomous Robots*, 24:303–313.

[Cully et al., 2015] Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B. (2015). Robots that can adapt like animals. *Nature*, 521(7553):503–507.

[Cully and Mouret, 2016] Cully, A. and Mouret, J.-B. (2016). Evolving a behavioral repertoire for a walking robot. *Evolutionary computation*, 24(1):59–88.

[de Croon et al., 2013] de Croon, G., O'Connor, L., Nicol, C., and Izzo, D. (2013). Evolutionary robotics approach to odor source localization. *Neurocomputing*, 121:481 – 497. Advances in Artificial Neural Networks and Ma-

chine LearningSelected papers from the 2011 International Work Conference on Artificial Neural Networks (IWANN 2011).

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

[Dickman et al., 2009] Dickman, B. D., Webster, D. R., Page, J. L., and Weissburg, M. J. (2009). Three-dimensional odorant concentration measurements around actively tracking blue crabs. *Limnol. Oceanogr. Methods*, 7:96–108.

[Divband Soorati and Hamann, 2015] Divband Soorati, M. and Hamann, H. (2015). The effect of fitness function design on performance in evolutionary robotics: The influence of a priori knowledge. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 153–160.

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

[Dönmez and Kocamaz, 2020] Dönmez, E. and Kocamaz, A. F. (2020). Design of mobile robot control infrastructure based on decision trees and adaptive potential area methods. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, 44(1):431–448.

[Duarte et al., 2014] Duarte, M., Oliveira, S., and Christensen, A. (2014). Hybrid control for large swarms of aquatic drones. In *ALIFE 14: The Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, pages 785–792. MIT Press.

[Duarte et al., 2012a] Duarte, M., Oliveira, S., and Christensen, A. L. (2012a). Automatic synthesis of controllers for real robots based on preprogrammed behaviors. In *International Conference on Simulation of Adaptive Behavior*, pages 249–258. Springer.

[Duarte et al., 2012b] Duarte, M., Oliveira, S., and Christensen, A. L. (2012b). Hierarchical evolution of robotic controllers for complex tasks. In *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on*, pages 1–6. IEEE.

[Duisterhof et al., 2021] Duisterhof, B. P., Li, S., Burgués, J., Reddi, V. J., and de Croon, G. C. (2021). Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9099–9106. IEEE.

[Duistermars et al., 2009] Duistermars, B. J., Chow, D. M., and Frye, M. A. (2009). Flies require bilateral sensory input to track odor gradients in flight. *Current Biology*, 19(15):1301 – 1307.

[Eiben et al., 2010a] Eiben, A. E., Haasdijk, E., and Bredeche, N. (2010a). *Em-*

bodied, on-line, on-board evolution for autonomous robotics, pages 361–382. Springer, Berlin, Heidelberg.

[Eiben et al., 2010b] Eiben, A. E., Karafotias, G., and Haasdijk, E. (2010b). Self-adaptive mutation in on-line, on-board evolutionary robotics. In *2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop*, pages 147–152. IEEE.

[Eiben and Smith, 2015] Eiben, A. E. and Smith, J. E. (2015). *Introduction to evolutionary computing*. Springer.

[Emmerich and Deutz, 2018] Emmerich, M. T. and Deutz, A. H. (2018). A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural computing*, 17:585–609.

[Farrell et al., 2002] Farrell, J. A., Murlis, J., Long, X., Li, W., and Cardé, R. T. (2002). Filament-based atmospheric dispersion model to achieve short time-scale structure of odor plumes. *Environmental Fluid Mechanics*, 2(1):143–169.

[Farrell et al., 2003] Farrell, J. A., Pang, S., and Li, W. (2003). Plume mapping via hidden markov methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 33(6):850–863.

[Feng et al., 2020] Feng, Q., Cai, H., Yang, Y., Xu, J., Jiang, M., Li, F., Li, X., and Yan, C. (2020). An experimental and numerical study on a multi-robot source localization method independent of airflow information in dynamic indoor environments. *Sustainable Cities and Society*, 53:101897.

[Ferrante et al., 2013] Ferrante, E., Duéñez-Guzmán, E., Turgut, A. E., and Wenseleers, T. (2013). Geswarm: Grammatical evolution for the automatic synthesis of collective behaviors in swarm robotics. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 17–24.

[Ferrante et al., 2012] Ferrante, E., Turgut, A. E., Huepe, C., Stranieri, A., Pinciroli, C., and Dorigo, M. (2012). Self-organized flocking with a mobile robot swarm: a novel motion control method. *Adaptive Behavior*, 20(6):460–477.

