Locating Odour Sources with Geometric Syntactic Genetic Programming

João Macedo[1,2(✉)], Lino Marques[1], and Ernesto Costa[2]

[1] ISR, Department of Electrical and Computer Engineering, University of Coimbra, 3030 290 Coimbra, Portugal
{jmacedo,lino}@isr.uc.pt

[2] CISUC, Department of Informatics Engineering, University of Coimbra, 3030 290 Coimbra, Portugal
ernesto@dei.uc.pt

Abstract. Using robots to locate odour sources is an interesting problem with important applications. Many researchers have drawn inspiration from nature to produce robotic methods, whilst others have attempted to automatically create search strategies with Artificial Intelligence techniques. This paper extends Geometric Syntactic Genetic Programming and applies it to automatically produce robotic controllers in the form of behaviour trees. The modification proposed enables Geometric Syntactic Genetic Programming to evolve trees containing multiple symbols per node. The behaviour trees produced by this algorithm are compared to those evolved by a standard Genetic Programming algorithm and to two bio-inspired strategies from the literature, both in simulation and in the real world. The statistically validated results show that the Geometric Syntactic Genetic Programming algorithm is able to produce behaviour trees that outperform the bio-inspired strategies, while being significantly smaller than those evolved by the standard Genetic Programming algorithm. Moreover, that reduction in size does not imply statistically significant differences in the performance of the strategies.

Keywords: Evolutionary Robotics · Odour source localisation · Genetic Programming · Geometric operators

1 Introduction

Olfaction enables the detection and localisation of distant targets, even if they are silent and invisible. While in nature animals use this sense to locate food, detect danger and mates, humans may use it for locating victims in disaster scenarios, detecting illegal substances or tracking sources of pollution. But, locating odour sources in realistic environments is not easy as the odour particles released flow with the wind, spreading through molecular diffusion and turbulent dispersion. The resulting chemical plumes are intermittent, containing local voids and peaks of concentration which hinder the ability to estimate local gradients. The process of locating an odour source has three well-defined stages [1],

P. A. Castillo et al. (Eds.): EvoApplications 2020, LNCS 12104, pp. 212–227, 2020.
https://doi.org/10.1007/978-3-030-43722-0_14

each requiring a distinct behaviour: (1) plume searching, where the agent must explore the environment, searching for initial odour cues; (2) plume tracking, where the agent is sensing the odour plume and must follow it to the vicinity of its source; and (3) source localisation, where the agent is close to the odour source and must pinpoint its location. Most of the existing works, including the present one, focus only on the plume finding and tracking stages, assuming that other sensory perceptions (e.g., vision) are used to identify the odour source once it is close enough.

Due to the great ability of animals to locate odour sources, many of the published works draw inspiration from their behaviours to devise search strategies [2,3]. The present paper proposes to go one step further, creating new strategies by means of a new kind of Evolutionary Algorithm. Evolutionary Algorithms (EA) are stochastic search heuristics inspired by Darwin's principles of evolution and by Mendel's genetics, which have produced good solutions to difficult problems from many application domains. They have been successfully applied to design robots and their controllers, yielding the field of Evolutionary Robotics (ER) [4]. In this work, a sub-family of EAs entitled Genetic Programming (GP) [5] is used to improve the search strategies encoded as behaviour trees. One of the purposes of evolving robotic controllers in the form of behaviour trees is to enable their interpretation by humans. However, GP suffers from bloat, a phenomenon which translates into an uncontrolled growth of the behaviour trees without a correspondent increase in performance. This growth not only renders the trees hard to be interpreted by humans, but also hampers their interpretation by computers, consequently slowing down the evolutionary process. While there are methods available in the literature to cope with bloat [6] these methods typically restrict the search ability of the evolutionary algorithm.

This work investigates the ability of Geometric Syntactic Genetic Programming (GSynGP) [7] to evolve the robotic controllers for locating odour sources. In its original version, this method has been shown to produce controlled variations of the individuals, resulting in an implicit control of their growth. The present paper extends GSynGP to enable it to evolve expression trees with multiple symbols per node. The best controllers produced by GSynGP are compared to those evolved by the standard Genetic Programming algorithm (SGP) and to two bio-inspired strategies from the literature, both in simulation and in the real world. The statistically validated experimental results show that GSynGP is able to produce behaviour trees that outperform the bio-inspired strategies, while being significantly smaller than those evolved by SGP. Moreover, there are no statistically significant differences between the fitness of the strategies evolved by the two GP algorithms.

