PSO-based Search Rules for Aerial Swarms Against Unexplored Vector Fields via Genetic Programming

Palina Bartashevich[1(✉)], Illya Bakurov[2], Sanaz Mostaghim[1], and Leonardo Vanneschi[2]

[1] Faculty of Computer Science, University of Magdeburg, Germany,
palina.bartashevich@ovgu.de, sanaz.mostaghim@ovgu.de,
[2] NOVA IMS, Universidade Nova de Lisboa, 1070-312, Lisboa, Portugal,
ibakurov@novaims.unl.pt, lvanneschi@novaims.unl.pt

Abstract. In this paper, we study Particle Swarm Optimization (PSO) as a collective search mechanism for individuals (such as aerial micro-robots) which are supposed to search in environments with unknown external dynamics. In order to deal with the unknown disturbance, we present new PSO equations which are evolved using Genetic Programming (GP) with a semantically diverse starting population, seeded by the Evolutionary Demes Despeciation Algorithm (EDDA), that generalizes better than standard GP in the presence of unknown dynamics. The analysis of the evolved equations shows that with only small modifications in the velocity equation, PSO can achieve collective search behavior while being unaware of the dynamic external environment, mimicking the zigzag upwind flights of birds towards the food source.

Keywords: Particle Swarm Optimization, Vector Fields, Semantics, Genetic Programming, EDDA

1 Introduction

This paper considers the Vector Field PSO (VF-PSO) algorithm [2], which is supposed to be used as a collective search mechanism for a swarm of aerial micro-robots acting under the influence of external unknown dynamics (such as wind) performed by vector fields. The main challenge is that the external dynamics of the environment are unknown to the swarm. As a result, due to the influence of unknown external factors, the velocity vectors of the individuals (e.g. robots) are constantly influenced by the external dynamics, and therefore the whole process of the collective search is misled. A previous study [2] suggested the use of a multi-swarm approach and collection of information about unknown dynamics by an explorer population, while another swarm, called optimizer, uses this information to correct their movements during the search process. However, maintaining explorers in some environments might not be possible, e.g. sensors not working under certain conditions, or loss of the connection between explorers

and optimizers, so they can not access the collected information (which is a realistic assumption for aerial robotic systems).

The goal of this paper is to find out whether it is possible to obtain a reasonably good approximation of the global optimal solution using only PSO equations in complete unawareness of the vector fields structure without explorer population, and how such velocity equations (further denoted as VFPS) should be designed in order to show the collective resistance to the unknown external dynamics. To answer these questions, we refer to previous research [1], which has only investigated the possibility of evolving such particle swarm equations using Geometric Semantic Genetic Programming (GSGP) [13]. GSGP has recently attracted much attention in the GP community due to its operators which in contrast to the traditional ones tend to be more effective as they induce a unimodal error surface for any supervised learning problem [21]. However, in [1] it was indicated that using pure GSGP for evolving new PSO equations is not as efficient as using standard Genetic Programming (GP), while the mixture of the GSGP and GP mutation operators was shown to be beneficial to produce high-quality individuals in the presence of unknown dynamics. The mixture of the above mentioned mutation operators in [1] was simulated by the Evolutionary Demes Despeciation Algorithm (EDDA) [20].

In this work, we use the findings of [1] to generate better VFPS equations, which are more robust to the unknown external dynamics than the standard PSO velocity equation. The study performed in this paper differs from [1], as the key aspect of the current work is the analysis and study of the evolved equations themselves and not of the evolutionary process of getting these equations. Besides, for the evolution of the equations, we use standard GP with EDDA on its top only as initialization technique to seed a better GP run.

So far, several applications of the GP to evolve new search algorithms have been already studied in the literature. The Extended Particle Swarms (XPSO) project [16, 15, 5, 11] demonstrated that by using GP it is possible to automatically evolve new PSO algorithms that perform better than standard ones designed by humans. Besides the framework of the XPSO project, some work regarding the evolution of PSO structures was also carried out by Dioşan and Oltean in [6] and [7]. Several studies have also applied GP to investigate other population-based metaheuristic optimizations apart from PSO: for instance, Runka et al. [17] and Taveres et al. [18] applied GP to evolve probabilistic rules used in Ant Colony optimization [8] to update its pheromone trails, and Di Chio et al. [4] used GP to evolve particle swarm equations for group-foraging problems in the simulation of behavioral ecology problems.

The paper is organized as follows. We describe the background about EDDA and VF-PSO in Section 2. In Section 3, we replicate and provide more detailed descriptions of the semantics introduced in [1] that allows us to use EDDA for the evolution of VFPS equations. Section 4 presents the experimental settings and Section 5 along with Section 6 discusses the obtained VFPS equations. The paper is concluded in Section 7.

2 Background

Evolutionary Demes Despeciation Algorithm EDDA [20] is developed as a biologically inspired semantics-based initialization technique for GSGP to create not only a syntactically but also a semantically diverse starting population. According to EDDA, the initial population is seeded with good quality individuals that have been previously evolved for few generations in other populations (called demes). For instance, a population of N individuals will be composed of the best individuals found by N different demes, which are evolved independently by using different operators. In [20] EDDA was applied to seed GSGP runs, which generated solutions with comparable or even better generalization ability and of significantly smaller size compared to traditional GSGP, where part of the demes was evolved using operators of standard GP, while the another using GSGP. However, according to [21], with only Geometric Semantic mutation (further denoted as GSM) it is already possible to obtain the same performance as using GSGP with both crossover and mutation operators and in some cases even outperform it. Thus, in this paper we use EDDA to seed standard GP, where the GSGP part of EDDA demes is evolved using only GSM in order to keep the individuals of reasonable size. A definition of the term semantics used in this paper is described in Section 3, taking into account the fact that we are developing an application aimed at evolving search algorithms.