[Ferri et al., 2009] Ferri, G., Caselli, E., Mattoli, V., Mondini, A., Mazzolai, B., and Dario, P. (2009). Spiral: A novel biologically-inspired algorithm for gas/odor source localization in an indoor environment with no strong airflow. *Robotics and Autonomous Systems*, 57(4):393–402.

[Floreano and Keller, 2010] Floreano, D. and Keller, L. (2010). Evolution of adaptive behaviour in robots by means of darwinian selection. *PLoS biology*, 8(1):e1000292.

[Francesca and Birattari, 2016] Francesca, G. and Birattari, M. (2016). Automatic design of robot swarms: achievements and challenges. *Frontiers in Robotics and AI*, 3:29.

[Francesca et al., 2015] Francesca, G., Brambilla, M., Brutschy, A., Garattoni, L., Miletitch, R., Podevijn, G., Reina, A., Soleymani, T., Salvaro, M., Pin-

ciroli, C., et al. (2015). Automode-chocolate: automatic design of control software for robot swarms. *Swarm Intelligence*, 9(2):125–152.

[Francesca et al., 2014] Francesca, G., Brambilla, M., Brutschy, A., Trianni, V., and Birattari, M. (2014). Automode: A novel approach to the automatic design of control software for robot swarms. *Swarm Intelligence*, 8(2):89–112.

[Francis et al., 2022] Francis, A., Li, S., Griffiths, C., and Sienz, J. (2022). Gas source localization and mapping with mobile robots: A review. *Journal of Field Robotics*.

[Gongora et al., 2017] Gongora, A., Monroy, J. G., and Gonzalez-Jimenez, J. (2017). A robotic experiment toward understanding human gas-source localization strategies. In *Olfaction and Electronic Nose (ISOEN), 2017 ISOCS/IEEE International Symposium on*, pages 1–3. IEEE.

[Grasso et al., 2000] Grasso, F. W., Consi, T. R., Mountain, D. C., and Atema, J. (2000). Biomimetic robot lobster performs chemo-orientation in turbulence using a pair of spatially separated sensors: Progress and challenges. *Robotics and Autonomous Systems*, 30(1-2):115–131.

[Grzes, 2017] Grzes, M. (2017). Reward shaping in episodic reinforcement learning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, page 565–573, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

[Haasdijk et al., 2010] Haasdijk, E., Eiben, A., and Karafotias, G. (2010). On-line evolution of robot controllers by an encapsulated evolution strategy. In *IEEE Congress on Evolutionary Computation*, pages 1–7. IEEE.

[Hamann, 2018] Hamann, H. (2018). *Swarm robotics: A formal approach*, volume 221. Springer.

[Hammad et al., 2019] Hammad, I., El-Sankary, K., and Gu, J. (2019). A comparative study on machine learning algorithms for the control of a wall following robot. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2995–3000. IEEE.

[Hamzei et al., 1999] Hamzei, G. S., Mulvaney, D. J., and Sillitoe, I. (1999). Becoming incrementally reactive: on-line learning of an evolving decision tree array for robot navigation. *Robotica*, 17(3):325–334.

[Harvey et al., 2008] Harvey, D. J., Lu, T.-F., and Keller, M. A. (2008). Comparing insect-inspired chemical plume tracking algorithms using a mobile robot. *IEEE Transactions on Robotics*, 24(2):307–317.

[Hauert et al., 2011] Hauert, S., Leven, S., Varga, M., Ruini, F., Cangelosi, A., Zufferey, J.-C., and Floreano, D. (2011). Reynolds flocking in reality with fixed-wing robots: communication range vs. maximum turning rate. In *2011 IEEE/RSJ international conference on intelligent robots and systems*, pages 5015–5020. IEEE.

[Hauert et al., 2009] Hauert, S., Zufferey, J.-C., and Floreano, D. (2009).

Evolved swarming without positioning information: an application in aerial communication relay. *Autonomous Robots*, 26(1):21–32.

[Hayes et al., 2002] Hayes, A. T., Martinoli, A., and Goodman, R. M. (2002). Distributed odor source localization. *IEEE Sensors Journal*, 2(3):260–271.

[Hayes-Roth, 1985] Hayes-Roth, F. (1985). Rule-based systems. *Communications of the ACM*, 28(9):921–932.