2 Background and Related Work

Consider a mobile robot r moving in R^2. The robot is equipped with the necessary sensors to measure the wind direction, the concentration of a target odour and the distance to nearby obstacles. The robot is placed in a bounded arena A where there is a single odour source S emitting at a constant rate. The location

of the odour source is unknown. The goal of the robot is to fulfil the first two stages of the odour source localisation process (i.e., to detect the plume and follow it to the vicinity of its source) before a time limit T. This section briefly presents some of the background and related works on odour source localisation.

2.1 Odour Source Localisation Strategies

Over the past decades, researchers have proposed various bio-inspired strategies for locating odour sources. Similarly to the natural organisms that provide inspiration, those strategies are meant to work under particular environmental conditions. The wind speed is amongst the most relevant environmental variables. In the presence of very weak winds, odour disperses mainly through diffusion, whereas in environments with strong wind the odour spreads mainly through turbulent advection. The resulting dispersion patterns are quite distinct, requiring fundamentally different search strategies.

In environments lacking strong wind, chemotactic strategies are typically employed, which use only chemical information to guide their search process. An example of such strategies is the method inspired by the *E. coli* bacteria [8], which consists of a biased random walk composed only by rotations and linear motions. On each iteration, the search agent compares the local chemical concentration to its previous measurement. If the concentration has increased, the agent makes a small rotation followed by a large straight motion, continuing searching roughly in the same direction. Otherwise, it makes a large rotation followed by a short straight motion, directing the search to another direction.

In environments where there is a strong air-flow, animals typically employ strategies that use the direction of the wind for guiding the search, i.e., they perform anemotaxis. A popular anemotactic strategy is inspired by the behaviour of the Male Silkworm Moth (SM) while tracking a trail of pheromone released by a female moth [9]. This algorithm is based on three behaviours: straight line upwind surges when detecting odour, and upwind-centred zigzag or spiral motions for re-encountering the chemical plume. Another popular anemotactic strategy is inspired by the Dung Beetle (DB) tracking a cow's pat [9]. In this approach, the robot starts with a plume finding behaviour, moving crosswind in search for odour cues. Upon sensing odour, it performs an odour-centred zigzag behaviour for tracking the plume to its source. In this behaviour, the robot moves diagonally upwind, changing direction every time it stops sensing odour.

2.2 Evolutionary Algorithms

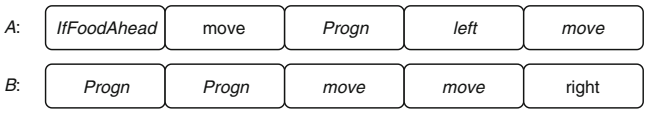
Evolutionary Algorithms (EA) are a family of stochastic search heuristics loosely inspired by the principles of evolution through natural selection and Mendel's genetics. The application of these heuristics to the automatic design of robots and their controllers yielded a novel research area known as Evolutionary Robotics (ER) [4]. While there are many ways to represent robotic controllers, a popular choice are Behaviour Trees (BT), which are human-readable directional graphs. The trees are composed by inner nodes, encoding decision or sequence functions,

and leaf nodes that encode the actions of the robot. In this work, a Genetic Programming (GP) algorithm is used for evolving robotic controllers in the form of Behaviour Trees. GPs [5] are a family of EAs which evolve computer programs that produce the solutions to a given problem, rather than evolving that solution directly. The evolved programs are typically represented by expression trees, composed by inner and leaf nodes. Thus, GPs may be seamlessly used to evolve behaviour trees. Genetic Programming has already been applied to the evolution of robotic controllers for odour source localisation [10]. That work differs from the present one as it focused solely on the evolution of chemotactic strategies, which where tested in an indoor environment without air flow. Moreover, the standard Genetic Programming algorithm used is known to suffer from bloat, producing large expression trees that hinder their interpretation by humans. The present paper investigates the applicability of a recently proposed geometric GP algorithm [7] which has been shown to implicitly control bloat. An extension to that algorithm is made to enable it to evolve expression trees with multiple symbols per node. This is useful to be able to evolve strategies with complex behaviours (e.g., zigzag), rather than elementary actions (e.g., move, rotate) as used in [10]. This algorithm is presented in Sect. 3.

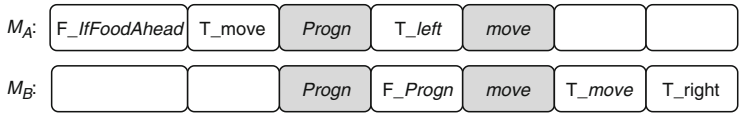
3 Geometric Syntactic Genetic Programming

Geometric variation operators are representation-independent operators based on a distance defined in the search space interpreted as a metric space [11]. 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 ball centred on the original individual whose radius defines the magnitude of the mutation. Geometric Syntactic Genetic Programming (GSynGP) [7] 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 an expression tree in prefix notation. The crossover operation uses the Longest Common Subsequence to align the genomes of the two parent individuals and to create two modification masks, which are used to alter a copy of one individual so that it becomes more similar to the other. 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.

