Novelty-driven Particle Swarm Optimization

Diana F. Galvao[1] *, Joel Lehman[2], and Paulo Urbano[1]

[1] University of Lisboa, Faculty of Sciences, BioISI Biosystems and Integrative Sciences Institute, Campo Grande, Lisboa, Portugal
fc37298@alunos.fc.ul.pt,pub@di.fc.ul.pt
[2] IT University of Copenhagen jleh@itu.dk

Abstract. Particle Swarm Optimization (PSO) is a well-known population-based optimization algorithm. Most often it is applied to optimize objective-based fitness functions that reward progress towards a desired objective or behavior. As a result, search increasingly focuses on higher-fitness areas. However, in problems with many local optima, such focus often leads to premature convergence that precludes reaching the intended objective. To remedy this problem in certain types of domains, this paper introduces Novelty-driven Particle Swarm Optimization (NdPSO), which is motivated by the novelty search algorithm in evolutionary computation. In this method particles are driven only towards instances significantly different from those found before. By ignoring the objective this way, NdPSO can circumvent the problem of deceptive local optima. Because novelty search has previously shown potential for solving tasks in genetic programming, this paper implements NdPSO as an extension of the grammatical swarm method, which combines PSO with genetic programming. The resulting NdPSO implementation is tested in three different domains representative of those in which it might provide advantage over objective-driven PSO. That is, deceptive domains in which it is easy to derive a meaningful high-level description of novel behavior. In each of the tested domains NdPSO outperforms both objective-based PSO and random-search, demonstrating its promise as a tool for solving deceptive problems.

Keywords: Particle Swarm Optimization, Novelty Search, Grammatical Evolution, Grammatical Swarm, Deceptive problems

1 Introduction

Particle Swarm Optimization (PSO) is a biologically-inspired population-based optimization algorithm [1]. Although PSO is a popular and effective algorithm, like other population-based methods it is susceptible to converge prematurely to local optima when applied to complex or deceptive problems [2, 3]. Most applications of PSO optimize an objective-based fitness function which estimates the progress to the desired outcome, e.g. minimizing squared error or heuristic

* corresponding author

similarity to a goal behavior. Guiding the search directly towards the ultimate goal causes increasing focus on higher-fitness areas at the expense of lower-fitness ones, reducing the overall exploration of the search space. In simple problems this focus aids efficiency, but can be harmful in deceptive ones. That is, the pervasiveness of local optima may render the objective accessible only by significant travel through areas with low objective-based fitness. By pruning such areas from consideration, an objective-driven algorithm may be unlikely to reach the desired objective.

Because local optima are a well-known issue in search, many researchers have proposed variations of PSO to circumvent premature convergence [4, 5]. While such variations may outperform the standard PSO algorithm in domains with limited deception, because they remain guided by heuristic distance to the objective, they are still vulnerable to premature convergence if the objective function is sufficiently deceptive. In this way, there is a clear relationship between objective-based search and premature convergence. Thus to avoid premature convergence in very deceptive domains it may paradoxically be necessary to guide search without considering the ultimate objective.

Novelty search is an evolutionary algorithm (EA) which takes this radical step [6], and has successfully been applied in neuroevolution [6, 7] and genetic programming [8,9]. The core insight motivating novelty search is that novelty, i.e. demonstrating qualitative difference from previously encountered individuals, is a valuable source of information. Thus instead of guiding search by estimated distance to the search's objective, novelty search is driven towards instances significantly different from those found before. By ignoring the objective completely, novelty search circumvents the problem of deception inherent in objective-based algorithms. Of course, without any pressure to optimize towards the objective, intuitively a raw search for novelty may seem unlikely to be effective at solving problems. Yet if measures of novelty are constructed that capture important dimensions of behavioral variation in the domain, the surprising result is a practical algorithm for solving deceptive problems [6, 8, 10]. The insight is that often demonstrating novelty requires exploiting meaningful regularities in a domain.

A key motivation for this paper is that because PSO is also susceptible to deception, integrating a drive towards novelty might sometimes also benefit the effectiveness of PSO. This paper thus introduces Novelty-driven PSO (NdPSO), a tool to combat the pathology of premature convergence in PSO. Because novelty search has shown prior promise in combination with genetic programming [10], here NdPSO is implemented as an extension of the Grammatical Swarm (GS) method [11], which is a PSO-based version of a GP technique called Grammatical Evolution (GE). This implementation is thus called novelty-driven Grammatical Swarm (NdGS).