[Howard et al., 2002] Howard, A., Matarić, M. J., and Sukhatme, G. S. (2002). Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Distributed autonomous robotic systems 5*, pages 299–308. Springer.

[Huang and Liang, 2002] Huang, H.-P. and Liang, C.-C. (2002). Strategy-based decision making of a soccer robot system using a real-time self-organizing fuzzy decision tree. *Fuzzy Sets and Systems*, 127(1):49–64.

[Hutchinson et al., 2018] Hutchinson, M., Oh, H., and Chen, W. (2018). Entrotaxis as a strategy for autonomous search and source reconstruction in turbulent conditions. *Information Fusion*, 42:179–189.

[Ishida et al., 1995] Ishida, H., Kagawa, Y., Nakamoto, T., and Moriizumi, T. (1995). Odor-source localization in clean room by autonomous mobile sensing system. In *Proceedings of the International Solid-State Sensors and Actuators Conference - TRANSDUCERS '95*, volume 1, pages 783–786. IEEE.

[Ishida et al., 2012] Ishida, H., Wada, Y., and Matsukura, H. (2012). Chemical sensing in robotic applications: A review. *IEEE Sensors Journal*, 12(11):3163–3173.

[Izquierdo and Lockery, 2010] Izquierdo, E. J. and Lockery, S. R. (2010). Evolution and analysis of minimal neural circuits for klinotaxis in caenorhabditis elegans. *Journal of Neuroscience*, 30(39):12908–12917.

[Jakobi et al., 1995] Jakobi, N., Husbands, P., and Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics. In *European Conference on Artificial Life*, pages 704–720. Springer.

[Jatmiko et al., 2007] Jatmiko, W., Sekiyama, K., and Fukuda, T. (2007). A pso-based mobile robot for odor source localization in dynamic advection-diffusion with obstacles environment: theory, simulation and measurement. *IEEE Computational Intelligence Magazine*, 2(2):37–51.

[Jin and Branke, 2005] Jin, Y. and Branke, J. (2005). Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317.

[Jing et al., 2021] Jing, T., Meng, Q.-H., and Ishida, H. (2021). Recent progress and trend of robot odor source localization. *IEEJ Transactions on Electrical and Electronic Engineering*.

[Jo, 2021] Jo, T. (2021). *Machine Learning Foundations: Supervised, Unsupervised, and Advanced Learning*. Springer Cham.

[Jones et al., 2018] Jones, S., Studley, M., Hauert, S., and Winfield, A. (2018). Evolving behaviour trees for swarm robotics. In *Distributed Autonomous Robotic Systems*, pages 487–501. Springer.

[Jones et al., 2019] Jones, S., Winfield, A. F., Hauert, S., and Studley, M. (2019). Onboard evolution of understandable swarm behaviors. *Advanced Intelligent Systems*, 1(6):1900031.

[Koch et al., 2021] Koch, J., Langosco, L., Pfau, J., Le, J., and Sharkey, L. (2021). Objective robustness in deep reinforcement learning. *arXiv preprint arXiv:2105.14111*.

[Kovincic et al., 2020] Kovincic, N., Gattringer, H., Müller, A., and Brandstötter, M. (2020). A boosted decision tree approach for a safe human-robot collaboration in quasi-static impact situations. In *International Conference on Robotics in Alpe-Adria Danube Region*, pages 235–244. Springer.

[Kowadlo and Russell, 2008] Kowadlo, G. and Russell, R. A. (2008). Robot odor localization: a taxonomy and survey. *The International Journal of Robotics Research*, 27(8):869–894.

[Koza, 1992] Koza, J. (1992). *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press.

[Koza, 1994] Koza, J. (1994). *Genetic programming II: automatic discovery of reusable programs*. MIT press.

[Koza and Rice, 1992] Koza, J. and Rice, J. (1992). Automatic programming of robots using genetic programming. In *AAAI*, volume 92, pages 194–207. Citeseer.

[Kuckling et al., 2018] Kuckling, J., Ligot, A., Bozhinoski, D., and Birattari, M. (2018). Behavior trees as a control architecture in the automatic modular design of robot swarms. In *Swarm Intelligence: 11th International Conference, ANTS 2018, Rome, Italy, October 29–31, 2018, Proceedings 11*, pages 30–43. Springer.

[Kuwana et al., 1996] Kuwana, Y., Shimoyama, I., Sayama, Y., and Miura, H. (1996). Synthesis of pheromone-oriented emergent behavior of a silkworm moth. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96*, volume 3, pages 1722–1729. IEEE.