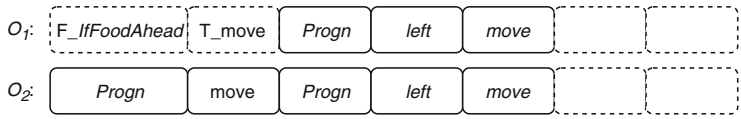
As an example, consider the Santa Fe Ant trail benchmark problem [5], for which the terminal set is {*left*, *right*, *move*} and the function set is {*IfFoodAhead*, *Progn*}. Two possible strategies are:



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



Due to space limitations, the algorithm for creating the modification masks is not reproduced in this work and we direct the interested reader to [7]. 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 where insertions or deletions must be made; and (3) the non-common symbols, marked with a $T\_$ or $F\_$ depending on whether they belong to the terminal or function set, 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. In this work, only a single iteration of the crossover operator is used.

3.1 Extended GSynGP

The original version of Geometric Syntactic Genetic Programming considered expression trees where each node contains a single symbol. However, in many cases the nodes of the expression trees may contain more than one symbol. This work is one of such cases, 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_i, while the remaining take the value from N_d. In this work k was set to 1.

As an example consider the following two nodes:

- N_d: $ZZ(dir = U,\ dist = 0.5,\ iters = 1,\ off = \pi/4,\ term = PL(5))$
- N_i: $SP(s\_dir = r,\ dist = 1,\ dist\_inc = 0.1,\ intvs = 7,\ term = SO())$

which are contained in the terminal set described in Sect. 4.3. 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 Fig. 1. As before, N_n is created with the symbol *SP*. The parameters *dir*, *iters* and *off* from N_d are not present in N_i and thus are ignored. The parameters *s_dir, dist_inc* and *intvs* are only present in N_i and thus are added to N_n with their original values. The parameter *term* is present in both nodes but takes a non-numerical value and thus it is added to N_n with the value from N_i. The only remaining parameter is *dis*, which is added to N_n with the mean of the values from N_d and N_i. 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.

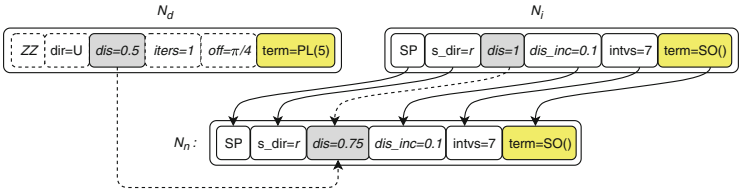


Fig. 1. Creation of a new node, merging the parameters of its parent nodes.

4 Experimental Setup

This section presents the robot, simulator and validation environments used for conducting the experiments, as well as provides details regarding the Genetic Programming algorithms used.

4.1 Odour-Seeking Robot

The robot used for validating the strategies is an in-house built two-wheeled differential unit based on DFRobot's MiniQ 2WD v2.1, that has been extensively modified. The robot is equipped with a E2V MICS 5524 sensor for measuring the gas concentration, two SHARP 2YOA21F57 proximity sensors for detecting nearby obstacles and an in-house built wind vane, which is able to sense the wind direction in 45° intervals. Figure 2 depicts a photograph of this robot, as well a schematic pointing out its various components. The robot uses a nodeMCU

ESP8266 to communicate through WiFi with a remote server running the Robot Operating System Framework [12]. The server receives the sensory information from the robot, executes the active controller and returns to the robot a motion command, in the form of a rotation or a linear motion. The STM32 board in the robot interprets the command and performs the low-level motion-control. The ESP8266 is also responsible for interfacing with the sensors, through the ADS1115 analog-to-digital converter.

Fig. 2. (Left) developed robot; (right) schematic of the robot.

