Surprise Search for Evolutionary Divergence

Daniele Gravina, Antonios Liapis, and Georgios N. Yannakakis

***Abstract*—Inspired by the notion of *surprise* for unconventional discovery we introduce a general search algorithm we name *surprise search* as a new method of evolutionary divergent search. Surprise search is grounded in the divergent search paradigm and is fabricated within the principles of evolutionary search. The algorithm mimics the self-surprise cognitive process and equips evolutionary search with the ability to seek for solutions that deviate from the algorithm's expected behaviour. The predictive model of expected solutions is based on historical trails of where the search has been and local information about the search space. Surprise search is tested extensively in a robot maze navigation task: experiments are held in four authored deceptive mazes and in 60 generated mazes and compared against objective-based evolutionary search and novelty search. The key findings of this study reveal that surprise search is advantageous compared to the other two search processes. In particular, it outperforms objective search and it is as efficient as novelty search in all tasks examined. Most importantly, surprise search is faster, on average, and more robust in solving the navigation problem compared to any other algorithm examined. Finally, our analysis reveals that surprise search explores the behavioural space more extensively and yields higher population diversity compared to novelty search. What distinguishes surprise search from other forms of divergent search, such as the search for novelty, is its ability to diverge not from earlier and seen solutions but rather from predicted and unseen points in the domain considered.**

***Index Terms*—Surprise search, novelty search, divergent search, deception, fitness-based evolution, maze navigation, NEAT.**

I. Introduction

OVER the last 50 years, evolutionary computation (EC) has shown vast potential in numerical and behavioural optimization. The most common approach to optimization in artificial evolution is via an *objective function*, which rewards solutions based on their 'goodness' [1], i.e. how close they are to an optimal behaviour (if such a behaviour is known beforehand) or how much they improve a performance metric. The objective function (or fitness function) encapsulates the principle of evolutionary pressure for fitting (adapting) within the environment. Despite the success of such approaches in a multitude of tasks [1], [2], they are challenged in deceptive fitness landscapes [3] where the global optimum is neighboured by low-quality solutions. In such cases, the local search of an objective-based evolutionary algorithm can guide search away from a global optimum and towards local optima. As a general principle, more deceptive problems challenge the design of a corresponding objective function; this paper follows [4] and views deception as the *intuitive definition of problem hardness*. Many algorithms have been proposed to tackle the problem of deception, primarily revolving around diversity preservation [5]–[7], which deters premature convergence while still rewarding proximity to the objective and divergent search [8] which abandons objectives in favour of rewarding diversity in the population.

All authors are with the Institute of Digital Games, University of Malta, Msida 2080, Malta (e-mail: daniele.gravina@um.edu.mt; antonios.liapis@um.edu.mt; georgios.yannakakis@um.edu.mt)

There are, however, problems which lack an easily defined objective — or a gradient to reaching it. For instance, open-ended evolution studies within artificial life [9] do not have a goal state and instead prioritize e.g. survival [10], [11]. In evolutionary art, music or design, a large body of research in computational creativity and generative systems [12]–[14] focuses on the *creative* capacity of search rather than on the objectives. In [13], computational creativity is considered on two core properties of a produced solution: value and novelty [13]. Value is the degree to which a solution is of high quality, whereas novelty is the degree to which a solution (or output) is dissimilar to existing examples. While objective-based EC can be seen as a metaphor of searching for value, a divergent EC strategy as *novelty search* [8], [15] can be considered as a metaphor of searching for novelty. An effective use of both in EC can lead to highly novel and valuable at the same time outcomes [16], thus realizing *quality diversity* [17]. However, according to [18], novelty and value are not sufficient for the discovery of highly creative and unconventional solutions to problems. While novelty can be considered as a static property, surprise considers the temporal properties of the discovery, an important dimension to assess the creativity of the generated solution [18], [19]. Further studies in general intelligence and decision making [20] support the importance of unexpectedness for problem-solving.

Driven by the notion of computational surprise for the purpose of creatively traversing the search space towards unexpected or serendipitous solutions, this paper proposes *surprise search* for the purposes of divergent evolutionary search. The hypothesis is that searching for unexpected — not merely unseen — solutions is beneficial to EC as it complements our search capacities with highly efficient and robust algorithms beyond the search for objectives or mere novelty. Surprise search is built upon the novelty search [8] paradigm that rewards individuals which differ from other solutions in the same population and a historical archive. Surprise is assumed to arise from a violation of expectations [21]: as such, it is different than novelty which rewards deviation from past and current behaviours. A computational, quantifiable model of surprise must build expectations based on trends in past behaviours, and predict future behaviours from which it must diverge from. In order to create expected behaviours, the algorithm maintains a lineage of where evolutionary search has been. These groups of evolutionary lineages require the right level of locality in the behavioural space — surprise can be inclusive of all behaviours (globally) or merely consider part of all possible behaviours (locally). Any deviation from these stepping stones of search would elicit surprise; alternatively, they can be viewed as serendipitous discovery if the deviation leads to a surprisingly good point in the behavioural space. The findings of this paper suggest that surprise constitutes a powerful drive for computational discovery as it incorporates predictions of an expected behaviour that it attempts to deviate from; these predictions may be based on behavioural relationships in the solution space as well as historical trends derived from the algorithm's sampling of the domain.