[Labella et al., 2006] Labella, T. H., Dorigo, M., and Deneubourg, J.-L. (2006). Division of labor in a group of robots inspired by ants' foraging behavior. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(1):4–25.

[Langdon and Poli, 2013] Langdon, W. B. and Poli, R. (2013). *Foundations of genetic programming*. Springer Science & Business Media.

[Lee et al., 2018] Lee, R., Mou, S., Dasagi, V., Bruce, J., Leitner, J., and Sünderhauf, N. (2018). Zero-shot sim-to-real transfer with modular priors. *Computing Research Repository (CoRR)*.

[Lehman et al., 2012] Lehman, J., Risi, S., D'ambrosio, D. B., and Stanley, K. O. (2012). Rewarding reactivity to evolve robust controllers without multiple trials or noise. In *Artificial Life Conference Proceedings 12*, pages 379–386. MIT Press.

[Lehman and Stanley, 2011] Lehman, J. and Stanley, K. O. (2011). Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 211–218.

[Liberzon et al., 2018] Liberzon, A., Harrington, K., Daniel, N., Gurka, R., Harari, A., and Zilman, G. (2018). Moth-inspired navigation algorithm in a turbulent odor plume from a pulsating source. *PloS one*, 13(6).

[Liu et al., 2007] Liu, W., Winfield, A. F., Sa, J., Chen, J., and Dou, L. (2007). Towards energy optimization: Emergent task allocation in a swarm of foraging robots. *Adaptive behavior*, 15(3):289–305.

[Lochmatter, 2010] Lochmatter, T. (2010). *Bio-inspired and probabilistic algorithms for distributed odor source localization using mobile robots*. PhD thesis, EPFL.

[Lochmatter et al., 2013] Lochmatter, T., Aydın Göl, E., Navarro, I., and Martinoli, A. (2013). *A Plume Tracking Algorithm Based on Crosswind Formations*, pages 91–102. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Luke, 2013] Luke, S. (2013). *Essentials of metaheuristics*, volume 2. Lulu Raleigh.

[Luke and Panait, 2002] Luke, S. and Panait, L. (2002). Lexicographic parsimony pressure. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 829–836.

[Macedo et al., 2018] Macedo, J., Fonseca, C. M., and Costa, E. (2018). Geometric crossover in syntactic space. In *European Conference on Genetic Programming*, pages 237–252. Springer.

[Macedo et al., 2016] Macedo, J., Marques, L., and Costa, E. (2016). Evolving neural networks for multi-robot odor search. In *Autonomous Robot Systems and Competitions (ICARSC), 2016 International Conference on*, pages 288–293. IEEE.

[Macedo et al., 2019] Macedo, J., Marques, L., and Costa, E. (2019). A comparative study of bio-inspired odour source localisation strategies from the state-action perspective. *Sensors*, 19(10):2231.

[Macedo et al., 2020] Macedo, J., Marques, L., and Costa, E. (2020). Locating odour sources with geometric syntactic genetic programming. In *European Conference on the Applications of Evolutionary Computation*. Springer.

[Macedo et al., 2021a] Macedo, J., Marques, L., and Costa, E. (2021a). Designing fitness functions for odour source localisation. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 103–104.

[Macedo et al., 2021b] Macedo, J., Marques, L., and Costa, E. (2021b). Evolving infotaxis for meandering environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8431–8436. IEEE.

[Macedo et al., 2022] Macedo, J. a., Marques, L., and Costa, E. (2022). Hybridizing bio-inspired strategies with infotaxis through genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '22, page 95–103, New York, NY, USA. Association for Computing Machinery.

[Marjovi et al., 2009] Marjovi, A., Nunes, J. G., Marques, L., and De Almeida, A. (2009). Multi-robot exploration and fire searching. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1929–1934. IEEE.

[Marques et al., 2022] Marques, L., Magalhães, H., Baptista, R., and Macedo, J. (2022). *Mobile robot olfaction state-of-the-art and research challenges*, pages 97–213. IET.

[Marques et al., 2003] Marques, L., Nunes, U., and De Almeida, A. (2003). Odour searching with autonomous mobile robots: An evolutionary-based approach. In *Proceedings of the IEEE Int. Conf. on Advanced Robotics*, pages 494–500.

[Marques et al., 2002a] Marques, L., Nunes, U., and de Almeida, A. T. (2002a). Cooperative odour field exploration with genetic algorithms. In *Proc. 5th Portuguese Conf. on Automatic Control (CONTROLO 2002)*, pages 138–143. Citeseer.