4.2 Testing Environments

The robotic simulator presented in [9] is used to evaluate the search strategies during evolution. The simulated arena is a scaled up version of the real world arena, measuring 40 by 30 m and containing no obstacles. A single odour source, modelled according to [13], is placed randomly within a bounded region. Its behaviour is characterised by two parameters: (1) the chemical release rate, which is set at $8.3 \cdot 10^9$ molecules/s; and (2) the filament release rate, which is set accordingly to the intended type of environment. The wind is simulated in a grid that covers the entire arena. The cells of this grid have a square shape, being the width of each cell equal to 7% of the arena's width. At each simulation step (which is set to 0.5 s), a wind vector is computed for each vertex of this grid. A Gaussian noise is then added to each vector, emulating the random phenomena of turbulence. Modifying the standard deviation of this Gaussian distribution, it is possible to achieve different levels of wind stability. The standard deviation, along with the filament emission rate and the initial wind velocity are used to create diverse environmental conditions, as described in the Sect. 4.3. The real world test arena (Fig. 3) is a 4 by 3 m enclosed rectangular environment with 50 cm of height. The walls along its length are made of plywood, whereas the

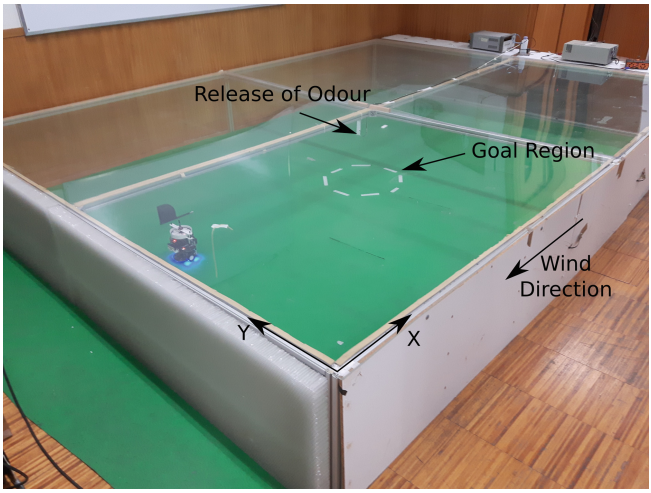


Fig. 3. Validation arena.

others are made of a honeycomb mesh to reduce turbulence. Behind the mesh, on one of the walls, a set of fans are mounted to create air flow with a mean speed of 1.6 m/s. A single odour source is created using an air pump and a bubbler with a 96% solution of ethanol. The bubbler is connected to a hose that emits the odour at the centre of the arena, i.e., at location (2, 1.5) m. A 25 cm circle centred on the end of this hose is marked on the floor, representing the goal region for the robot. The coordinate system of the arena has its origin in the corner closest to the camera, with the x-axis coinciding with the wall along its length and the y-axis coinciding with the wall along its width.

4.3 Genetic Programming Algorithms

The Genetic Programming algorithms used in this work evolve the robotic search strategies in the form of behaviour trees. Each tree is composed by inner and leaf nodes, both of which contain a main symbol (i.e. the function or action to execute) and a list of parameters required by that symbol. The algorithms work as follows: at each generation, a number *n_offspring* of individuals are created. Each offspring may result from crossover (sub-tree or geometric) from two parents, or be a copy of an existing individual. In either case, the individuals are chosen from the population by tournament selection. The offspring may then be mutated, at which case either its main symbol or a parameter is altered with equal probability. At the end of each generation, a new population is created containing the *elite_size* best individuals from the old population, as well as the best *pop_size-elite_size* new individuals to ensure that the population size remains constant. During preliminary experiments, the algorithms exhibited signs of premature convergence, which was fought by injecting a set of random

and elitist immigrants (in equal number) into the population at each generation. The parameters used by the algorithms are presented on Table 1. The two Genetic Programming algorithms use the same parameters, differing solely in the type of crossover. SGP uses sub-tree crossover, whereas GSynGP uses geometric crossover with one iteration.

Table 1. Parameters of the Genetic Programming algorithms

Parameter	Value	Description
gens	75	Number of generations used
pop_size	50	Size of the population used by the algorithm
n_immigrants	20	Number of immigrants injected per generation
n_offspring	30	Number of offspring created per generation
max_depth	5	Maximum depth of the trees in the initial population
p_cross	0.7	Crossover rate
p_mut	0.3	Mutation rate
tourn_size	2	Size of the tournaments for selecting the parent individuals

The function set used by the GPs contains a set of binary functions devised to make it easier to use the sensors' signals, both from the present and past moments: $F_{set} = \{SO, HDO, PL(t), WS(s), Progn\}$. *Progn* is a sequence function, which executes its two sub-trees in order; SO and HDO respectively inform whether the robot is currently sensing odour and if it has already sensed odour in this trial; $PL(t)$ informs whether the robot has not sensed odour for a period longer than t seconds, $t \in [1, plume\_lost\_time * 0.75]$, where *plume_lost_time* is a predefined threshold, set at 60 s, above which it is considered that the robot has definitively lost the plume; $WS(s)$ returns true if the sensed wind speed is higher than s m/s, $s \in \{0.5, 1.0\}$.