Experiments in this paper test NdGS in three domains representative of those for which the algorithm might be most effective. Such domains are deceptive (otherwise an objective-based search method is likely to be more effective) and provide an intuitive way to characterize a space of behaviors that captures important domain features (otherwise it is difficult to quantify novelty). The

first domain requires evolving a hidden sequence obscured by a deceptive fitness function. The other domains are reinforcement learning benchmarks imported from genetic programming that are also known to be deceptive. The experiments compare the performance of NdGS, traditional objective-driven GS and purely random search. Across the tested domains NdPSO performs the best, highlighting its potential for solving deceptive problems.

2 Background

The next sections describe the PSO algorithm used in the experiments, the Grammatical Swarm extension that enables PSO to evolve programs, and the novelty search method merged with PSO in this paper's approach.

2.1 Particle Swarm Optimization

This section reviews the main concepts of PSO, a population-based optimization algorithm inspired by schooling and flocking behaviors of animals, introduced by Kennedy and Eberhard [12].

In PSO, the population (or swarm) is composed of particles moving through a $\mathbb{R}^d$ search space, that optimize a fitness function with the following domain $f : \mathbb{R}^d \to \mathbb{R}$, with d representing the dimensionality of the search space.

Each particle $i \in (1, 2, 3, ..., N)$ is associated with two d-dimensional vectors, one recording its position x and the other its velocity v.

Particles also store a summary of their previous experiences in a simple memory component. In particular, the particle records the position of the maximum fitness value it has encountered, called its personal-best or *pbest*. Particles also share information with each other, and record the point in the search space where the overall best fitness has been obtained among all particles. This point, which a particle may not have visited (but has heard from another particle) is called its global-best *gbest*. These components help balance exploiting promising areas with exploring more broadly [13].

In practice, communication between particles is often restricted by use of a neighborhood topology, meaning that a particle's *gbest* may be calculated from the best search locations recorded by its neighboring particles [14]. The most commonly chosen topology is a fully-connected neighborhood [14]. In such a topology all particles directly communicate with all other particles. Thus information propagates quickly, which can cause fast convergence. In the Ring topology, particles communicate only with their immediate neighbors based on their position in a circular list, causing information to flow more slowly than in the Fully-connected topology. The result is that some groups of particles can converge to one point in the search space, while other groups can converge to others. The Von Neumann topology offers an intermediate speed of information flow; each particle is assigned a fixed location on a two-dimensional toroidal neighborhood grid, and can communicate with its nearest neighbors in the four cardinal directions.

At each time step t, the velocity of each particle is adjusted as a function of its position, previous velocity, *pbest* and *gbest*. In particular, the i^{th} particle's velocity is updated as follows:

$$v_i(t+1) = \omega.v_i(t) + \varphi_1.r_1(pbest_i(t) - x_i(t)) + \varphi_2.r_2(gbest(t) - x_i(t)) \ , \quad (1)$$

where ω is a parameter specifying the particle's inertia, which determines how strongly the particle maintains its previous velocity (i.e. the higher the inertia, the slower that velocity changes). The experiments apply a dynamic inertia value, which is initialized to 0.9 and during search decreases linearly to 0.4 using Equation 3. The real numbers r_1 and r_2 are chosen randomly within an interval (typically between 0 and 1), and φ_1 and φ_2 are the acceleration coefficients. It is also common to restrict the maximum velocity ($v_{max} \in [-v_{max}, v_{max}]$) to prevent instability. Note that the particle's maximum velocity may be static, as in our experiments, or calculated dynamically [15]. The particle's new position is calculated according to Equation 2.

$$x_i(t+1) = x_i(t) + v_i(t+1) \ . \quad (2) \qquad \omega = \omega_{max} - \frac{\omega_{min}}{maxIterations} * NumIteration \ , \quad (3)$$

In this way, the particles are driven through the search space towards locally optimal points. Because they are attracted both to their own best position and the overall best position, over time a consensus may emerge as knowledge of the seemingly most promising point in the search space spreads through all neighborhoods. This process often results in convergence.