[Marques et al., 2002b] Marques, L., Nunes, U., and de Almeida, A. T. (2002b). Olfaction-based mobile robot navigation. *Thin solid films*, 418(1):51–58.

[Marques et al., 2006] Marques, L., Nunes, U., and de Almeida, A. T. (2006). Particle swarm-based olfactory guided search. *Autonomous Robots*, 20(3):277–287.

[Martin Moraud and Martinez, 2010] Martin Moraud, E. and Martinez, D. (2010). Effectiveness and robustness of robot infotaxis for searching in dilute conditions. *Frontiers in neurorobotics*, 4:1.

[Mataric, 1994] Mataric, M. J. (1994). Reward functions for accelerated learning. In *Machine learning proceedings 1994*, pages 181–189. Elsevier.

[Maxim et al., 2009] Maxim, P. M., Spears, W. M., and Spears, D. F. (2009). Robotic chain formations. *IFAC Proceedings Volumes*, 42(22):19–24.

[McDermott et al., 2012] McDermott, J., White, D. R., Luke, S., Manzoni, L., Castelli, M., Vanneschi, L., Jaskowski, W., Krawiec, K., Harper, R., De Jong, K., et al. (2012). Genetic programming needs better benchmarks. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 791–798.

[McGuire et al., 2019] McGuire, K., De Wagter, C., Tuyls, K., Kappen, H., and de Croon, G. C. (2019). Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Science Robotics*, 4(35):eaaw9710.

[Mehta et al., 2020] Mehta, B., Handa, A., Fox, D., and Ramos, F. (2020). A user's guide to calibrating robotics simulators. *arXiv preprint arXiv:2011.08985*.

[Mermoud et al., 2014] Mermoud, G., Upadhyay, U., Evans, W. C., and Martinoli, A. (2014). Top-down vs. bottom-up model-based methodologies for distributed control: a comparative experimental study. In *Experimental Robotics*, pages 615–629. Springer.

[Moctezuma et al., 2012] Moctezuma, L. E. G., Lobov, A., and Lastra, J. L. M. (2012). Decision making by using tree-like structures on industrial controllers. In *2012 Tenth International Conference on ICT and Knowledge Engineering*, pages 77–83. IEEE.

[Moraglio, 2008] Moraglio, A. (2008). *Towards a geometric unification of evolutionary algorithms*. PhD thesis, University of Essex.

[Moraglio et al., 2012] Moraglio, A., Krawiec, K., and Johnson, C. G. (2012). Geometric semantic genetic programming. In *International Conference on Parallel Problem Solving from Nature*, pages 21–31. Springer.

[Moran, 2006] Moran, M. E. (2006). The da vinci robot. *Journal of endourology*, 20(12):986–990.

[Mostaghim et al., 2010] Mostaghim, S., Trautmann, H., and Mersmann, O. (2010). Preference-based multi-objective particle swarm optimization using desirabilities. In *International Conference on Parallel Problem Solving from Nature*, pages 101–110. Springer.

[Mouret and Chatzilygeroudis, 2017] Mouret, J.-B. and Chatzilygeroudis, K. (2017). 20 years of reality gap: a few thoughts about simulators in evolutionary robotics. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1121–1124.

[Mouret and Clune, 2015] Mouret, J.-B. and Clune, J. (2015). Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*.

[Mouret et al., 2013] Mouret, J.-B., Koos, S., and Doncieux, S. (2013). Crossing the reality gap: a short introduction to the transferability approach. *arXiv preprint arXiv:1307.1870*.

[Murata and Kurokawa, 2007] Murata, S. and Kurokawa, H. (2007). Self-reconfigurable robots. *IEEE Robotics & Automation Magazine*, 14(1):71–78.

[Nedjah and Junior, 2019] Nedjah, N. and Junior, L. S. (2019). Review of methodologies and tasks in swarm robotics towards standardization. *Swarm and Evolutionary Computation*, 50:100565.

[Nelson et al., 2009] Nelson, A. L., Barlow, G. J., and Doitsidis, L. (2009). Fit-

ness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345–370.

[Nolfi et al., 2016] Nolfi, S., Bongard, J., Husbands, P., and Floreano, D. (2016). *Evolutionary Robotics*, pages 2035–2068. Springer International Publishing, Cham.