Vector Field PSO VF-PSO is a collective search mechanism based on the movement of a population of particles, motivated by the real case scenario of aerial micro-robots, acting in an n-dimensional search space S under the influence of unknown dynamic conditions (e.g. wind influence). It performs a variation of the standard PSO algorithm [10], which is based on two simple rules for updating the particles i velocity $\boldsymbol{v}_i(t)$ and its corresponding position $\boldsymbol{x}_i(t) \in S$ at time step t:

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) + \sum_{k=0}^{K} \boldsymbol{VF}(\mathbf{g}^k) \tag{1}$$

$$\boldsymbol{v}_i(t+1) = w\boldsymbol{v}_i(t) + c_1\boldsymbol{\phi_1}(\boldsymbol{x}_i^{pbest}(t) - \boldsymbol{x}_i(t)) + c_2\boldsymbol{\phi_2}(\boldsymbol{x}^g(t) - \boldsymbol{x}_i(t)) \tag{2}$$

The only difference from standard PSO is the additional term in Equation 1, which incorporates vector fields $\boldsymbol{VF}$ to induce the unknown external conditions in the search space S. According to the definition, a vector field is a function that takes any point in the space $\boldsymbol{x} \in S$ and assigns a vector $\boldsymbol{VF}(\boldsymbol{x})$ to it: $\boldsymbol{x} \mapsto \boldsymbol{VF}(\boldsymbol{x})$. In a discrete setting, a vector field is defined on the the grid of cells $\{\mathbf{g}^k\}_{k=1}^{M} \in G \subset S$, where for each cell $\mathbf{g}^k, k \in \{1..M\}$ an associated vector exists as a piecewise constant field $\boldsymbol{VF}(\mathbf{g}^k)$. Following this, the sum of vectors at $K << M$ cells, which particle i intersects along the movement from its previous position $\boldsymbol{x}_i(t)$ to the next $\boldsymbol{x}_i(t+1)$, is added to the current position of the particle to simulate the drift in Equation 1, where $\boldsymbol{x}_i(t) \in \mathbf{g}^0$ and $\boldsymbol{x}_i(t+1) \in \mathbf{g}^K$. More description on the other terms used in the equations described above is provided in the first part of the following Section 3.

3 Semantics for VFPS Evolution in EDDA

As for the evolution of VFPS equations in this work we use standard GP, but with EDDA for the initialization, we have to take into consideration that EDDA uses in the evolution part of the demes which is evolved by GSM. Thus, we have to introduce corresponding semantics. In order to do this, in this section we extend and provide more detailed description of the semantics for VFPS evolution, which was first introduced in [1].

Referring to Pawlak et al. [14], in GP semantics is typically contextualized within a specific *programming task* that is to be solved in a given *program set* P. Thus, in order to introduce the definition of semantics used in this paper, we have to define what the *program set* P and the *programming task* are in our case.

Program Set In our case, we consider GP individuals as acceleration vectors $\boldsymbol{a}_i$ of the Equation 2, where $\boldsymbol{a}_i(t) = \boldsymbol{v}_i(t+1) - w\boldsymbol{v}_i(t)$. To define the program set P, we must specify the set of terminal symbols $\mathcal{T}$ and the set of primitive functions $\mathcal{F}$ used to code this type of individual, i.e. $\boldsymbol{a}_i$.

According to Equation 2, an acceleration function $\boldsymbol{a}_i$ can be considered as a composition of three atomic elements: the current positions of the particles $\boldsymbol{x}_i$, the local best positions of the particles $\boldsymbol{x}_i^{pbest}$ and the global best of the swarm $\boldsymbol{x}^g$. These three elements are part of the terminal set $\mathcal{T}$. Furthermore, we are also interested in the "old" velocity vector of the particle, i.e. $\boldsymbol{v}_i(t)$, in the sense of frictional force and its possible combinations with constants, like in air drag or fluid friction. According to Long et al. [12], for small particles air resistance is approximately proportional to their velocities $\boldsymbol{v}_i$ and can be expressed in the form: $\boldsymbol{F}_{drag} = -b\boldsymbol{v}_i$ or $\boldsymbol{F}_{drag} = -b\boldsymbol{v}_i^2$, where b is a constant that depends on the properties of the particular type of air or fluid. Additionally, we also consider the center of the swarm $\boldsymbol{x^c}$ and its diversity σ_x in the terminal set, as in [15], along with the limited set of permissible constants $C \in \{-0.5, 0.5\}$, as well as a set of random vectors $\boldsymbol{R}(0,1)$, each of which contains uniformly distributed numbers different for each dimension, within the range $[0,1]$. So, our terminal set is: $\mathcal{T} = \{\boldsymbol{x}_i, \boldsymbol{v}_i, \boldsymbol{x}_i^{pbest}, \boldsymbol{x}^g, \boldsymbol{x^c}, \sigma_x, C, \boldsymbol{R}(0,1)\}$, where $\boldsymbol{x}_i$ is the current position of the particle i, $\boldsymbol{v}_i$ is its "old" velocity, $\boldsymbol{x}_i^{pbest}$ is the best location previously visited by particle i, $\boldsymbol{x}^g$ is the best location visited by the entire swarm and σ_x is the average distance of each particle $\boldsymbol{x_i}$ to the center of mass $\boldsymbol{x^c}$.