Barebones PSO The Barebones PSO algorithm was developed by Kennedy [5], and it is a variant of the standard PSO algorithm designed to mitigate premature convergence. The main idea is to minimize the degree to which good performance depends upon well-tuned settings of parameters like ω, φ_1 and φ_2. While other PSO variations have similar motivation, Barebones PSO completely eliminates these parameters from the algorithm. The main difference from the standard PSO algorithm is a simplified position update (Equation 4) where $\sigma = |pbest_{ij}(t) - gbest_j(t)|$.

$$x_{ij}(t+1) \sim N\left(\frac{pbest_{ij}(t) + gbest_j(t)}{2}, \sigma\right) \ , \quad (4)$$

As shown above, the position of each particle is iteratively sampled from the Gaussian distribution. Barebones PSO favors exploration in the earlier stages of the simulation because the personal best positions will at first be far from the global best one, leading to sampling from a probability distribution with a higher variance. As the simulation proceeds and personal bests converge to the global best, variance will approach zero, thereby focusing on exploitation.

Barebones PSO is a popular variation of the standard PSO algorithm because it has few parameters, and because of its aim to deal with the problem of premature convergence. Over many studies, Barebones PSO has provided better results than the standard PSO algorithm [5, 16, 17]. These factors motivate its use as an additional comparison algorithm in the this paper's experiments.

2.2 Grammatical Swarm

The Grammatical Swarm (GS) algorithm [11] combines PSO with the Grammatical Evolution (GE) mapping process. GE is an Evolutionary Algorithm (EA) able to evolve computer programs in any language that can be described in grammatical form [18]. The insight is that programs can be represented based on a Backus Naur Form (BNF) grammar instead of on parse trees as in tree-based GP.

In GS, most commonly particles are given fixed-length dimensionality, which is the approach adopted in these experiments. To map from PSO's floating point representation to the integer codon representation of GE, each real-valued element of the particle's position is rounded to the nearest integer value. This new array of integers is then mapped through a fixed grammar into a program.

Mapping process In the GS mapping process a particle's location in the search space is processed to construct a program (which can then be evaluated in the domain).

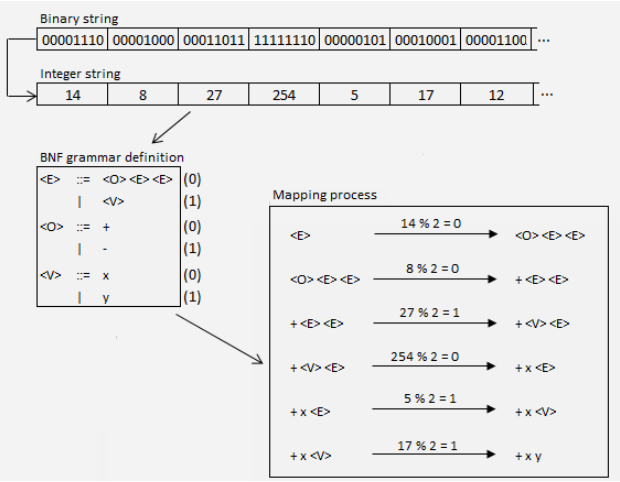

Fig. 1: GE and GS mapping process. This example is based on [11].

An example of the mapping process in GE and GS is shown in Fig. 1. Note that an integer representation is used in this example for simplicity of understating. In this example, the first codon (00001110) is converted to its decimal value (14). From the underlying grammar, the start symbol <E> has two possible rules that can be applied. The modulus of the value of the codon with the number of alternatives determines which rule is selected ($Alternative = CodonValue\%Num.of Alternatives$)

For this example, 14 % 2 = 0, which means that the first rule, <O> <E> <E>, is selected, i.e. the start symbol is replaced with those three non-terminal

symbols. Next, the leftmost symbol <O> and the second codon (8) are processed in the same way. The process then iterates until no non-terminal symbols remain. When the genotype has fewer codons than non-terminals, the genome is "wrapped", i.e. the algorithm continues reading again from the beginning of the genotype. The number of times the genotype wraps is limited in practice to avoid infinite cycles. In this case, the respective particle is marked as invalid. Our preliminary experiments revealed that if particles are forced to be valid (i.e. by updating a particle's velocity until its position becomes valid), the performance of all algorithms improved considerably. Therefore, all results in this paper include this procedure.