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

[Nolfi et al., 1994] Nolfi, S., Floreano, D., Miglino, O., and Mondada, F. (1994). How to evolve autonomous robots: Different approaches in evolutionary robotics. In *Artificial life iv: Proceedings of the fourth international workshop on the synthesis and simulation of living systems*, pages 190–197. MIT press.

[Nolfi and Parisi, 1996] Nolfi, S. and Parisi, D. (1996). Learning to adapt to changing environments in evolving neural networks. *Adaptive behavior*, 5(1):75–98.

[Nordin and Banzhaf, 1997] Nordin, P. and Banzhaf, W. (1997). An on-line method to evolve behavior and to control a miniature robot in real time with genetic programming. *Adaptive Behavior*, 5(2):107–140.

[Nouyan et al., 2008] Nouyan, S., Campo, A., and Dorigo, M. (2008). Path formation in a robot swarm. *Swarm Intelligence*, 2(1):1–23.

[Nurzaman et al., 2011] Nurzaman, S. G., Matsumoto, Y., Nakamura, Y., Koizumi, S., and Ishiguro, H. (2011). 'yuragi'-based adaptive mobile robot search with and without gradient sensing: From bacterial chemotaxis to a levy walk. *Advanced Robotics*, 25(16):2019–2037.

[Otfinoski, 2007] Otfinoski, S. (2007). *Rockets*. Great Inventions. Cavendish Square Publishing.

[O'Grady et al., 2010] O'Grady, R., Groß, R., Christensen, A. L., and Dorigo, M. (2010). Self-assembly strategies in a group of autonomous mobile robots. *Autonomous Robots*, 28(4):439–455.

[Park and Oh, 2020] Park, M. and Oh, H. (2020). Cooperative information-driven source search and estimation for multiple agents. *Information Fusion*, 54:72–84.

[Parker, 2008] Parker, L. E. (2008). *Multiple Mobile Robot Systems*, pages 921–941. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Perez et al., 2011] Perez, D., Nicolau, M., O'Neill, M., and Brabazon, A. (2011). Evolving behaviour trees for the mario ai competition using grammatical evolution. In *European Conference on the Applications of Evolutionary Computation*, pages 123–132. Springer.

[Quigley et al., 2009] Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. (2009). ROS: an open-source

robot operating system. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan.

[Reynolds, 1987] Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, pages 25–34.

[Ristic and Gilliam, 2019] Ristic, B. and Gilliam, C. (2019). Decentralised scalable search for a hazardous source in turbulent conditions. In *Unmanned Robotic Systems and Applications*. IntechOpen.

[Ristic et al., 2016] Ristic, B., Skvortsov, A., and Gunatilaka, A. (2016). A study of cognitive strategies for an autonomous search. *Information Fusion*, 28:1–9.

[Rodríguez et al., 2017] Rodríguez, J. D., Gómez-Ullate, D., and Mejía-Monasterio, C. (2017). On the performance of blind-infotaxis under inaccurate modeling of the environment. *The European Physical Journal Special Topics*, 226(10):2407–2420.

[Rokach and Maimon, 2005] Rokach, L. and Maimon, O. (2005). Decision trees. In *Data mining and knowledge discovery handbook*, pages 165–192. Springer.

[Rozas et al., 1991] Rozas, R., Morales, J., and Vega, D. (1991). Artificial smell detection for robotic navigation. In *Fifth International Conference on Advanced Robotics' Robots in Unstructured Environments*, pages 1730–1733. IEEE.

[Rubenstein et al., 2014] Rubenstein, M., Cornejo, A., and Nagpal, R. (2014). Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799.

[Ruddick et al., 2018] Ruddick, J., Marjovi, A., Rahbar, F., and Martinoli, A. (2018). Design and performance evaluation of an infotaxis-based three-dimensional algorithm for odor source localization. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1413–1420. IEEE.

[Russell et al., 2003] Russell, R. A., Bab-Hadiashar, A., Shepherd, R. L., and Wallace, G. G. (2003). A comparison of reactive robot chemotaxis algorithms. *Robotics and Autonomous Systems*, 45(2):83–97.

[Ryan et al., 1998] Ryan, C., Collins, J. J., and Neill, M. O. (1998). Grammatical evolution: Evolving programs for an arbitrary language. In *European conference on genetic programming*, pages 83–96. Springer.

[Şahin, 2004] Şahin, E. (2004). Swarm robotics: From sources of inspiration to domains of application. In *International Workshop on Swarm Robotics*, pages 10–20. Springer.