This paper builds upon and extends significantly our earlier work which introduced the notion of surprise search [22] and compared its performance against novelty search and objective-based search [23] in the maze navigation testbed of [8]. The current paper extends the preliminary study of [23] by introducing two new maze navigation problems of increased complexity, and analysing the impact of several parameters to the behaviour of surprise search. This paper also includes an extensive comparison between novelty search and surprise search both in the behavioural and the genotypic space. Finally, to further examine how surprise search performs across a wide range of problems, we test how it scales in sixty randomly generated mazes of varying degrees of complexity. The key findings of the paper suggest that surprise search is as efficient as novelty search and both algorithms, unsurprisingly, outperform fitness-based search. Furthermore, surprise search appears to be the most robust algorithm in the four testbed tasks and to be the most successful and fastest algorithm in the 60 randomly generated mazes. While both novelty and surprise search converge to the solution significantly faster than fitness-based search, surprise search solves the navigation problem faster, on average, and more often than novelty search. The experiments of this paper validate our hypothesis that surprise can be beneficial as a divergent search approach and provide evidence for its supremacy over novelty search in the tasks examined.

II. Deception, Divergent Search and Quality Diversity

This section motivates surprise search by providing a brief overview of the challenges faced by fitness-based approaches when handling *deceptive* problems and how *divergent search* has been used to address such challenges. The section concludes with a discussion on the relationship between surprise search and *quality diversity* algorithms.

A. Deception in Evolutionary Computation

The term *deception* in the context of EC was introduced by [3] to describe instances where highly-fit building blocks, when recombined, may guide search away from the global optimum. Since that first mention, the notion of deception (including deceptive problems and deceptive search spaces) has been refined and expanded to describe several problems that challenge evolutionary search for a solution. [4] argues that "the only challenging problems are deceptive". EC-hardness is often attributed to deception, as well as sampling error [24] and a rugged fitness landscape [25]. In combinatorial optimisation problems, the fitness landscape can affect optimisation when performing local search. Such a search process assumes that there is a high correlation between the fitness of neighbouring points in the search space, and that genes in the chromosome are independent of each other. The latter assumption refers to *epistasis* [26] which is a factor of GA-hardness: when epistasis is high (i.e. where too many genes are dependent on other genes in the chromosome), the algorithm searches for a unique optimal combination of genes but no substantial fitness improvements are noticed [26].

As noted, epistasis is evaluated from the perspective of the fitness function and thus is susceptible to deception; [27] argue that deceptive functions can not have low epistasis, although fitness functions with high epistasis are not necessarily deceptive. Such approaches are often based on the concepts of correlation, i.e. the degree to which an individual's fitness score is well correlated to its neighbours' in the search space, or epistasis, i.e. the degree of interaction among genes' effects. As noted by [8], most of the factors of EC-hardness originate from the fitness function itself; however, poorly designed genetic operators and poorly chosen evolutionary parameters can exacerbate the problem.

B. Divergent Search

By definition, the biggest issue of a deceptive problem is premature convergence to local optima, while the global optimum is difficult to reach as the deceptive fitness landscape lead the search away from it. Several approaches have been proposed to counter this behaviour, surveyed by [15]. For instance, speciation [28] and niching [29] are popular diversity maintenance techniques, which enforce local competition among similar solutions. Similarity can be measured on the genotypical level [5], on the fitness scores [6], or on the age of the individuals [7]. Multiple promising directions are favoured by this localised competition, making premature convergence less likely. Alternatives such as coevolution [30] can lead to an arms race that finds a better gradient for search, but can suffer when individuals' performance is poor or one individual vastly outperforms others [31]. Finally, techniques from multi-objective optimisation can, at least in theory, explore the search space more effectively by evaluating individuals in more than one measures of quality [32] and thus avoid local optima in one fitness function by attempting to improve other objectives; however, multi-objective optimisation can not guarantee to circumvent deception [33].

Divergent search methods differ from previous approaches as they explicitly ignore the objective of the problem they are trying to solve. While the approaches described above provide control mechanisms, modifiers or alternate objectives which complement the gradient search towards a better solution, divergent algorithms such as novelty search [8] motivates exploration of the search space by rewarding individuals that are phenotypically (or behaviourally) different without considering whether they are objectively "better" than others. Novelty search is neither random walk nor exhaustive search, however, as it gives higher rewards to solutions that are different from others in the current and past populations by maintaining a memory of the areas of the search space it has explored via a *novelty archive*. The archive contains previously found novel individuals, and the highest-scoring individuals in terms of novelty are continuously added to it as evolution carries on. The distance measure which assesses "difference" is based on a behaviour characterization, which is problem-dependent: for a maze navigation task, distance may be calculated on the agents' final positions or directions [8], [17], for robot locomotion it may be on the position of a robot's centre of mass [8], for evolutionary art it may be on properties of the images such as brightness and symmetry [34].

C. Quality Diversity

A recent trend in evolutionary computation is inspired by a species' tendency to face a strong competition for survival within its own niche [17]. EC algorithms of this type seek for the discovery of both *quality* and *diversity* at the same time, following the traditional approach within computational creativity of seeking outcomes characterized by both quality (or *value*) and novelty [13]. Such evolutionary algorithms have been named *quality diversity* algorithms [17] and aim to find a maximally diverse population of highly performing individuals. Examples of such algorithms include novelty search with local competition [35] and MAP-Elites [36] as well as algorithms that constrain the feasible space of solutions — thereby forcing high quality solutions — while searching for divergence such as *constrained novelty search* [16], [37].