2.3 Novelty Search

The novelty search algorithm was introduced by Lehman and Stanley in 2008 as an alternative to objective-based optimization in evolutionary computation [6]. The key idea behind novelty search is to ignore the objective of the search, and instead reward novel individuals, i.e. those with behaviors different from those previously encountered in the search. The insight is that by not optimizing a measure of progress towards the objective, novelty search is not susceptible to premature convergence to local optima. Of course, the significant trade-off is that the search as a whole becomes less explicitly controlled.

While novelty search may record the value of the underlying objective-based fitness function to track whether any solutions have been discovered, this fitness function does not guide the search. Instead novelty search applies a novelty metric, which measures the novelty of an individual in comparison to other individuals in the search, according to its behavior. This enables rewarding individuals with novel behaviors, which incentivizes exploring the space of behaviors. This exploration can often lead to discovering the desired objective of search as a side effect.

Novelty search requires that each individual be assigned a behavior descriptor, which is a vector summarizing the individual's behavior. Thus applying novelty search to a new domain requires that the experimenter devise a quantitative characterization of behavior. For example, in a maze navigation task an individual's behavior descriptor can be its Cartesian coordinates at the end of an evaluation. Because they bias the search, different behavior descriptors applied to the same domain may result in varied performance. In this way, effective novelty search may depend upon a characterization of behavior that succinctly captures relevant aspects of behavior in a domain.

When the behavior space is large, novelty search often benefits when an archive of past behaviors is maintained. The idea is to prevent repeatedly cycling through a series of behaviors, reflecting only a fleeting sense of novelty. By archiving past behaviors, novelty can be measured relative to where search has been and where it currently is, thus incentivizing behaviors that are genuinely novel. Two common archiving strategies are to add behaviors with novelty that is higher than a threshold value, or to select individuals randomly to archive. In

the experiments presented here, all behaviors had a five percent probability of being archived.

To calculate the novelty of an individual requires defining the distance between its behavior descriptor and that of others in the population and in the archive. Given the behavior descriptor and the distance metric between such descriptors, the novelty score is calculated as shown in Equation 5, where μ_i is the i^{th} nearest neighbor of x, and $dist$ is a Euclidean distance metric.

$$\rho(x) = \frac{1}{k} \sum_{i=0}^{k} dist(x, \mu_i) \ . \tag{5}$$

The novelty of each individual is the average distance of its behavior to its k-nearest neighbors. This way, individuals in a less dense area of the behavior space are given higher novelty scores, thus creating pressure in the search towards further novelty.

3 Novelty-driven PSO

Algorithm 1, below, provides the pseudo-code for the novelty-driven PSO algorithm. Introducing novelty search into PSO does not change the core of the algorithm (only in the reward scheme driving search), resulting in pseudocode that is not significantly different from the standard PSO algorithm.

Algorithm 1 Novelty-driven PSO algorithm

```
1: procedure NSPSOProcedure
2: createPopulation: foreachParticle
3:     setRandomPosition
4:     setRandomVelocity
5:     evaluatePopulation
6:     setPbestCurrent
7:     executeNovelty
8:     setPnovelCurrent
9:     end foreach
10:    updateGbest
11:    updateGnovel
12: loop:
13:    if NotDone then foreachParticle
14:        calculateVelocity
15:        calculateNewPosition
16:        evaluatePopulation
17:        updatePbest
18:        executeNovelty
19:        updatePnovel
20:        end foreach
21:        updateGbest
22:        updateGnovel
23:    close;
```

The main change to PSO is to replace the objective-based fitness function with a novelty metric. The standard velocity update equation (Equation 1) shows that particles are attracted by the best position they have encountered and the

overall best position. In novelty-driven PSO, the equation is the same, but the concept of "best position" changes. That is, in novelty-driven PSO, the *pbest* position is not the position where the particle obtained best fitness but the position where it has encountered highest novelty. Similarly, the overall best position (*gbest*) is the position where the maximum value of novelty has been obtained. For this reason, these quantities are referred to as *pnovel* and *gnovel*.