The terminal set used by the GPs contains the elementary behaviours that constitute the strategies of the Silkworm Moth and Dung Beetle, which are adapted from [1]: $T\_set = \{$*Break*, *ZZ(dir, dis, iters, off, term)*, *PCZZ(dir, dis, off, term)*, *SP(s_dir, dis, dis_inc, intvs, iters, term)*$\}$. *Break* stops the robot for a control step and the interpretation of the behaviour tree is resumed from its root; *ZZ* encodes a simple zigzag motion, which is carried out with an offset *off* to a specified direction *dir*, with each linear motion having a given length *dis*. This behaviour is made for a number of iterations *iters* until a termination criteria *term* is met; *PCZZ* encodes an odour-centred zigzag motion similar to that of the dung beetle, where the robot attempts to regain contact with the plume once its lost. It consists of performing straight motions, each with a predefined length *dis* and an offset *off* to a specified direction *dir*. This behaviour is performed until a termination criteria *term* is met; *SP* encodes a rectilinear spiral motion, composed by *intvs* line segments until a termination criteria *term* is met. Each iteration of this motion is composed by three steps: (1) rotating $2 \cdot \pi / intvs$

radians, (2) moving linearly for a given distance *dis* and increment *dis* with an amount *dis_inc*. *s_dir* controls whether the spiral is made to the left (l) or to the right (r). The values of the aforementioned parameters, found through preliminary experimentation, are:

- *term* $\in \{C,\ PL(t = 5),\ PL(t = 40)\}$, where C halts the behaviour when all iterations are completed.
- *dir* $\in \{U,\ D,\ X\}$, where *U, D, X* respectively stand for upwind, downwind and crosswind;
- *off* $\in \{0,\ \pi/3,\ \pi/4,\ \pi/6\}$;
- *dis* $\in \{motion\_length/2,\ motion\_length,\ motion\_length * 2\}$;
- *iters* $\in \{1,\ 2,\ 3\}$;
- *s_dir* $\in \{l,\ r\}$;
- *dis_inc* $\in \{motion\_length * x\}$, where $x \in [0, 1]$;
- *intvs* $\in \{4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10\}$;

where *motion_length* is set to 0.5 m.

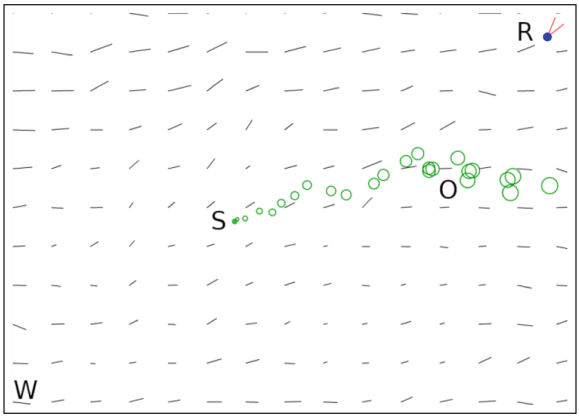


Fig. 4. Screenshot of the second evaluation environment. The wind vectors W are presented as black line segments, the odour source S is the filled green circle, the odour filaments O are empty green circles and the robot R is the filled blue circle with two red lines representing the ranges of its proximity sensors. (Color figure online)

Evaluation Mechanism. Evaluating odour source localisation strategies is not straightforward, as chance may enable a bad strategy to find the source. Moreover, accurately matching the odour and air flow conditions created in simulation and in the real world is not an easy task. For those reasons, an evaluation mechanism was devised to provide a good assessment of a strategy's quality while increasing its robustness to the reality gap. This evaluation mechanism consists of evaluating each strategy in 3 environments, each having different air flow and

odour dispersion patterns. Using distinct sets of environmental conditions should encourage the GPs to find strategies that perform well across various scenarios. The environments are made different by varying the initial wind speed *WS*, the stability of the wind and the rate at which the odour filaments are emitted. The stability of the wind is varied by using different values for the standard deviation (WAV) of the Gaussian noise added to the computed wind vectors (see Sect. 3.1.1 of [9]). As described in Sect. 4.2, the modelled chemical source emits odour in filaments. Two variables regulate odour emission: the chemical emission rate ($\bar{Q}$), which is set at $8.3 \cdot 10^9$ molecules/s and the filament emission rate (FER), which is set differently for each environment. A screenshot of the second evaluation environment is presented in Fig. 4.