It is worth pointing out that Equation 2 contains two different random vectors $\boldsymbol{\phi_1}$ and $\boldsymbol{\phi_2}$, which are reflected in the terminal set $\mathcal{T}$ by the set of random vectors $\boldsymbol{R}(0,1)$. The role of $\boldsymbol{\phi_1}$ and $\boldsymbol{\phi_2}$ in PSO is to diversify the particles, keeping them from moving exactly towards the global $\boldsymbol{x}^g$ and personal best $\boldsymbol{x}_i^{pbest}$ positions. On the other hand, previous studies [9, 22, 3] showed that the iterated multiplication of random factors $\boldsymbol{\phi_1}$ and $\boldsymbol{\phi_2}$ can also lead to delay in convergence and attraction to inappropriate directions, dissimilar direction changes in different vectors and, as the result, to a limitation in the particle movements.

Moreover, among all the evolved acceleration equations reported by Poli et al. [16, 15, 5], only one has three different random vectors – $\boldsymbol{R}_1, \boldsymbol{R}_2, \boldsymbol{R}_3$. This

equation is: $\boldsymbol{R}_1(\boldsymbol{x}^g - \boldsymbol{x}_i) - 0.75\boldsymbol{R}_2\boldsymbol{R}_1\boldsymbol{x}_i\boldsymbol{x}_g^2 - 0.25\boldsymbol{R}_3\boldsymbol{R}_2\boldsymbol{R}_1\boldsymbol{x}_i\boldsymbol{x}_g$. All the other provided individuals have mostly none or only one random vector $\boldsymbol{R}$. Considering that the probability of more frequent usage of random vectors in constructing VFPS equations by means of GSM is increased due to the larger size of the evolved individuals, we limit the number of different random vectors in the terminal set up to three: $\{\boldsymbol{R}_1, \boldsymbol{R}_2, \boldsymbol{R}_3\} \in \boldsymbol{R}(0, 1)$.

Function set is introduced as: $\mathcal{F} = \{+, -, *, sin\sphericalangle, cos\sphericalangle, \langle\cdot,\cdot\rangle, \times, LF\}$, where, given vectors $\boldsymbol{e_1} = (e_1^1, e_1^2)$ and $\boldsymbol{e_2} = (e_2^1, e_2^2)$, $*$ is the element-by-element multiplication of two vectors, i.e as the result we get another vector $\boldsymbol{e_1} * \boldsymbol{e_2} = (e_1^1 * e_2^1, e_1^2 * e_2^2)$; $\langle\cdot,\cdot\rangle$ is the dot product of two vectors, i.e. the result is a scalar equal to $\langle \boldsymbol{e_1}, \boldsymbol{e_2}\rangle = e_1^1 * e_2^1 + e_1^2 * e_2^2$; $\times$ is the cross product of two vectors, i.e. the result is a scalar equal to $\boldsymbol{e_1} \times \boldsymbol{e_2} = ||\boldsymbol{e_1}|| * ||\boldsymbol{e_2}|| * sin\sphericalangle(\boldsymbol{e_1}, \boldsymbol{e_2})$, where $||\boldsymbol{e_1}|| = \sqrt{(e_1^1)^2 + (e_1^2)^2}$ is the magnitude of the corresponding vector; $cos\sphericalangle$ is a cosine of the angle between two vectors calculated as $cos\sphericalangle(\boldsymbol{e_1}, \boldsymbol{e_2}) = \frac{\boldsymbol{e_1}\cdot\boldsymbol{e_2}}{||\boldsymbol{e_1}||*||\boldsymbol{e_2}||}$, so $sin\sphericalangle(\boldsymbol{e_1}, \boldsymbol{e_2}) = \sqrt{1 - cos\sphericalangle(\boldsymbol{e_1}, \boldsymbol{e_2})^2}$; and, finally, LF performs a logistic function applied to each component of the vector $\boldsymbol{e}$, i.e. $LF(\boldsymbol{e}) = (\frac{1}{1+exp(-e^1)}, \frac{1}{1+exp(-e^2)})$.