One important aspect of novelty is that it is both relative and dynamic. That is, each particle calculates its novelty with respect to other particles' current behaviors and past archived behaviors. In this way, a highly novel behavior becomes less novel over time as other particles become drawn to it, and it is added to the archive.For this reason, the tracking of the *pnovel* and *gnovel* take this dynamism into account. In particular, the novelty of each particle's *pnovel* behavior are recalculated each iteration. If the novelty score of the current particle position is higher than the recalculated novelty score of its *pnovel* than the latter is updated. In this case, the current position will overwrite *pnovel*, and the particle's behavior will also be cached as the *pnovel* behavior. *Gnovel* is the *pnovel* over all particles with the highest score. Importantly, because the behavior vector for each *pnovel* is cached, no additional domain evaluations are required for such recalculations, and they thus incur little computational overhead.

Note that when more than one *pnovel* shares the same highest novelty score, and the previous *gnovel* is among such high-scoring *pnovel* search points, empirically it was found to be important to conserve the current *gnovel*. That is, if the current *gnovel* particle is one the top ranked *pnovel* particles, it always is chosen to remain as the *gnovel*. But if the current *gnovel* is not among of the tied top *pnovel* scores, one of the top ranked *pnovel* is randomly chosen. That is, *gnovel* changes only when there a *pnovel* with a strictly higher novelty score.

4 Experiments

The aim of these experiments is to compare the performance of the traditional objective-driven GS method with the performance of the NdPSO algorithm proposed in this paper. Recall that novelty search pursues novel behaviors, which makes it best-suited for deceptive problems in which it is possible and intuitive to characterize an individual's behavior. Thus the choice of test domains reflects the type of problem for which the approach is likely to be appropriate. Objective-driven GS and NdGS are tested in three domains: Mastermind, the Santa Fe trail, and a Maze navigation problem.

All experiments have been performed in Netlogo, using a Java implementation of the GE algorithm [19] (the jGE library) and the respective Netlogo extension [20].

4.1 Mastermind

Inspired by the classic board game, in the Mastermind domain considered here the task is to discover the correct sequence before the maximum number of

attempts is exhausted. As in O'Neill [11], four possible colors are considered, and the correct code for the eight given positions was fixed to 3 2 1 3 1 3 2 0. NdGS and GS both use the same grammar as in the original GS experiment (Figure 2).

```
<pin> ::= <pin> <pin> | 0 | 1 | 2 | 3
```

Fig. 2: BNF-Koza grammar definition for the Mastermind problem.

In this domain, objective-based fitness is scored as follows: One point is awarded for each correctly colored pin (regardless of its position, although limited in extent by the total number of pins in the target sequence with that color). If all pins are in the correct position, an additional point is awarded. If the genotype has more than eight codons, it is truncated to the first eight. The fitness score is normalized by dividing the raw score by the maximum score possible ($fitness = score/maxScore$). Note that $maxScore$ in this experiment is 9. By design, this problem is highly deceptive, with local optima corresponding to all sets of correct colors in wrong positions.

In contrast to objective-based PSO, novelty-driven PSO requires a characterization of behavior. In this problem two distinct characterizations were considered: behaviorMm1 - the fitness value, and a behavior inspired by the original game: behaviorMm2 - a tuple of two integers consisting of the number of correct colors and the number of correct positions. Two things are important to note. First, searching for novel values of objective fitness is different than simple objective-based search. That is, novelty search will be driven to accumulate all possible fitness values, not only the largest ones. Thus the performance of objective-driven GS and novelty-driven GS may diverge even though they are driven by the same underlying information. Second, behaviorMm2 provides an ideal decomposition of the domain, one which is unavailable to the objective fitness function (i.e. both colors and placements are important). Thus this characterization highlights the potential for injecting experimenter knowledge into the search process, which may otherwise be complicated in objective-driven search because of the need to reduce performance information to a single number.

4.2 Santa Fe Trail

The Santa Fe trail is a difficult and popular benchmark in both GP [21] and GE [18].