Surprise search can complement the search for diversity and replace other divergent search algorithms commonly used for quality diversity. It can, for instance, be used in combination with local competition instead of novelty search as in [35]. Towards that goal, a recent study employed surprise search for game weapon generation in a constrained fashion [38]. The *constrained surprise search* algorithm rewarded the generation of surprising weapons — thereby maintaining diversity — with guaranteed high quality imposed by constraints on weapon balance and effectiveness.

III. The Notion of Surprise Search

This section examines the notion of surprise as a driver of evolutionary search. To this end, we first describe the concept of surprise, we then highlight the differences between surprise and novelty and, finally, we frame surprise search in the context of divergent search.

A. What is Surprise?

The study of surprise has been central in neuroscience, psychology, cognitive science, and to a lesser degree in computational creativity and computational search. In psychology and emotive modelling studies, surprise defines one of Ekman's six basic emotions [39]. Within cognitive science, surprise has been defined as a temporal-based cognitive process of the unexpected [21], [40], a violation of a belief [41], a reaction to a mismatch [21], or a response to novelty [14]. In computational creativity, surprise has been attributed to the creative output of a computational process [14], [18].

While variant types and taxonomies of surprise have been suggested in the literature — such as aggressive versus passive surprise [18] — we can safely derive a definition of surprise that is general across all disciplines that study surprise as a phenomenon. For the purposes of this paper we define surprise as the *deviation from the expected* and we use the notions *surprise* and *unexpectedness* interchangeably due to their highly interwoven nature.

B. Novelty vs. Surprise

Novelty and surprise are different notions by definition as it is possible for a solution to be both novel and/or expected to variant degrees. Following the core principles of Lehman and Stanley [8] and Grace et al. [18], novelty is defined as the degree to which a solution is *different from prior* solutions to a particular problem. On the other hand, surprise is the degree to which a solution is *different from the expected* solution to a particular problem.

Expectations are naturally based on inference from past experiences; analogously surprise is built on the temporal model of past behaviours. To exemplify the difference between the notions of novelty and surprise, [22] uses a card memory game where cards are revealed, one at a time, to the player who has to predict which card will be revealed next. The novelty of game outcome (i.e. next card) is the highest if all past revealed cards are different. The surprise value of the game outcome in that case is low as the player has grown to expect a new, unseen, card every time. On the other hand, if seen cards are revealed after a while then the novelty of the game outcome decreases, but surprise increases as the game deviates from the expected behaviour which calls for a new card every time.

Surprise is a temporal notion as expectations are by nature temporal. Prior information is required to predict what is expected; hence a *prediction of the expected* [19] is a necessary component for modelling surprise computationally. By that logic, surprise can be viewed as a *temporal novelty* process or as novelty on the prediction (rather than the behavioural) space. Surprise search maintains a prediction (a gradient) of where novelty has been on the prediction space, which is derived from the behavioural space. In that sense surprise resembles a *time derivative* of novelty.

C. Novelty Search vs. Surprise Search

According to Grace et al. [18], novelty and value (i.e. objective in the context of EC) are not sufficient for the discovery of unconventional solutions to problems (or creative outputs) as novelty does not cater for the temporal aspects of discovery. Novelty search rewards divergence from *current* behaviours (i.e. other individuals of the population) and *prior* behaviours (i.e. an archive of previously novel solutions) [8]; in this way it provides the necessary stepping stones toward achieving an objective (i.e. value). Surprise, on the other hand, complements the search for novelty as it rewards divergence from the *expected* behaviour. In other words while novelty search attempts to deviate from previously seen solutions, surprise search attempts to deviate from solutions that are expected to be seen in the future.

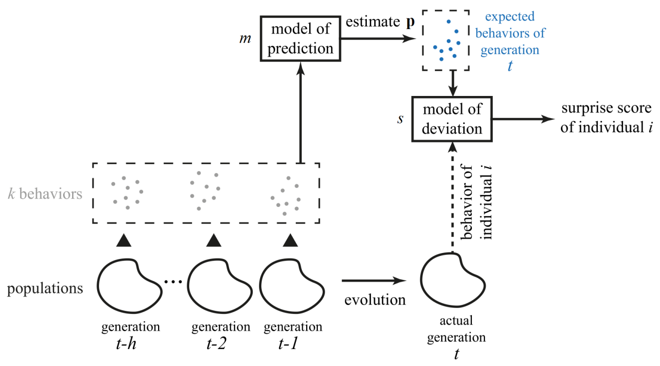


Fig. 1: High-level overview of the surprise search algorithm when evaluating an individual i in a population at generation t. The h previous generations are considered, with respect to k behavioural characteristics per generation, to predict the expected k behaviours of generation t. The surprise score of individual i is the deviation of the behaviour of i from a subset of these k expected behaviours.