[Salvato et al., 2021] Salvato, E., Fenu, G., Medvet, E., and Pellegrino, F. A. (2021). Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning. *IEEE Access*, 9:153171–153187.

[Saska, 2015] Saska, M. (2015). Mav-swarms: unmanned aerial vehicles stabilized along a given path using onboard relative localization. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 894–903. IEEE.

[Saulnier et al., 2017] Saulnier, K., Saldana, D., Prorok, A., Pappas, G. J., and Kumar, V. (2017). Resilient flocking for mobile robot teams. *IEEE Robotics and Automation letters*, 2(2):1039–1046.

[Scheper and De Croon, 2017] Scheper, K. Y. and De Croon, G. C. (2017). Abstraction, sensory-motor coordination, and the reality gap in evolutionary robotics. *Artificial Life*, 23(2):124–141.

[Scheper et al., 2016] Scheper, K. Y., Tijmons, S., de Visser, C. C., and de Croon, G. C. (2016). Behavior trees for evolutionary robotics. *Artificial life*, 22(1):23–48.

[Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[Sekhavat, 2017] Sekhavat, Y. A. (2017). Behavior trees for computer games. *International Journal on Artificial Intelligence Tools*, 26(02):1730001.

[Shah and Gopal, 2010] Shah, H. and Gopal, M. (2010). A fuzzy decision tree-based robust markov game controller for robot manipulators. *International Journal of Automation and Control*, 4(4):417–439.

[Shucker and Bennett, 2007] Shucker, B. and Bennett, J. K. (2007). Scalable control of distributed robotic macrosensors. In *Distributed Autonomous Robotic Systems 6*, pages 379–388. Springer.

[Shucker et al., 2008] Shucker, B., Murphey, T. D., and Bennett, J. K. (2008). Convergence-preserving switching for topology-dependent decentralized systems. *IEEE Transactions on Robotics*, 24(6):1405–1415.

[Silva et al., 2015] Silva, F., Urbano, P., Correia, L., and Christensen, A. L. (2015). odneat: An algorithm for decentralised online evolution of robotic controllers. *Evolutionary Computation*, 23(3):421–449.

[Singh et al., 2023] Singh, S. H., van Breugel, F., Rao, R. P., and Brunton, B. W. (2023). Emergent behaviour and neural dynamics in artificial agents tracking odour plumes. *Nature Machine Intelligence*, 5(1):58–70.

[Song et al., 2020] Song, C., He, Y., Ristic, B., and Lei, X. (2020). Collaborative infotaxis: Searching for a signal-emitting source based on particle filter and gaussian fitting. *Robotics and Autonomous Systems*, 125:103414.

[Song et al., 2019] Song, C., He, Y., Ristic, B., Li, L., and Lei, X. (2019). Multi-agent collaborative infotaxis search based on cognition difference. *Journal of Physics A: Mathematical and Theoretical*, 52(48):485202.

[Soysal and Sahin, 2005] Soysal, O. and Sahin, E. (2005). Probabilistic aggreg-

ation strategies in swarm robotic systems. In *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, pages 325–332. IEEE.

[Spears et al., 2004] Spears, W. M., Spears, D. F., Hamann, J. C., and Heil, R. (2004). Distributed, physics-based control of swarms of vehicles. *Autonomous Robots*, 17(2-3):137–162.

[Sprague et al., 2018] Sprague, C. I., Özkahraman, Ö., Munafo, A., Marlow, R., Phillips, A., and Ögren, P. (2018). Improving the modularity of auv control systems using behaviour trees. In *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, pages 1–6. IEEE.

[Stockie, 2011] Stockie, J. M. (2011). The mathematics of atmospheric dispersion modeling. *SIAM Review*, 53(2):349–372.

[Sungkono et al., 2016] Sungkono, S. K., Yohanes, B. W., and Santoso, D. (2016). Decision tree analysis for humanoid robot soccer goalkeeper algorithm. In *2016 6th International Annual Engineering Seminar (InAES)*, pages 46–50. IEEE.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[Svec and Gupta, 2012] Svec, P. and Gupta, S. K. (2012). Automated synthesis of action selection policies for unmanned vehicles operating in adverse environments. *Autonomous Robots*, 32(2):149–164.

[Swere and Mulvaney, 2003] Swere, E. and Mulvaney, D. J. (2003). Robot navigation using decision trees. *Electr. Eng*, pages 15–17.

[Szabo, 2015] Szabo, T. (2015). Autonomous collision avoidance for swarms of mavs: based solely on rssi measurements. Master's thesis, Delft University of Technology.