Programming task According to the definition in [14], the programming task, usually denoted as (FC, f), is defined by a set of fitness cases $FC \subseteq X \times O$ and a fitness function $f : \mathcal{S} \to \mathbb{R}_{\geq 0}$. A *fitness case* FC is a pair consisting in a program input $in \in X$ and a corresponding output $out \in O$. In this sense, FC represents the *training set* of the programming task. Figure 1 describes the programming task used in this paper. In our case, *fitness case* FC represents a

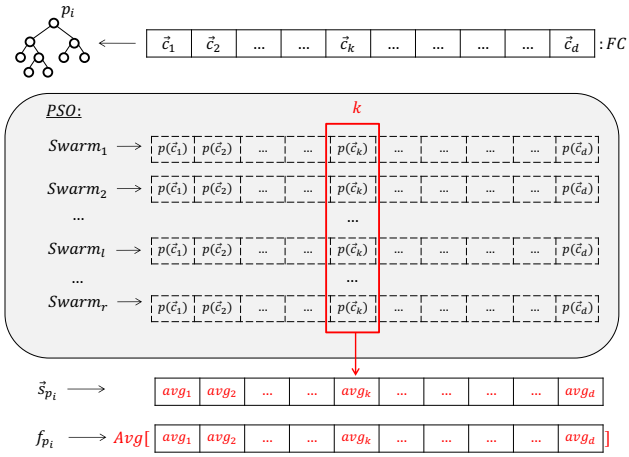


Fig. 1: Semantics $\boldsymbol{s}_p$ of the individual p on a problem class FC represented as a vector of average outputs calculated for each of the problem instances $\{c_j\}_{j=1}^d := FC$ within r runs, where $Swarm_l$ denotes the fixed initial population within the corresponding run for all problem instances. Fitness function $\boldsymbol{f}_p$ of individual p is performed as the average of the elements in its semantics vector $\boldsymbol{s}_p$.

certain optimization function $h(\boldsymbol{x} - \boldsymbol{c}) : \mathbb{R}^n \to \mathbb{R}$, where $\boldsymbol{c}$ is a corresponding

optimum solution. As an input for the program p_i, we consider a d-dimensional vector $\{\boldsymbol{c}_j\}_{j=1}^{d} \in \mathbb{R}^n, \boldsymbol{c}_j \neq \boldsymbol{c}_i \neq 0$ of different global optimal solutions, representing a class of shifted functions $FC := \{h(\boldsymbol{x} - \boldsymbol{c}_j)\}_{j=1}^{d}$. So, an output is the vector of respective function values at the global best position x_j^g found using $p_i : \boldsymbol{c}_j \mapsto h(\boldsymbol{x}_j^g - \boldsymbol{c}_j), \forall j \in \{1, .., d\}$. Evaluation of p_i inside VF-PSO lasts for r runs on each element of the input vector $\{\boldsymbol{c}_j\}_{j=1}^{d}$. The average of the outputs within runs $avg_j = \sum_{1}^{r} p_i(\boldsymbol{c}_j), \forall j \in \{1, .., d\}$ defines the elements of the vector $\boldsymbol{s}_{p_i}$, which represents a point in the semantic space $\mathcal{S}$ and *the semantics* of an individual p_i on a problem class FC. Without loss of generality, in this paper we considered problem classes, whose function values at the global optimum are equal to zero (see Section 4). Thus, as the *target* is a zero vector $\boldsymbol{t} \in \mathbb{R}^d$, the *fitness* for individual p_i in our case is the average of the elements of its semantics $\boldsymbol{s}_{p_i}$: $f_{p_i} = \frac{1}{d} \sum_{k=1}^{d} [\boldsymbol{s}_{p_i}(k) - \boldsymbol{t}(k)] = \frac{1}{d} \sum_{k=1}^{d} \boldsymbol{s}_{p_i}(k)$.

4 Experimental Study

The objective of the experimental study is to evolve and to analyze new VFPS equations which are able to find the approximate global optimum solution in total unawareness of the external dynamics. Before discussing the experimental results, let us briefly present the experimental settings.

Experimental VF-PSO settings Following [16, 15, 5, 2], in our experiments we consider the following problem classes: $FC^1 - Sphere$, $FC^2 - Rosenbrock$ and $FC^3 - Ackley$. As in [2], the vector fields are considered in the two-dimensional search space $S : [-15.0, 15.0] \times [-15.0, 15.0]$ and their function descriptions can be found in Table 1. Every problem instance was considered in combination with each of the five vector fields (VF). Additionally, we also consider the case without vector field (denoted further as VF0). Given that the influence of the

Table 1: Function descriptions of the vector fields

"Cross"	$\boldsymbol{VF1}(x_1, x_2) = (x_2, x_1)$
"Rotation"	$\boldsymbol{VF2}(x_1, x_2) = (-x_2, x_1)$
"Sheared"	$\boldsymbol{VF3}(x_1, x_2) = (x_1 + x_2, x_2)$
"Wave"	$\boldsymbol{VF4}(x_1, x_2) = (-\sin(x_2), \cos(x_1 \cdot x_2 - x_1^2))$
"Tornado"	$\boldsymbol{VF5}(x_1, x_2) = (-x_1 - x_2, x_1)$

considered vector fields is weaker near the origin of the Cartesian system (see the vector fields descriptions in Table 1), we shifted the global optimum for each of the problem instances to the left upper corner of the search space S, namely to the region $\Omega : [-11.0, -9.0] \times [9.0, 11.0]$. We define $d = 10$ problem instances in each problem class $FC^i : \{h^i(\boldsymbol{x} - \boldsymbol{c}_j)\}_{j=1}^{d}$, where $\boldsymbol{c}_j$ is a random vector from the given interval Ω and $h^i(\boldsymbol{x})$ is the objective function of the corresponding problem class FC^i. In the training set, we use VF-PSO with 10 particles, whose initial positions are chosen uniformly at random in the whole search space S. Initial velocity is set to 0. The components of the velocity vector are constrained

within $\boldsymbol{v} \in [-2.0, +2.0]$. The inertia weight w is equal to 0.6. Acceleration coefficients are $C_1, C_2 = 1$. Each VFPS has been run for $N_{max} = 30$ iterations on each problem instance for $r = 5$ times. In the testing phase, the number of runs was increased up to $r = 100$ with $N_{max} = 50$ each. The population was enlarged up to 20 particles and the global optimal solution was fixed at $c = (-10.0, 10.0)$ for all problem classes. The results of the testing phase are reported in Section 6 and compared in terms of the best function values obtained during the all iterations, and the *success rate*, which indicates the percentage of runs, where this fitness is smaller than a certain threshold ϵ by the end of the search process. We used $\epsilon = 0.1$.