Highly relevant to this study is the work on computational models of surprise for artificial agents [42], which however does not consider using such a model of surprise for search. Other aspects of unexpectedness such as intrinsic motivation [43] and artificial curiosity [44] have also been modelled. The concepts of novelty within reinforcement learning research are also interlinked to the idea of surprise search [43], [45]. Artificial curiosity and intrinsic motivation differ from surprise search as the latter is based on evolutionary divergent search and motivated by open-ended evolution, similarly to novelty search. Specifically, surprise search does not keep a persistent world model as [44] does; instead it focuses on the current trajectory of search using the latest points of the search space it has explored (and ordering them temporally). Additionally, it rewards deviations from expected behaviours agnostically rather than based on how those deviations improve a world model. This allows surprise search to backtrack and re-visit areas of the search space it has already visited, which is discouraged in both novelty search and curiosity.

Inspired by the above arguments and findings in computational creativity, we view surprise for computational search as the degree to which expectations about a solution are violated through observation [18]. Our hypothesis is that if modelled appropriately, surprise may enhance divergent search and complement or even surpass the performance of traditional forms of divergent search such as novelty. The main findings of this paper validate our hypothesis.

IV. The Surprise Search Algorithm

This section discusses the principles of designing a surprise search algorithm for any task or search space. To realise surprise as a search mechanism, an individual should be rewarded when it *deviates from the expected behaviour*, i.e. the evaluation of a population in evolutionary search is adapted. This means that surprise search can be applied to any EC method, such as NEAT [28] in the case study of this paper.

As discussed above, surprise search must reward an individual's deviation from the expected behaviour. This goal can be decomposed into two tasks: *prediction* and *deviation*. At the highest descriptive level, surprise search uses local information from past generations to predict behaviour(s) of the population in the current generation; observing the behaviours of each individual in the actual population, it rewards individuals that deviate from predicted behaviour(s): this is summarized in Fig. 1. The following sections will describe the models of prediction and deviation, and the parameters which can affect their behaviour.

A. Model of Prediction

As shown in Fig. 1, the predictive model uses local information from previous generations to estimate (in a quantitative way) the expected behaviour(s) in the current population. Formally, predicted behaviours ($\mathbf{p}$) are created via eq. (1), where m is the predictive model that uses a degree of local (or global) behavioural information (expressed by k) from h previous generations. Each parameter (m, h, k) may influence the scope and impact of predictions, and are problem-specific both from a theoretical (as they can affect performance of surprise search) and a practical (as certain domains may limit the possible choice of parameters) perspective.

$$\mathbf{p} = m(h, k) \tag{1}$$

How much history of prior behaviours (h) should surprise search consider?: In order to predict behaviours in the current population, the predictive model must consider previous generations. In order to estimate behaviours of a population at generation t, the predictive model must find trends in the populations of generations $t-1, t-2, \cdots, t-h$. The minimum number of generations to consider in order to observe an evolutionary trend, therefore, is $h = 2$ (the two prior generations to the one being evaluated). However, behaviours that have performed well in the past could also be included in a *surprise archive*, similar to the novelty archive of novelty search [8], and subsequently used to make predictions of interesting future behaviours. Such a surprise archive would serve as a more persistent history ($h > 2$) but considering only the interesting historical behaviours rather than all past behaviours.

How local (k) are the behaviours surprise search needs to consider to make a prediction?: Surprise search can consider behavioural trends of the entire population when creating a prediction (global information). In that case, $k = 1$ and all behaviours are aggregated into a meaningful average metric for each prior generation. The current generation's expected behaviours are similarly expressed as a single (average) metric; deviation of individuals in the actual population is derived from that single metric. At the other extreme, surprise search can consider each individual in the population and derive an estimated behaviour based on the behaviours of its ancestors in the genotypic sense (parents, grandparents etc.) or behavioural sense (past individuals with the closest behaviour). In this case $k = P$ where P is the size of the population, and the number of predictions to deviate from will similarly be P. Therefore, the parameter k determines the level of *prediction locality* which can vary from 1 to P; intermediate values of k split prior populations into a number of population groups using problem-specific criteria and clustering methods.

What predictive model (m) should surprise search use?: Any predictive modelling approach can be used to predict a future behaviour, such as a simple linear regression of a number of points in the behavioural space, non-linear extrapolations, or machine learned models. Again, we consider the predictive model, m, to be problem-dependent and contingent on the h and k parameters.

B. Model of Deviation

To put pressure on unexpected behaviours, we need an estimate of the deviation of an observed behaviour from the expected behaviour (if $k = 1$) or behaviours. Following the principles of novelty search [8], this estimate is derived from the *behaviour space* as the average distance to the n-nearest expected behaviours (prediction points). The *surprise score* s for an individual i in the population is calculated as:

$$s(i) = \frac{1}{n}\sum_{j=0}^{n} d_s(i, p_{i,j}) \qquad (2)$$

where d_s is the domain-dependent measure of behavioural difference between an individual and its expected behaviour, $p_{i,j}$ is the j-closest prediction point (expected behaviour) to individual i and n is the number of prediction points considered; n is a problem-dependent parameter determined empirically ($n \leq k$).