The goal is to evolve a computer program that can efficiently guide an artificial ant to eat all pieces of food placed in the trail (Figure 3). Beginning from the upper left corner, the artificial ant can move forward in the direction it is currently facing or turn 90 degrees to the right or left. Each action takes one discrete unit of time to perform. The ant can perceive if the cell in front

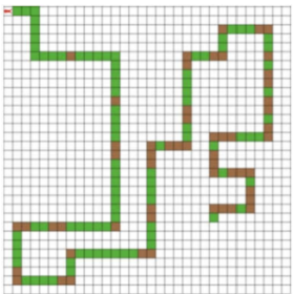

Fig. 3: The Santa Fe Trail.

```
<code> ::= <line> | <code> <line>
<line> ::= <condition> | <op>
<condition> ::= ifelse food-ahead [ <line> ][ <line> ]
<op> ::= turn-left | turn-right | move
```

Fig. 4: BNF-O'Neill grammar definition for the Santa Fe trail.

of it contains food, an operator that executes instantaneously (i.e. it does not consume any time). Fig.4 shows the grammar used in this domain.

For objective-driven GS, the traditional fitness function simply counts the units of food eaten by the ant after all time has been exhausted. The standard maximum number of steps is used in this experiment, i.e. 615.

For novelty-driven GS, two behavior descriptors are applied. A simpler descriptor - behaviorSF1 - adopts the fitness function as the characterization of behavior, as in the Mastermind domain. A more informative characterization - behaviorSF2 - considers the amount of food eaten, with the constraint that the eaten units must not be disconnected from other eaten units along the length of the trail.

For example, if the ant first eats 3 food units that follow the trail, then leaves the trail and eats one more unit in another area of the trail, its behavior descriptor is appended with a 3 (although the ant ate in total 4 units of food), because the last unit is not connected to any other eaten units along the true path of the trail. However, if the ant eats 3 food units at the beginning of the trail, goes off the trail, and collects three more units along a later part of the trail, the score will then be 6. Additionally, this second characterization is sampled over time to provide temporal information about the ant's behavior. In particular, it is sampled every 41 timesteps, resulting in a vector of length 15 by the completion of an ant's evaluation.

4.3 Maze navigation problem

The Medium Map domain (Figure 5) is a deceptive and discrete maze navigation task introduced by Lehman and Stanley [22]. The goal in this domain is to find a program that guides an agent in a grid-world domain to the goal location before exhausting the time limit. In our experiments the time limit was set to 500 steps. This maze is suitable for testing GS and NdGS because the placement of the walls create deception. That is, the shortest path to the goal is blocked, meaning that to solve the task requires exploring areas that superficially appear further from the goal.

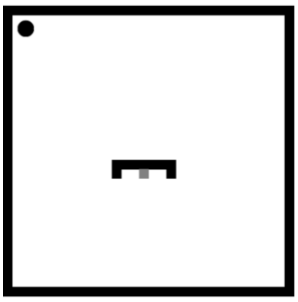

Fig. 5: Medium Map used for the maze navigation problem.

```
<expr> ::= <line> | <expr> <line>
<line> ::= ifelse <condition> [ <expr> ][ <expr> ] | <op>
<condition> ::= wall-ahead? | wall-left? | wall-right?
<op> ::= turn-left | turn-right | move
```

Fig. 6: BNF-Koza grammar definition for the Medium Maze problem.

The possible actions for each agent are: *turn-left*, *turn-right*, *move* and the boolean operators *wall-ahead*, *wall-left* and *wall-right*. Similarly to as in the Sante Fe Trail, the turning actions respectively turn the agent 90 degrees left or right, while the move operation causes the agent to progress forward one unit in the direction it currently faces. All but the last three actions consume a timestep.

Defining *dist* as the euclidean distance of the agent from the goal location, the fitness function for objective-based GS is calculated as $fitness = \frac{1}{1+dist}$.

For novelty-driven GS, the behavior descriptor adopted was the coordinates of the agent's ending position - behaviorMP1. In this way, objective-based search is looking for ways to get closer to the goal, while novelty search instead explores how to reach a diversity of places within the maze.

4.4 Experimental parameters

As in O'Neill's original GS experiments [11], the following parameter settings were adopted in all experiments: $\varphi_1 = \varphi_2 = 1.0$, $\omega_{max} = 0.9$, $\omega_{min} = 0.4$ and $v_{ij} \in [-255, 255]$. A maximum of 10 wraps was allotted for an individual to be considered valid, and a coordinate's value was bounded between 0 and 255, inclusive. When a particle exceeds the maximum or minimum value for a particular dimension, the value is clipped such that it lies on the extreme of that range. The swarm consisted of 30 particles, with a search space of dimensionality 100. Simulations ran for 1000 iterations.