Table 2. Environmental parameters

Parameter	Env. 1	Env. 2	Env. 3
WS	0.1 m/s	1.5 m/s	1.5 m/s
Wind direction	0 rad	0 rad	0 rad
WAV	0.3 rad	0.2 rad	0.3 rad
FER	0.05 Hz	0.7 Hz	2.0 Hz
Kx	6	6	6
Arena size	40 m × 30 m	40 m × 30 m	40 m × 30 m
Cell size	2.8 m	2.8 m	2.8 m
Start region	(38 m, 28 m)	(38 m, 2 m)	(30 m, 28 m)
Simulation step	0.5 s	0.5 s	0.5 s
Simulation time	600 s	600 s	600 s

In each of the three evaluation environments the robot departs from a different start region, reducing the possibility of chance enabling bad solutions to find the chemical source. The start position of the robot in each environment is chosen at the beginning of each run and used for all evaluations. The coordinates of those positions are drawn randomly from Gaussian distributions centred on the corresponding coordinates of each start region and with a standard deviation of 0.5. The position of the odour source is also drawn randomly for each trial. Its x-coordinate is drawn from a uniform distribution, ranging between 40% and 45% of the arena's length, while its y-coordinate is between 48% and 53% of the arena's width. The values of the parameters used to create the three environments are presented on Table 2. As previously described, the process of locating odour sources has different stages, each requiring a distinct behaviour. On each evaluation, the search strategy must lead the robot to find and track the odour plume as efficiently as possible. A trial ends successfully when the robot reaches a location within 25 cm from the odour source. Conversely, it ends unsuccessfully if the time limit runs out or if the plume is lost for longer than

plume_lost_time. The fitness of an individual S is given by the mean performance values attained in the three environments, being the performance $F_i(S)$ in environment i computed by Eq. 1.

$$F_i(S) = \begin{cases} \alpha \frac{t_{si}}{T} + (1-\alpha)(\frac{t_{ti}}{T} + \frac{d_i}{D_i}) & \text{if the plume has been found} \\ c & \text{if the plume has not been found} \end{cases} \quad (1)$$

where t_{si} is the time taken to find the plume, t_{ti} is the time spent tracking the plume, T is the evaluation time, d_i is the final distance to the source and D_i is the maximum possible distance to the source in environment i. α and c are constant values which, after preliminary experimentation, were set respectively to 0.5 and 2. This is a minimisation problem.

5 Experimental Results

The first step of this work consisted on devising modular implementations of two well-known strategies from the literature (SM and DB), where each module is a elementary behaviour that can later be used by the evolutionary process, as described in [1]. The two strategies were then optimised using a $(1+1)$ version of the standard Genetic Programming algorithm that relied solely on parameter mutation. Each optimisation trial was given the same amount of evaluations as the population-based approaches (i.e., each trial ran for $pop\_size \cdot gens$ iterations). All of the other parameters are equal to those used by the population-based approaches, which were presented in Sect. 4.3. Thirty independent optimisation trials were made for these approaches, being the performance of the strategies measured with Eq. 1. The population-based versions of SGP and GSynGP also ran for 30 independent trials using the parameters described in Sect. 4.3. The mean performance values and sizes of the best strategies found by each algorithm are presented on Table 3. As can be seen, the bio-inspired strategies attain worse mean performance values than those produced by evolution. The worst performing strategies are produced by SM, with a mean fitness of 0.299, whereas the best strategies are produced by GSynGP, with a mean performance of 0.258. The strategies produced by GSynGP are also the most consistent, having the lowest std. dev. of 0.03. Regarding the sizes of the evolved strategies, those evolved by SGP are on average 17 nodes larger than those produced by

Table 3. Results of the search strategies.

		SM	DB	SGP	GSynGP
Fitness	Mean	0.299	0.285	0.273	0.258
	Std. Dev.	0.044	0.0468	0.044	0.030
Size	Mean	15	5	26.133	9.133
	Std. Dev.	-	-	17.600	8.131

GSynGP. Moreover, the strategies produced by GSynGP can be considered to be of an appropriate size, being only 4 nodes larger than DB and 6 nodes smaller than SM.

Table 4. Results of the Wilcoxon test applied to the fitness values.

	SM-DB	SGP-SM	SGP-DB	SGP-GSynGP	GSynGP-SM	GSynGP-DB
Z	−1.18	−2.29	−1.14	−1.51	−3.30	−2.7
p	0.237	0.022	0.254	0.131	0.001	0.007