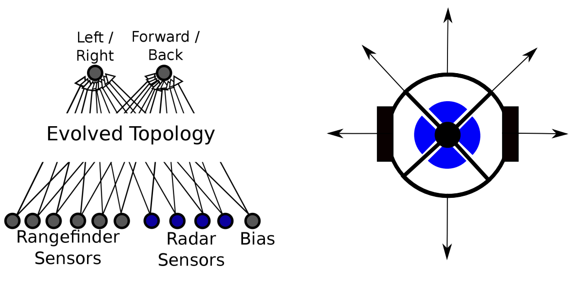
C. Important notes

The evolutionary dynamics of surprise search are similar to those achieved via novelty search. A temporal window of where the search has been is maintained by looking at the prior behaviours (expressed by h and k), which are used to make a prediction of the expected behaviours. However, surprise search works on a different search space (the prediction space) which is orthogonal to the behavioural space used by novelty, as it deviates from the expected and not from the actual behaviours. Therefore, a new form of divergent search emerges, which looks at previous behaviours only in an implicit way in order to derive the prediction space. A concern could be if surprise search is merely a version of random walk, especially considering that it deviates from predictions which can differ from actual behaviours. Several comparative experiments in section VII show that surprise search is different and far more effective compared to various random benchmark algorithms. The source code of the surprise search algorithm is publicly available here[1].

V. Maze Navigation Test Bed

Inspired by the work of Lehman and Stanley for testing novelty search [8], we use a maze navigation problem as a testbed for surprise search, as it particularly suits the definition of deceptive problem. The maze navigation task is made of a closed two-dimensional maze, which contains a start position and a goal position: the navigation task has a deceptive landscape—which is directly visible to a human observer—due to the several local optima present in the search space of the problem. *Cul-de-sacs* added in the shortest path to the goal makes the problem more deceptive, as EC must visiting positions with lower fitness scores before reaching the goal, making the problem harder and more deceptive. In this case, navigation is performed by virtual robot controllers with sensors and mechanisms for controlling their direction and speed: the mapping between the two is provided is via an artificial neural network (ANN) evolved via neuroevolution of augmenting topologies [28]. Starting from the mazes introduced in [8], we designed two additional mazes of enhanced complexity and deceptiveness. This section briefly describes the maze navigation problem, the four mazes adopted, and the parameters for the experiment of Section VII.

[1]http://www.autogamedesign.eu/mazesurprisesearch



(a) Neural Network (b) Sensors

Fig. 2: Robot controller for the maze navigation task. Fig. 2a shows the network's inputs and outputs. Fig. 2b shows the layout of the sensors: the six black arrows are rangefinder sensors, and the four blue pie-slice sensors act as a compass towards the goal.

A. The Maze Navigation Task

The maze navigation task consists of finding the path from a starting point to a goal in a two-dimensional maze, in a fixed number of simulation steps. The problem becomes harder when mazes include dead-ends and the goal is far away from the starting point. As in [8], the robot has six range sensors to measure its distance from the closest obstacle, plus four range radars that fire if the goal is in their arc (see Fig. 2b). Therefore, the robot's ANN receives 10 inputs from the sensors and it controls two actuators, i.e. whether to turn or change the speed (see Fig. 2a). Evolving a controller able to successfully navigate a maze is a challenging problem, as EC needs to evolve a complex mapping between the input (sensors) and the output (movement) in an unknown environment. Even if it can be considered a toy problem, it is an interesting testbed as it stands for a general deceptive search space. Two properties have made this environment a canonical test for divergent search (e.g. [8], [17]): the ease of manually designing deceptive mazes and the low computational burden, which enables researchers to run multiple comparative tests among

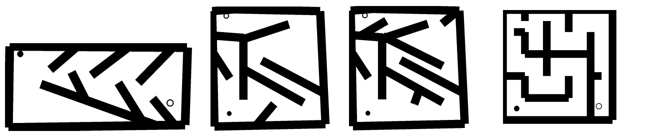

(a) Medium (b) Hard (c) Very hard (d) Extremely hard

Fig. 3: The maze testbeds that appear in [8] (Fig. 3a and 3b) and new mazes introduced in [46] and this paper (Fig. 3c and 3d respectively). The filled circle is the robot's starting position and the empty circle is the goal. The maze size is 300×150 units for medium and 200×200 for the other mazes.

algorithms. Furthermore, the generality of the findings can be tested with automatically generated mazes, as in Section VIII.

B. Mazes

This paper tests the performance of surprise search on four mazes (see Fig. 3), two of which (*medium* and *hard*) have been used in [8]. The medium maze (see Fig. 3a) is somewhat challenging as an algorithm should evolve a robot that avoids dead-ends placed alongside the path to the goal. The hard maze (see Fig. 3b) is more deceptive, due to the dead-end at the leftmost part of the maze; an algorithm must search in less promising (less fit) areas of the maze to find the global optimum. For these two mazes we follow the experimental parameters set in [8] and consider a robot successful if it manages to reach the goal within a radius of five units at the end of an evaluation of 400 simulation steps.

Beyond the two mazes of [8], two additional mazes (*very hard* and *extremely hard*) were designed to test an algorithm's performance in even more deceptive environments. The very hard maze (see Fig. 3c) is a modification of the hard maze introduced in [46] with more dead ends and winding passages. The extremely hard maze is a new maze (see Fig. 3d) that features a longer and more complex path from start to goal, thereby increasing the deceptive nature of the problem.

If we define a maze's complexity as the shortest path between the start and the goal, complexity increases substantially from medium (240 units), to hard (360 units), to very hard (442 units) and finally to the extremely hard maze (552 units); note that the last three mazes are of equal size. The high problem complexity of the very hard and the extremely hard mazes led us to empirically increase the number of simulation steps for the evaluation of a robot to 500 and 1000 simulation steps, respectively. By increasing the simulation time in the more deceptive mazes we manage to achieve reasonable performances for at least one algorithm examined which allows for a better analysis and comparison.