Three different neighborhood topologies are tested: Fully-connected, Ring and Von Neumann. All results reported for objective-driven GS and GS (Barebones) use the Fully-connected topology which provided the best results. Interestingly, with objective-driven GS the Ring and Von Neumann topologies have effect opposite from when used in NdGS (where their use improves results significantly).

In the Santa Fe trail and in the maze navigation problem 3 nearest neighbors were used for calculating novelty in NdGS, and 15 nearest neighbors were used in the Mastermind problem. Because the behavior *NdGS-behaviorSF2* in Santa

Fe is composed of many samples taken over an ant's evaluation, the best result was obtained by including the archive.

5 Results and discussion

The results presented in Table 1 are the best obtained for each algorithm in the considered domain, over the combinations of topologies and other parameters described in the section above. In all problems NdPSO outperforms the objective-based algorithms tested, both in number of solutions found and in the best fitnesses discovered averaged over all runs. In particular, the average best fitnesses are significantly higher for the NdPSO methods than for any of the control algorithms (Student's t-test; $p < 0.05$). Interestingly, in the Mastermind problem, which is considered a difficult GP problem, NdPSO managed to succeed in all runs while objective-driven GS achieved success in only 14% of runs.

Table 1: Comparison of the results obtained for Grammatical Swarm, Novelty-driven Grammatical Swarm and Random Search averaged over 100 runs

	Mean best fitness	Std Deviation	Median	Successful runs
Mastermind				
GS	0.90	0.04	0.89	14
GS (Barebones)	0.87	0.04	0.89	2
NdGS-behaviorMm1	0.92	0.05	0.89	25
NdGS-behaviorMm2	**1.00**	**0.00**	**1**	**100**
Random	0.86	0.05	0.89	0
Santa Fe Trail				
GS	78.31	15.65	89	52
GS (Barebones)	74.25	16.95	88	49
NdGS-behaviorSF1	85.46	8.54	89	75
NdGS-behaviorSF2	**87.89**	**6.52**	**89**	**78**
Random	75.81	15.74	88	40
Maze problem				
GS	0.43	0.36	0.24	27
GS (Barebones)	0.50	0.36	0.31	34
NdGS-behaviorMP1	**0.68**	**0.38**	**1**	**58**
Random	0.30	0.30	0.17	15

While every NdPSO treatment outperforms the control algorithms, it is important to note that the selection of the behavior descriptor is a significant factor. In particular, a domain-specific descriptor that integrates additional information than the raw objective-based fitness measure led to further performance gains. Additionally, performance of NdPSO is affected by interactions between the behavior characterization and whether or not an novelty archive is included. In

particular, when using behaviorSF2 in the Santa Fe Trail and the Maze navigation problem, an archive provided better results.

Despite its promise in previous work, applying objective-based Barebones GS did not improve upon on the results of standard GS, performing worse than GS in 2 out of 3 domains. This failure may be particular to the chosen domains, or mat result from combining the Barebones approach with GS.

A final interesting result is that the best neighborhood topology differs across objective-driven and novelty-driven GS. As described earlier, the best performance for objective-driven GS (with and without Barebones PSO) resulted from the fully-connected topology, whereas in NdGS the Ring topology substantially improved performance. This result highlights a qualitative difference between an objective-driven search and a novelty-driven one; best practices for one paradigm may not directly transfer to the other.

6 Conclusion and Future Work

This paper introduced Novelty-driven PSO (NdPSO), a method that aims to mitigate the challenge of premature convergence in traditional objective-driven PSO. NdPSO was combined with Grammatical Swarming and tested in three domains. In each domain it outperformed two objective-based control algorithms and random search. In this way, NdPSO shows promise for solving deceptive PSO problems and encourages further follow-up investigation. In particular, future research will compare NdPSO to novelty-driven GE, to examine how its performance compares to the evolutionary novelty search that inspired it.

Acknowledgments. This work was funded by FCT project EXPL/EEI-SI/1861/2013.