[Trautmann and Weihs, 2006] Trautmann, H. and Weihs, C. (2006). On the distribution of the desirability index using Harrington's desirability function. *Metrika*, 63(2):207–213.

[Trianni, 2008] Trianni, V. (2008). *Evolutionary swarm robotics: evolving self-organising behaviours in groups of autonomous robots*. Springer.

[Tuci and Trianni, 2014] Tuci, E. and Trianni, V. (2014). On the evolution of homogeneous two-robot teams: clonal versus aclonal approaches. *Neural Computing and Applications*, 25(5):1063–1076.

[Tzafestas, 2013] Tzafestas, S. G. (2013). *Introduction to mobile robot control*. Elsevier.

[Usui and Arita, 2003] Usui, Y. and Arita, T. (2003). Situated and embodied evolution in collective evolutionary robotics. In *In Proc. of the 8th International Symposium on Artificial Life and Robotics*. Citeseer.

[Vanneschi, 2016] Vanneschi, L. (2016). An introduction to geometric semantic genetic programming. In *NEO 2015: Results of the Numerical and Evolu-*

tionary Optimization Workshop NEO 2015 held at September 23-25 2015 in Tijuana, Mexico, pages 3–42. Springer.

[Vergassola et al., 2007] Vergassola, M., Villermaux, E., and Shraiman, B. I. (2007). 'infotaxis' as a strategy for searching without gradients. *Nature*, 445(7126):406–409.

[Villarreal et al., 2016] Villarreal, B. L., Olague, G., and Gordillo, J. L. (2016). Synthesis of odor tracking algorithms with genetic programming. *Neurocomputing*, 175:1019–1032.

[Vuong et al., 2015] Vuong, T. P., Loukas, G., Gan, D., and Bezemskij, A. (2015). Decision tree-based detection of denial of service and command injection attacks on robotic vehicles. In *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE.

[Waibel et al., 2009] Waibel, M., Keller, L., and Floreano, D. (2009). Genetic team composition and level of selection in the evolution of cooperation. *IEEE Transactions on Evolutionary Computation*, 13(3):648–660.

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

[Weissburg and Dusenbery, 2002] Weissburg, M. J. and Dusenbery, D. B. (2002). Behavioral observations and computer simulations of blue crab movement to a chemical source in a controlled turbulent flow. *Journal of Experimental Biology*, 205(21):3387–3398.

[White et al., 2013] White, D. R., McDermott, J., Castelli, M., Manzoni, L., Goldman, B. W., Kronberger, G., Jaśkowski, W., O'Reilly, U.-M., and Luke, S. (2013). Better gp benchmarks: community survey results and proposals. *Genetic Programming and Evolvable Machines*, 14:3–29.

[Wilkerson and Tauritz, 2011] Wilkerson, J. L. and Tauritz, D. R. (2011). A guide for fitness function design. In *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 123–124.

[Wilson et al., 2014] Wilson, S., Pavlic, T. P., Kumar, G. P., Buffin, A., Pratt, S. C., and Berman, S. (2014). Design of ant-inspired stochastic control policies for collective transport by robotic swarms. *Swarm Intelligence*, 8(4):303–327.

[Zagal and Ruiz-Del-Solar, 2007] Zagal, J. C. and Ruiz-Del-Solar, J. (2007). Combining simulation and reality in evolutionary robotics. *Journal of Intelligent and Robotic Systems*, 50(1):19–39.

[Zagal et al., 2004] Zagal, J. C., Ruiz-del Solar, J., and Vallejos, P. (2004). Back to reality: Crossing the reality gap in evolutionary robotics. *IFAC Proceedings Volumes*, 37(8):834–839.

[Zarzhitsky et al., 2005] Zarzhitsky, D., Spears, D., Thayer, D., and Spears, W. (2005). Agent-based chemical plume tracing using fluid dynamics. In *Formal Approaches to Agent-Based Systems: Third International Workshop, FAABS*

2004, Greenbelt, MD, April 26-27, 2004, Revised Selected Papers 3, pages 146–160. Springer.

[Zhuang and Hadfield-Menell, 2020] Zhuang, S. and Hadfield-Menell, D. (2020). Consequences of misaligned ai. *Advances in Neural Information Processing Systems*, 33:15763–15773.

[Ziegler and Banzhaf, 2001] Ziegler, J. and Banzhaf, W. (2001). Evolving control metabolisms for a robot. *Artificial Life*, 7(2):171–190.