VI. Algorithm Parameters for Maze Navigation

This section provides details about the general and specific parameters for all the algorithms compared. We primarily test the performance of three algorithms: objective search, novelty search and surprise search, and include three baseline algorithms for comparative purposes. All algorithms use NEAT to evolve a robot controller with the same parameters as in [8], where the maze navigation task and the mazes of Fig. 3 were introduced. Evolution is carried on a population of 250 individuals for a maximum of 300 generations in the medium and hard maze for a fair comparison to results obtained in [8]. However, the number of generations is increased to 1000 for the more deceptive mazes (very hard and extremely hard) to allow us to analyse the algorithms' behaviour over a longer evolutionary period. The NEAT algorithm uses speciation and recombination, as described in [28]. The specific parameters of all compared algorithms are detailed below.

A. Objective search

Objective search uses the agent's proximity to the goal as a measure of its fitness. Following [8], proximity is measured as the Euclidean distance between the goal and the position of the robot at the end of the simulation. This distance does not account for the maze's topology and walls, and can be deceptive in the presence of dead-ends.

B. Novelty Search

Novelty search uses the same novelty metric and parameter values as presented in [8]. In particular, the novelty metric is the average distance of the robot from the nearest neighbouring robots among those in the current population and in a novelty archive. *Distance* in this case is the Euclidean distance between two robot positions at the end of the simulation; this rewards robots ending in positions that no other robot has explored yet. The parameter for the novelty archive (e.g. the initial novelty threshold for inserting individuals to the archive is 6) is as given in [8].

Sensitivity Analysis: While in [8] novelty is calculated as the average distance from the 15 nearest neighbours, the introduction of new mazes in this paper mandates that the n parameter of novelty search is tested empirically. For that purpose we vary n from 5 to 30 in increments of 5 across all mazes and select the n values that yield the highest number of maze solutions (successes) in 50 independent runs of 300 generations for the medium and hard maze, and 1000 generations for the other mazes. If there is more than one n value that yields the highest number of successes then the lowest average evaluations to solve the maze is taken into account as a selection criterion. Figure 5a shows the results obtained by this analysis across all mazes.

The best results are indeed obtained with 15 nearest neighbours for the medium and hard maze, as in [8] (49 and 48 successes, respectively). In the very hard maze there is no difference between 10 and 15 in terms of successes (39) but $n = 15$ yields less evaluations, while in the extremely hard maze $n = 10$ yields less evaluations and more successes (24) than any other value tested. In summary, reported results in Section VII use $n = 15$ for the medium, hard and very hard maze, and $n = 10$ for the extremely hard maze.

C. Surprise search

Surprise search uses the surprise metric of eq. (2) to reward *unexpected behaviours*. As with the other algorithms compared, *behaviour* in the maze navigation domain is expressed

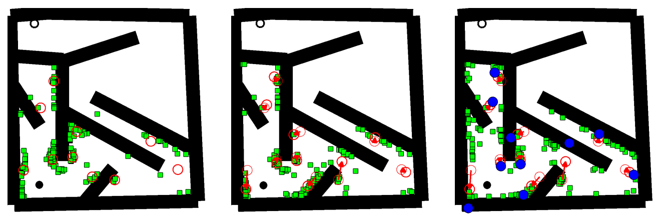
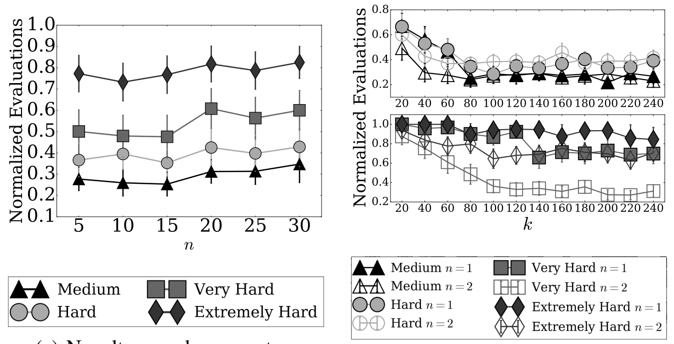

(a) Generation $t-2$ (b) Generation $t-1$ (c) Generation t

Fig. 4: The key phases of the surprise search algorithm as applied to the maze navigation domain. Surprise search uses a history of two generations ($h = 2$) and 10 behavioural clusters ($k = 10$) in this example. Cluster centroids and prediction points are depicted as empty red (light gray in grayscale) and solid blue (dark gray in grayscale) circles, respectively.



(a) Novelty search parameters (b) Surprise search parameters

Fig. 5: **Sensitivity Analysis:** selecting n for novelty search (Fig. 5b), k and n for surprise search (Fig. 5a). Figures depict the average number of evaluations (normalized by the total number of evaluations) obtained out of 50 runs . Error bars represent the 95% confidence interval.

as the position of the robot at the end of a simulation. The behavioural difference d_s in eq. (2) is the Euclidean distance between the robots' final position and a considered prediction point, p.

Following the general formulation of surprise in Section IV-A, the prediction points are a function of a model m that considers k local behaviours of h prior generations. In this comparative study we use the simplest possible prediction model (m) which is a one-step linear regression of two points ($h = 2$) in the behavioural space. Thus, only the two previous generations are considered when creating prediction points to deviate from in the current generation (see Fig. 4). In the first two generations the algorithm performs mere random search due to a lack of prediction points.