Experimental EDDA and GP settings For EDDA we used 100 demes, each of which containing 100 individuals. The individuals in each deme were themselves initialized by means of the ramped half-and-half method, with maximum initial depth equal to 3. After initialization, each deme was left to evolve for 5 generations using a given set of genetic operators. In EDDA, $m\%$ of demes use GSM (GSGP demes), while the remaining $(100 - m\%)$ use standard genetic operators (GP demes). In our experiments, we test $m \in \{25, 50, 75\}$. A maximum depth limit of 5 is imposed only during the evolution of the GP demes, while in the main evolutionary process (MEP) this limit was enlarged to 11. After 5 generations, the best individual is selected and copied into the population that constitutes the MEP. The mutation step ms of GSM in the GSGP demes was randomly generated with uniform probability in [0,1] at each mutation event, following [19]. The codomain of the possible outputs of the randomly generated trees in GSM was bounded in $[0, 1]$ by wrapping them inside a logistic function, as in [19]. To select parents for variation, tournaments of 5% of the population size were used and survival was elitist, as it always copied the best individual into the next population. While evolving VFPS in the GP demes and in MEP, the probability of applying crossover and mutation was set to 0.9 and 0.1 respectively.

5 Evolved VFPS

In this section, we present and discuss the best 5 force VFPS equations, that we were able to evolve, in terms of the performance on the training set. All these equations were obtained using EDDA-50% as an initialization technique, and standard GP in the main evolutionary process.

VFPS1 was evolved on the "Sphere-VF1" (Cross) problem:

$$\boldsymbol{a}_i = \boldsymbol{R}_1(\boldsymbol{x}_i^{pbest} - \boldsymbol{x}_i) + \sigma_x^2(\boldsymbol{x}^g - \boldsymbol{x}_i) \tag{3}$$

Interestingly, it has a similar structure to the acceleration of the standard PSO (i.e. both cognition and social components are present), but with divergence factor for the deterministic social component. Thus, when the particles are too sparse in a swarm, the social component has more weight, so the particles are more attracted by the global best position than by the personal best local ones.

On the other hand, when the particles are denser, the influence of the social component in the swarm decreases and the particles are more attracted by their respective local best position, reproducing local search behavior.

VFPS2 was evolved on the "Sphere-VF1" (Cross) problem:

$$\boldsymbol{a}_i = (\boldsymbol{R}_1 + (\boldsymbol{x}^g - \boldsymbol{x}_i))\langle \boldsymbol{x}_i^{pbest} + \boldsymbol{v}_i, \boldsymbol{x}_i^{pbest} + \boldsymbol{v}_i \rangle \tag{4}$$

It includes a separate independent random component, which is added to the deterministic social component, while their sum is weighted by the squared length value of $(x_i^{pbest} + v_i)$ vector, obtained as its dot product by itself.

VFPS3 was evolved on the "Sphere-VF3" (Sheared) problem:

$$\boldsymbol{a}_i = \sigma_x \left((\boldsymbol{x}_i^{pbest} - \boldsymbol{x}_i) + (\boldsymbol{v}_i + \sigma_x)(\boldsymbol{x}^g - \boldsymbol{x}_i) \right) \tag{5}$$

This equation is interesting, because it is completely deterministic, and both cognition and social components are present, as in standard PSO. Similarly to **VFPS1**, it contains a divergence factor, but contrarily to **VFPS1**, this factor influences both the cognition and social components.

VFPS4 was evolved on the "Sphere-VF1" (Cross) problem:

$$\boldsymbol{a}_i = \boldsymbol{R}_1(\boldsymbol{x}_i^{pbest} - \boldsymbol{x}_i) + \boldsymbol{R}_2(\sigma_x + \boldsymbol{R}_2)(\boldsymbol{x}^g - \boldsymbol{x}_i) \tag{6}$$

It contains two random standard PSO components, along with a divergence factor for the social component. It is expected to behave similarly to **VFPS1**, but with the difference that its behavior should be more like the one of standard PSO when the swarm has a high density (i.e. small σ_x values).

VFPS5 was evolved on the "Sphere-VF1" (Cross) problem

$$\boldsymbol{a}_i = \langle \boldsymbol{x}_i, 0.5\sigma_x \rangle((\boldsymbol{x}_i^{pbest} - \boldsymbol{x}_i) + \boldsymbol{x}^c + \sigma_x \boldsymbol{x}^g) \tag{7}$$

Contrarily to the other reported equations, it is completely deterministic and contains both the center and the spread of the swarm.