References

1. Kennedy, J and Eberhart, R: Particle swarm optimization. In: 1Proceedings of IEEE International Conference on Neural Networks, volume 4. IEEE Int (1995)
2. Peer, E. S., F. Van den Bergh, and A. P. Engelbrecht. Using neighbourhoods with the guaranteed convergence PSO, In Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE, pp. 235-242. IEEE (2003)
3. Ratnaweera, A., S. K. Halgamuge, and H. C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evol. Comput., vol. 8, no. 3, pp. 240255,(2004)
4. Vesterstrom, Jakob, and Rene Thomsen. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. Evolutionary Computation, 2004. CEC2004. Congress on. Vol. 2. IEEE, (2004)
5. Kennedy, J., Bare bones particle swarms. In Swarm Intelligence Symposium, 2003. SIS03. Proceedings of the 2003 IEEE, 8087, IEEE.

6. Lehman, Joel, and Kenneth O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In Proceedings of the Eleventh International Conference on Artificial Life (ALIFE XI), Cambridge, MA, (2008). MIT Press
7. Lehman, Joel, Kenneth O. Stanley, and Risto Miikkulainen. Effective diversity maintenance in deceptive domains. In Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference, pp. 215-222. ACM, (2013)
8. Lehman, Joel, and Kenneth O. Stanley. Efficiently evolving programs through the search for novelty. In Proceedings of the 12th annual conference on Genetic and evolutionary computation, pp. 837-844. ACM, (2010)
9. Lehman, Joel, and Kenneth O. Stanley. Novelty search and the problem with objectives. In Genetic Programming Theory and Practice IX, pp. 37-56. Springer New York, (2011)
10. Urbano, Paulo, and Loukas Georgiou. "Improving grammatical evolution in santa fe trail using novelty search." Advances in Artificial Life, ECAL. Vol. 12. 2013.
11. ONeill, Michael, and Anthony Brabazon. "Grammatical swarm: The generation of programs by social programming." Natural Computing 5.4 (2006): 443-462.
12. Kennedy, J. and Eberhart, R.,Particle swarm optimization. In Proceedings of IEEE International Conference on Neural Networks, 4, (1995)
13. Ozcan, E. and Mohan, C.K. (1998). Analysis of a simple particle swarm optimization system. Intelligent engineering systems through artificial neural networks, 8, 253258
14. Medina, A. J. R., Pulido, G. T., and Ramrez-Torres, J. G. (2009). A Comparative Study of Neighborhood Topologies for Particle Swarm Optimizers. In IJCCI (pp. 152-159)
15. Ali, M. and Kaelo, P. (2008). Improved particle swarm algorithms for global optimization. Applied mathematics and computation, 196, 578593
16. Omran, M., and S. Al-Sharhan. "Barebones particle swarm methods for unsupervised image classification." Evolutionary Computation, 2007. CEC 2007. IEEE Congress on. IEEE, 2007.
17. Yao, Jingzheng, and Duanfeng Han. "Improved barebones particle swarm optimization with neighborhood search and its application on ship design." Mathematical Problems in Engineering 2013 (2013).
18. Ryan, Conor, J. J. Collins, and Michael O. Neill. "Grammatical evolution: Evolving programs for an arbitrary language." Genetic Programming. Springer Berlin Heidelberg, 1998. 83-96.
19. Georgiou, Loukas, and William J. Teahan. "jGE-A Java implementation of Grammatical Evolution." Proceedings of the 10th WSEAS International Conference on SYSTEMS, Vouliagmeni. Athens: WSEAS. 2006.
20. Georgiou, Loukas, and William J. Teahan. "Grammatical evolution and the santa fe trail problem." Evolutionary Computation (ICEC 2010) (2010).
21. Koza, J. R. (1991). Genetic evolution and co-evolution of computer programs. In Langton, C. T. C., Farmer, J. D., and Ras-mussen, S., editors, Artificial Life II, volume X ofSFI Studies in the Sciences of Complexity, pages 603-629. Addison-Wesley, Santa Fe Institute, New Mexico, USA
22. Lehman, Joel, and Kenneth O. Stanley. "Efficiently evolving programs through the search for novelty." Proceedings of the 12th annual conference on Genetic and evolutionary computation. ACM, 2010.