The locality (k) of behaviours is determined by the number of behavioural clusters in the population that is obtained by running k-means on the final robot positions. The surprise search algorithm applies k-means clustering at each generation by seeding the initial configuration of the k centroids with the centroids obtained in the previous generation; this seeding process is omitted only in the first generation due to the lack of earlier clusters. This way the algorithm is able to pair centroids in subsequent generations and track their behavioural history. Using the k pairs of centroids of the last two generations, the algorithm creates k prediction points for the current generation through a simple linear projection. Surprise search rewards the behaviour that obtains a surprising outcome given the temporal sequence of the final points of the robot across generations. Surprise search is thus orthogonal to objective and novelty search as it rewards robots that visit areas outside the predicted space(s), without any explicit knowledge of the final goal.

It should be noted that for high values of k the k-means algorithm might end up not assigning any data point to a particular cluster; the chance of this happening increases with k and the sparseness of data (in particular in datasets containing outliers) [47]. Further, the seeding initialization procedure we follow for k-means in this domain aims to behaviourally connect centroids across generations so as to enable us to predict and deviate from the expected behaviour in the next generation. The adopted initialization procedure (i.e. inheriting from centroids of the previous generation) does not guarantee that all seeded centroids will be allocated a robot position during the assignment step of k-means, as robot positions (data points) might change drastically from one generation to the next. In the case of surprise search, when an empty cluster appears in the current generation (i.e. in positions where a cluster existed in a past generation but not currently), then its prediction is not updated (i.e. moved) until a final robot position gets close to the empty cluster's centroid. Predicted centroids that have not been recently updated (due to empty clusters) are still considered when calculating the surprise score, and indirectly act as an archive of earlier predictions. However, this archive is not persistent as the number of 'archived' prediction points can increase or decrease during the course of evolution, and depends on k.

Sensitivity Analysis: To choose appropriate parameters for k (information locality) and n (number of prediction points) in the prediction and deviation models respectively, a sensitivity analysis is conducted for all mazes. We obtain k empirically by varying its value between 20 and P in increments of 20 for each maze. We also test all k for $n = 1$ and $n = 2$ in this paper. As in the sensitivity analysis for novelty search we select the k and n values that yield the highest number of successes in 50 independent runs. If there is more than one k, n combination that yields the highest number of successes we select the combination that solves the maze in the fewest average evaluations.

Figure 5b shows the average number of evaluations for all k values tested, for $n = 1$ and $n = 2$. It is clear that higher k values result to less evaluations on average. Moreover, it seems that $n = 2$ leads to better performance in the two more deceptive mazes. Based on the above selection criteria, we pick $k = 200$ and $n = 1$ for the medium maze, which gives the highest number of successes (50) and the lowest number of evaluations ($16,364$ evaluations on average). For the hard maze we select $k = 100$ and $n = 1$, as it is the most robust (49 success) and fastest ($23,214$ evaluations on average) among tested values. Following the same procedure $k = 200$ and $n = 2$ in the very hard maze, and $k = 220$ and $n = 2$ in the extremely hard maze (see Fig. 5b).

In this paper we started our investigations with the simplest possible prediction model (m), which is a linear regression, and the shortest possible time window of two generations for the history parameter (h). The impact of the history parameter and the prediction model on the algorithm's performance is not examined empirically in this paper and remains open to future investigations. We get back to this discussion in Section IX.

D. Other baseline algorithms

Three more baseline algorithms are included for comparative purposes. *Random search* is a baseline proposed in [8] that uses a uniformly-distributed random value as the fitness function of an individual. The other two baselines are variants of surprise search that test the impact of the predictive model. *Surprise search (random)*, SS_r, selects k random prediction points ($p_{i,j}$ in eq. 2) within the maze following a uniform distribution, and tests how surprise search would perform with a highly inaccurate predictive model. *Surprise search (no prediction)*, SS_{np}, uses the current generation's actual clusters as its prediction points ($p_{i,j}$ in eq. 2), thereby, omitting the prediction phase of the surprise search algorithm. SS_{np} uses real data (cluster centroids) from the current generation rather than predicted data regarding the current generation, and tests how the algorithm performs divergent search from real data. Note that SS_{np} is reminiscent of novelty search, except that it uses deviation from cluster centroids (not points) and does not use a novelty archive. The same parameter values (k and n) are used for these variants of surprise search.

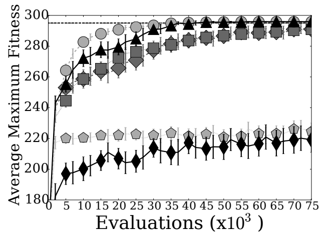
VII. Surprise Search in Authored Deceptive Mazes

The robot maze navigation problem is used to compare the performance of surprise, novelty and objective search. To test the algorithms' performance, we follow the approach proposed in [48] and compare their *efficiency* and *robustness* in all four test bed mazes. We finally analyse some typical examples on both the behavioural and the genotypical space of the generated solutions. All results reported are obtained from 100 independent evolutionary runs; reported significance and corresponding p values are obtained via two-tailed Mann-Whitney U-test, with a significance level of 5%.