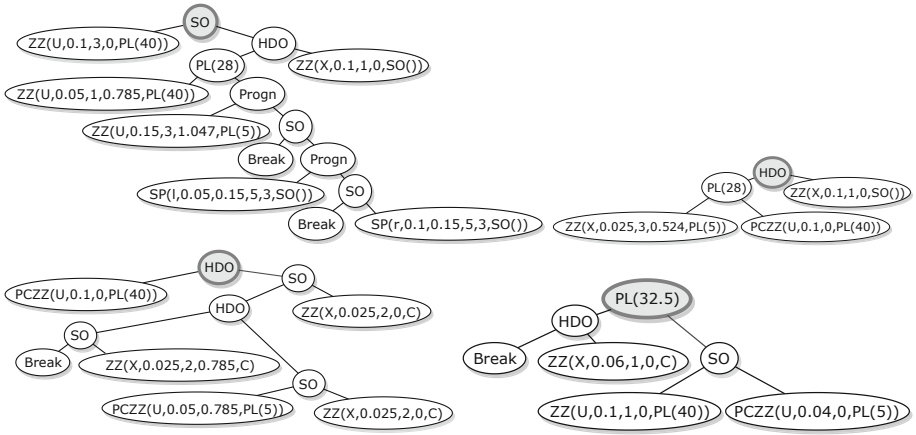


Fig. 5. Overall best strategy for the Silkworm Moth (top-left), Dung Beetle (top-right), SGP (bottom-left) and GSynGP (bottom-right) algorithms. The root nodes are shaded and are drawn with thicker strokes. The parameters of each node follow the same order as presented in Sect. 4.3.

In order to be able to draw more robust conclusions, the results obtained are statistically validated using a confidence interval of 95%. The Kolmogorov-Smirnov test is used to assess the normality of the distributions. Its results show that, at the chosen confidence interval, the fitness values of the DB cannot be considered to follow a normal distribution and, consequently, this analysis must resort to non-parametric tests. The Friedman's Anova is applied to assess whether there are significant differences between the performance of all strategies. It outputs a p-value of 0.009, indicating the presence of statistically significant differences. The next step consists of using the Wilcoxon test to perform pairwise comparisons between the strategies. The Bonferroni correction is used to adjust the significance value to 0.0083. The results of the Wilcoxon test are

presented on Table 4, showing that the two bio-inspired strategies are not significantly different. At the chosen confidence level, the strategies produced by SGP do not outperform those produced by any other algorithm. On the other hand, the strategies produced by GSynGP are found to perform significantly better than those of the two bio-inspired approaches. Analysing the sizes of the trees produced, the Kolmogorov-Smirnov test indicates that the data produced by GSynGP cannot be considered to follow a normal distribution. For that reason, the Wilcoxon test is used to compare the sizes of the strategies. Its results show that the trees evolved by GSynGP are significantly smaller than those produced by SGP ($Z = -4.48$, $p = 0.0$).

5.1 Real World Validation

This section presents the real world validation of the best strategy produced by each algorithm. In the previous simulation experiments, each algorithm produced 30 strategies, one for each run. As the environments used in the evaluation process have stochastic components, it is not possible to directly compare fitness values obtained from different trials. For that reason, the best strategies produced by each algorithm were evaluated in the conditions of the 30 independent trials and the one with the lowest mean fitness value was chosen as the overall best for that algorithm. The overall best strategies are depicted in Fig. 5. The real world validation consists of using the overall best strategy found by each algorithm to control a mobile robot tasked with locating an odour source in the indoor environment described in Sect. 4.2. Due to the real world arena being significantly smaller than the simulated one, the distance parameters of the strategies presented in Fig. 5 are a tenth of the values found in simulation. Similarly to the simulation experiments, each strategy has a maximum evaluation time of 600s and the evaluation is halted if the robot reaches a 25 cm distance from the odour source (goal region) or if it looses contact with the plume for a period longer than 60 s. The robot departs from location (0.5, 0.5) m with its heading set at 0 rad. Odour is emitted at location (2, 1.5) m. Due to the time required for the validation, only 5 trials were made for each strategy. In most experiments, the strategies were able to consistently lead the robot to the vicinity of the odour source within the available time. The two unsuccessful trials occurred when the robot was being controlled by DB and by the best strat-

Table 5. Validation results

Strategy	Time search		Time track		Success rate
	Mean	Std. Dev.	Mean	Std. Dev.	
SM	42.710 s	3.899	44.072s	1.993	5/5
DB	67.757 s	4.672	61.896s	8.979	4/5
SGP	125.801 s	4.358	71.141s	5.411	4/5
GSynGP	60.288 s	3.487	46.876s	3.487	5/5