6 Analysis and Discussion of the Evolved VFPS

Median values over 100 runs and corresponding standard errors obtained by these five VFPS equations during the test phase, and the ones obtained by standard PSO, are reported in Table 2. The Kolmogorov-Smirnov non-parametric test has been performed to analyze the statistical significance under the alternative hypothesis that VFPS and PSO results are drawn from the same distribution, with significance level $p = 0.05$. The results of the success rate for each VFPS are shown in Table 3. As can be seen from Table 2, most of the obtained median fitness values for the reported evolved equations (denoted in each row of Table 2 as -PS) under the influence of vector fields (each column from VF1-VF5 of the corresponding objective function) are significantly smaller than those ones obtained by standard PSO equation (first row of Table 2 for every

column from VF1-VF5). Standard PSO is extremely bad under considered vector fields influence with medium fitnesses $>> 1$ along with almost everywhere a 0% success rate in Table 3. While the median fitnesses of the evolved VFPS are mostly < 1 with more than a 50% success rate on Sphere function and a non-zero success rate on other objective functions under any of the considered VF influence. The only exception is the results obtained on VF4, where PSO is the best performant for Ackley problem, and second best in Sphere and Rosenbrock, with a 100% success rate under VF4 conditions for almost all problems. This particular observation is due to the vector field characteristic, i.e. the values of VF4 according to its description in Table 1 are within $[-1, 1]$, which perform rather small disturbances in comparison to magnitude of other VFs. Table 2 and Table 3 reveal that the evolved VFPS equations can obtain reasonably good approximation of global optimal solution in contrast to standard PSO velocity rule in total unawareness of the external disturbance.

The particles behavior of all the VFPS presented above tend to be resistant to the hard disturbances on unimodal objective functions. Such behavior is obtained since almost all reported VFPS were evolved on the Sphere problem class under VF "Cross" (see Section 5), which is characterized by high intensity vectors redirecting away from the goal. Considering that, according to the findings of previous studies [2], the other considered VFs are not so challenging as VF "Cross", the evolved VFPS are expected to almost always be successful on the Sphere landscape, regardless of the VF type under which they are considered. The results of Table 2 and Table 3 confirm this expectation. Moreover, while analyzing the particles trajectories of the evolved VFPS, one can consider that the particles which are initially placed at the lower left corner of the search space, can now reproduce very "straightforward" movements towards the goal in contrast to standard PSO (as an example see Figure 2) on all problems, despite their landscape structure. That might seem to be beneficial on Ackley, preventing the particles from getting trapped in multiple local optima and moving faster towards the region with the globally optimal solution. However, due to the strong resistance of the evolved VFPS to any appeared disturbance near the goal, they

Table 2: Median and standard error (in brackets) over 100 runs of fitness values found by each VFPS (denoted as -PS) in five vector fields (VF1-VF5) on Sphere, Rosenbrock and Ackley. VF0 indicates the case without vector field. Best results in bold. An asterisk indicates statistical significance ($p < 0.05$) between a result obtained by VFPS and PSO according to non-parametric Kolmogorov-Smirnov test.

	Sphere						Rosenbrock						Ackley					
	VF0	VF1	VF2	VF3	VF4	VF5	VF0	VF1	VF2	VF3	VF4	VF5	VF0	VF1	VF2	VF3	VF4	VF5
PSO	**.000**	13.64	6.49	5.45	.004	4.59	.009	38.302	1695.85	9.072	.110	346.05	**.003**	7.265	3.553	4.554	**.129**	3.864
	(.000)	(17.28)	(7.706)	(5.003)	(.004)	(3.654)	(.004)	(44.25)	(2313.22)	(10.59)	(.156)	(319.79)	(.003)	(.602)	(2.322)	(1.446)	(.052)	(.88)
-PS1	**.000***	.192*	.484*	.089*	**.002**	.353*	**.003**	4.918*	2.405*	2.266*	**.082**	.744*	.031*	2.515*	1.616	2.467*	.262	2.139*
	(.000)	(.159)	(.293)	(.100)	(.001)	(.193)	(.005)	(4.31)	(.422)	(1.384)	(.097)	(.508)	(.028)	(.318)	(1.643)	(1.071)	(.164)	(.407)
-PS2	.092*	.122*	**.036***	.135*	.119	.126*	.848*	.633*	**.590***	1.29*	1.74*	.961*	2.44*	2.33*	1.79	2.92*	2.19*	2.28*
	(.068)	(.081)	(.024)	(.129)	(.128)	(.085)	(.843)	(.482)	(.866)	(.694)	(2.93)	(1.103)	(.571)	(.99)	(1.08)	(.415)	(.890)	(.282)
-PS3	.012*	.120*	.050*	**.038***	.015	**.028***	.052*	**.201***	4.109*	.987*	.188*	**.477***	.214*	1.332*	2.459	3.075*	.499*	**.591***
	(.013)	(.091)	(.030)	(.016)	(.013)	(.032)	(.040)	(.283)	(5.00)	(.203)	(.142)	(.249)	(.105)	(1.227)	(1.41)	(.558)	(.086)	(.529)
-PS4	.096*	.053*	.082*	.091*	.034*	.038*	5.22*	.972*	.624*	**.541***	2.87*	1.53*	2.086*	1.312*	**1.450**	2.744*	2.212*	1.710*
	(.015)	(.016)	(.073)	(.052)	(.020)	(.009)	(5.96)	(.338)	(.473)	(.66)	(2.049)	(2.27)	(1.39)	(.719)	(1.217)	(.175)	(.261)	(1.503)
-PS5	.052*	**.049***	.133*	.088*	.078	.079*	1.588*	.900*	.684*	.729*	.542*	.924*	1.110*	**1.157***	2.251	1.521*	2.456*	1.774*
	(.042)	(.029)	(.099)	(.062)	(.079)	(.037)	(1.705)	(1.116)	(.479)	(.23)	(.798)	(.764)	(1.483)	(.865)	(.409)	(1.207)	(.325)	(.618)

Table 3: Success rate (in %) for 5 evolved VFPS (denoted as -PS) over 100 runs. Best results on each of the VF within considered objective function are in bold.