egy of SGP. Despite not reaching the goal region, the strategies made the robot halt its motion at approximately 35 cm from goal region, remaining still until the end of the trial. From an observers' perspective, all strategies exhibited a plume searching behaviour, moving crosswind while attempting to sense odour. All strategies also exhibited similar plume tracking behaviours, moving upwind after sensing odour. The main difference between the behaviours exhibited was the speed, as the strategies produced by SGP and DB were considerably slower than the others due to making shorter motions. These short motions also meant that the robot did not go as deep into the plume before starting the tracking stage. As a result, the robot is more likely to loose contact with the odour plume when controlled by the SGP and DB strategies than when using the GSynGP and SM strategies. While this may not have been an issue in simulation, the anemometer used in the real world experiments has a resolution of only 45°, making it possible for the robot to unintentionally move diagonally to upwind and thus increasing the chance of losing the plume. The time periods taken by the strategies to find and track the odour plume are presented in Table 5. As can be seen, the SM is the fastest strategy, requiring an average of 42.71 s to find the odour plume and 44.072 s to reach the vicinity of its source. The strategy produced by GSynGP is the second fastest, requiring 60.288 s to find the plume and 46.876 s to reach the goal region. The DB and SGP strategies are the slowest, being their average times affected by the trials where the goal was not reached and the robot remained still until the end of the trial. The larger standard deviations are also indications of this.

6 Conclusions and Future Work

This paper presented an extension to Geometric Syntactic Genetic Programming that enables it to evolve expression trees with multiple symbols per node. This method is applied to evolve search strategies for a mobile robot tasked with locating an odour source. The search strategies produced are compared to those evolved by a standard Genetic Programming algorithm and to two bio-inspired search strategies from the literature. The statistically validated results show that the extended GSynGP is able to evolve solutions that are significantly smaller than those produced by SGP without loss of performance. Moreover, the results show that GSynGP is able to outperform the two strategies from the literature, while the SGP can only match their performance. The best strategy of each algorithm was validated by controlling a real robot attempting to locate an odour source in an indoor environment. In this experiment, GSynGP and SM managed to successfully locate the odour source in all trials, whereas DB and SGP failed 1 of the 5 trials conducted. Furthermore, the SM was the fastest to find and track the plume, followed by GSynGP, whereas SGP required the most time.

In the future, efforts should be made to encourage behavioural diversity as a means to counteract premature convergence and, possibly, achieve better quality solutions. The strategies should also be encouraged to re-encounter the odour

plume rather than simply halting the motion after loosing contact with the odour. Finally, efforts should be made to quantify and reduce the reality gap, so that strategies perform more similarly both in simulation and in the real world.

Acknowledgement. J. Macedo acknowledges the Portuguese Foundation for Science and Technology (FCT) for Ph.D. studentship SFRH/BD/129673/2017. This work was supported by national funds of FCT/MCTES under projects UID/EEA/00048/2019 and UID/CEC/00326/2019, and it is based upon work from COST Action CA15140: Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO).

References

1. Macedo, J., Marques, L., Costa, E.: 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 (2019)
2. Russell, R.A., Bab-Hadiashar, A., Shepherd, R.L., Wallace, G.G.: A comparison of reactive robot chemotaxis algorithms. Robot. Auton. Syst. **45**(2), 83–97 (2003)
3. Harvey, D.J., Lu, T.F., Keller, M.A.: Comparing insect-inspired chemical plume tracking algorithms using a mobile robot. IEEE Trans. Robot. **24**(2), 307–317 (2008)
4. Nolfi, S., Floreano, D., Floreano, D.D.: Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-organizing Machines. MIT Press, Cambridge (2000)
5. Koza, J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection, vol. 1. MIT Press, Cambridge (1992)
6. Silva, S., Costa, E.: Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. Genet. Program Evolvable Mach. **10**(2), 141–179 (2009)
7. Macedo, J., Fonseca, C.M., Costa, E.: Geometric crossover in syntactic space. In: Castelli, M., Sekanina, L., Zhang, M., Cagnoni, S., García-Sánchez, P. (eds.) EuroGP 2018. LNCS, vol. 10781, pp. 237–252. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77553-1_15
8. Marques, L., Nunes, U., de Almeida, A.T.: Particle swarm-based olfactory guided search. Auton. Robots **20**(3), 277–287 (2006)
9. Macedo, J., Marques, L., Costa, E.: A comparative study of bio-inspired odour source localisation strategies from the state-action perspective. Sensors **19**(10), 2231 (2019)
10. Villarreal, B.L., Olague, G., Gordillo, J.L.: Synthesis of odor tracking algorithms with genetic programming. Neurocomputing **175**, 1019–1032 (2016)
11. Moraglio, A.: Towards a geometric unification of evolutionary algorithms (2007)
12. Quigley, M., et al.: ROS: an open-source Robot Operating System. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) Workshop on Open Source Robotics, Kobe, Japan, May 2009
13. Farrell, J.A., Murlis, J., Long, X., Li, W., Cardé, R.T.: Filament-based atmospheric dispersion model to achieve short time-scale structure of odor plumes. Environ. Fluid Mech. **2**(1), 143–169 (2002)