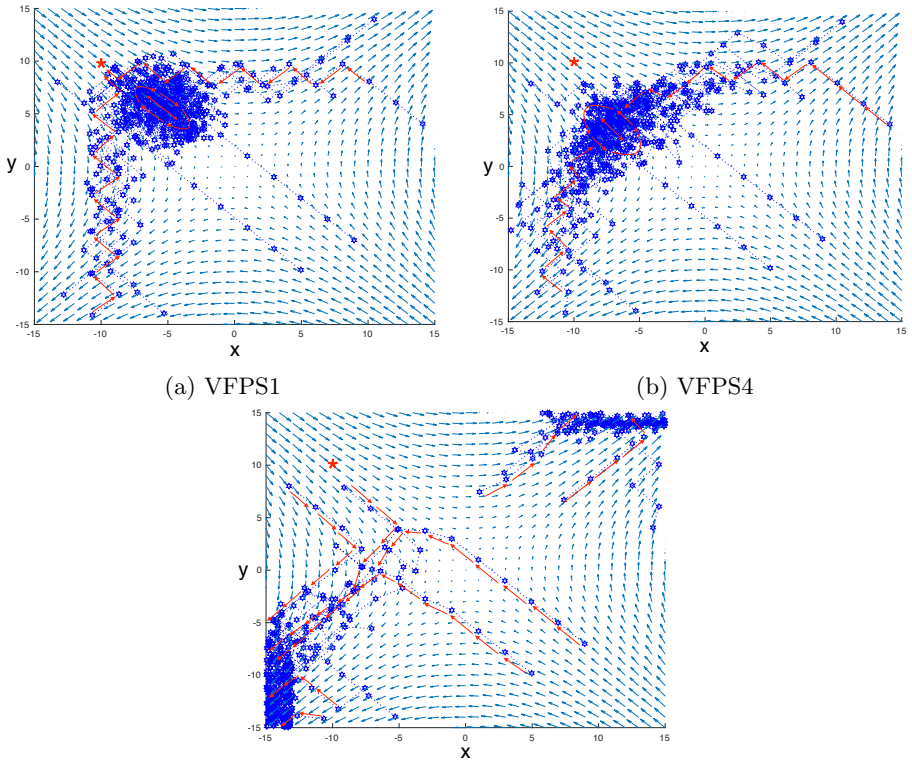
	Sphere						Rosenbrock						Ackley					
	VF0	VF1	VF2	VF3	VF4	VF5	VF0	VF1	VF2	VF3	VF4	VF5	VF0	VF1	VF2	VF3	VF4	VF5
PSO	**100**	3	1	3	**100**	4	99	0	0	0	**100**	0	**100**	0	0	0	53	0
-PS1	**100**	53	53	73	**100**	52	99	10	7	11	**100**	3	**100**	1	1	1	**61**	0
-PS2	86	77	**88**	**93**	82	89	13	24	15	20	24	17	**3**	0	3	1	2	0
-PS3	**100**	**97**	69	82	**100**	**100**	98	**47**	10	17	95	**34**	99	**5**	0	0	26	**17**
-PS4	**100**	5	9	41	**100**	11	**100**	1	0	3	**100**	4	**100**	0	0	1	55	1
-PS5	92	94	**88**	90	94	91	19	13	**30**	**25**	18	17	5	0	0	1	2	3

cannot reach the exact global solution (as supposed on Sphere), being misled by the inherited resistant behavior to the local optima surrounding the global one. This is reflected by the high values in Table 3 and the low ones in Table 2. The only exception might be **VFPS3** (evolved on "Sphere-Sheared"), which performs quite well in case of VF5 in comparison to the other VFPS. And, according to its results for the Sphere and Rosenbrock problems, **VFPS3** seems to be a good all-rounder. Characterized by small intensity vectors in the region around the optimal solution, evolution under "Sheared" VF made the structure of **VFPS3** able to seek the particles towards the goal. On Rosenbrock-VFs, in certain cases VFPS are able to obtain a reasonable approximation of the global optimal solution. As mostly VFPS are based on the swarm diversity, getting into the valley region of Rosenbrock, particles start producing more local search behavior. However, constant redirections by VF prevent them from convergence to their local best solutions, so that they are more likely to reach the global optimum.

It is also worth pointing out that, when we observe the trajectories (e.g. in Figure 2), the VFPS particles, being able to move against the flow, produce zigzag movements (i.e. in Fig. 2a and Fig. 2b), i.e. they behave similarly to a sailboat, which cannot travel directly into the wind but uses a zigzag pattern to move against it, while usual PSO particles are just blown away by the flow (Figure 2c). This is an interesting finding, considering that such behavior is also typical for birds such as the albatross, which perform a zigzag upwind search in response to odor cues towards the food source [23]. Such behavior is mostly obtained by **VFPS1** (Fig. 2a). As soon as the particles are gathered together, they start to reproduce oscillating behavior near the best found by them point, moving together back and forth.

7 Conclusions and Future Work

Analysis of the evolved programs has demonstrated that with small modifications in the velocity rule, PSO can achieve solid collective search behavior in total unawareness of external dynamics, mimicking trajectories of the zigzag upwind birds flights towards the food source. These findings deepen our understanding on the swarm dynamics in the presence of external influence (performed in this study by vector fields) and may shed light on the underlying mechanism of information exchange in natural swarms under dynamic unknown stimuli (e.g.



(a) VFPS1

(b) VFPS4

(c) PSO

Fig. 2: Trajectories of the particles (in blue) starting from the same initial positions on "Ackley-Cross" obtained by VFPS1, VFPS4 and PSO in (a-c). A five-pointed red snowflake at $(-10, 10)$ indicates the unknown target (global optimum). Red arrows show the most characteristic general direction of the particles movement at certain regions.

wind), which might find its further applications for outdoor swarm robotics systems. In the future, we plan to increase the functional and terminal sets of the GP system and to test it on other (i.e. non-PSO) metaheuristic algorithms. Also, we intend to implement our system using a parallel and distributed framework, in order to improve the speed of the overall calculations.

References

1. P. Bartashevich, I. Bakurov, S. Mostaghim, and L. Vanneschi. Evolving PSO Algorithm Design in Vector Fields Using Geometric Semantic GP. In *Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO 2018), Kyoto, July 2018. 2 pages. To appear.*
2. P. Bartashevich, L. Grimaldi, and S. Mostaghim. PSO-based Search mechanism in dynamic environments: Swarms in Vector Fields. In *2017 IEEE Congress on Evolutionary Computation*, pages 1263–1270, 2017.

3. M. Clerc. Stagnation analysis in Particle Swarm Optimization or what happens when nothing happens. Technical report, 2006.
4. C. Di Chio and P. Di Chio. *Group-Foraging with Particle Swarms and Genetic Programming*, pages 331–340. Springer Berlin Heidelberg.
5. C. Di Chio, R. Poli, and W. B Langdon. Evolution of force-generating equations for PSO using GP. In *Proc. of the 2005 AI*IA Workshop on Evolutionary Computation*.
6. L. Dioşan and M. Oltean. *Evolving the Structure of the Particle Swarm Optimization Algorithms*, pages 25–36. Springer Berlin Heidelberg.
7. L. Diosan and M. Oltean. What else is the Evolution of PSO Telling Us? *J. Artif. Evol. App.*, 2008.
8. M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 2006.
9. A. Erskine and J. M. Herrmann. Critical Dynamics in Particle Swarm Optimization. *CoRR*, 2014.
10. J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.
11. W. B. Langdon and R. Poli. Evolving problems to learn about particle swarm optimisers and other search algorithms. *IEEE Transactions on Evolutionary Computation*, 11(5):561–578, 2007.
12. N. L. Lyle and W. Howard. *The Velocity Dependence of Aerodynamic Drag: A Primer for Mathematicians*, pages 127–135. Math. Association of America, 1999.
13. A. Moraglio and K. Krawiec. Semantic Genetic Programming. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 603–627. ACM.
14. T. P. Pawlak, B. Wieloch, and K. Krawiec. Review and comparative analysis of geometric semantic crossovers. *Genetic Programming and Evolvable Machines*, 16:351–386, 2015.
15. R. Poli, C. Di Chio, and W. B. Langdon. Exploring Extended Particle Swarms: A Genetic Programming Approach. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, pages 169–176, New York, USA, 2005.
16. R. Poli, W. B. Langdon, and O. Holland. *Extending Particle Swarm Optimisation via Genetic Programming*, pages 291–300. Springer, 2005.
17. A. Runka. Evolving an Edge Selection Formula for Ant Colony Optimization. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, pages 1075–1082. ACM, 2009.
18. J. Tavares and F. B. Pereira. *Evolving Strategies for Updating Pheromone Trails: A Case Study with the TSP*, pages 523–532. Springer Berlin Heidelberg, 2010.
19. L. Vanneschi. *An Introduction to Geometric Semantic Genetic Programming*, pages 3–42. Springer International Publishing, 2017.
20. L. Vanneschi, I. Bakurov, and M. Castelli. An initialization technique for geometric semantic GP based on demes evolution and despeciation. In *2017 IEEE Congress on Evolutionary Computation*, pages 113–120.
21. L. Vanneschi, S. Silva, M. Castelli, and L. Manzoni. *Geometric Semantic Genetic Programming for Real Life Applications*, pages 191–209. Springer New York, 2014.
22. D. N. Wilke, S. Kok, and A. A. Groenwold. Comparison of linear and classical velocity update rules in particle swarm optimization: notes on scale and frame invariance. *International Journal for Numerical Methods in Engineering*, 70(8):985–1008, 2007.
23. T. Wyatt. *Pheromones and Animal Behavior: Chemical Signals and Signatures*. Cambridge University Press, UK, 2014.