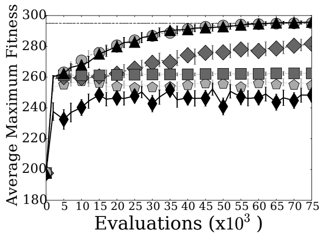
A. Efficiency

Efficiency is defined as the *maximum fitness over time*, where fitness is calculated as $300 - d(i)$; $d(i)$ the Euclidean distance between the final position of robot i and the goal, as in [8]. Figure 6 shows the average maximum fitness across evaluations for each approach for the four mazes.

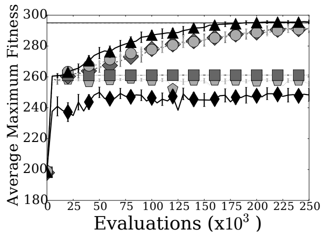
In the medium maze, we can observe that both surprise and novelty search converge after approximately 35,000 evaluations. Even if novelty seems to yield a higher average maximum fitness values than surprise search, the difference is insignificant. Novelty search, on average, obtains a final maximum fitness of 295.84 ($\sigma = 1.47$), while surprise search obtains a fitness of 295.93 ($\sigma = 0.72$); $p > 0.05$. By looking at the 95% confidence intervals, it seems that novelty search yields higher average maximum fitness between



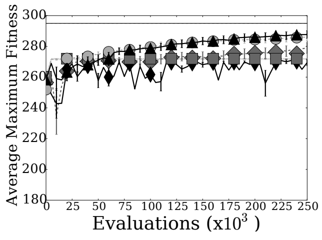
(a) Medium maze



(b) Hard maze



(c) Very hard maze



(d) Extremely hard maze

▲	Surprise Search	◆	Surprise Search (Random)
●	Novelty Search	◆	Surprise Search (No prediction)
■	Objective Search	⬟	Random

Fig. 6: **Efficiency** (average maximum fitness) comparison for the four mazes in Fig. 3. The graphs depict the evolution of fitness over the number of evaluations. Values are averaged across 100 runs of each algorithm and the error bars represent the 95% confidence interval of the average.

7500 and 25,000 evaluations. This difference is due to the predictions that surprise search tries to deviate from. Early during evolution, two consecutive generations may have robots far from each other, lead to distant and erratic predictions. Eventually, we have a convergence and the predictions become more consistent, allowing surprise search to solve the maze. Both objective search and SS_{np} seem fairly efficient to solve the maze; however, they are not able to find the goal in all the runs. The random baselines, instead, perform poorly and show very little improvement as evolution progresses. The baselines' performance proves that surprise search is different from a random walk and that the prediction model positively affects the performance of the algorithm.

In a more deceptive test, the hard maze, we can see from Fig. 6b that novelty and surprise perform much better than all other algorithms; differences in efficiency between novelty and surprise search are not significant. Surprise search and novelty search find the goal in 99 and 93 out of 100 runs respectively, SS_{np} finds the solution in 61 runs, while for the rest of the baselines the success rate is far below. It is interesting to note that objective search reaches a high fitness score around 260 at the very beginning of the evolutionary process, and then it stops to improve. This is due to the dead-end at the upper right corner of the maze (Fig. 3b), which prevents the algorithm from discovering the global optimum. In order to discover the global optimum, in fact, the algorithm needs to explore the least fit areas of the search space, such as the bottom-right corner of the maze.

In the very hard maze, objective search never finds the solution in 100 runs; Fig. 6c shows that this algorithm is not able to reach the goal because, as in the hard maze, it reaches the left-most dead end and is unable to bypass that local optimum. Unsurprisingly, the random and SS_R baselines also perform poorly. On the other hand, novelty search finds the solution in 85 out of 100 runs, while surprise search finds a solution in 99 runs. Interestingly, SS_{np} finds 88 solutions out of 100 runs, similar to novelty search. This can be explained by looking at how SS_{np} is implemented. Its behaviour is quite similar to novelty search as it merely uses the local behaviours of the current generation; the key difference is that these behaviours are clustered in SS_{np}. In such a complex maze surprise search seems to handle maze deceptiveness in a better way; it obtains a final maximum fitness of 295.77 ($\sigma = 3.59$) which is higher (but not significantly) than that of novelty search (292.15; $\sigma = 9.85$); $p > 0.05$. From the confidence intervals of Fig. 7c, it appears that surprise search is performing better or significantly better than novelty search after $50,000$ evaluations until the end of the run.

In the most deceptive (extremely hard) maze, objective search, random and SS_r do not find any solution in 100 runs, performing poorly in terms of efficiency. This is not surprising as all of these algorithms perform consistently poorly in all but the simplest mazes. Novelty search and surprise search find the solution 48 and 67 times, respectively, while the SS_{np} obtains 39 solutions. As can be seen in Fig. 6d, surprise search yields a higher maximum fitness after 150,000 evaluations, with a final maximum fitness of 287.68 ($\sigma = 12.16$).

Another way of estimating efficiency is the effort it takes an algorithm to find a solution. In this case, surprise clearly manages to be more advantageous. In the medium maze surprise search manages to find the goal, on average, in $16,084$ evaluations ($\sigma = 11,588$) which is faster than novelty ($19,814$; $\sigma = 15,441$) and significantly faster than objective search ($48,186$; $\sigma = 23,590$) and SS_{np} ($26,452$; $\sigma = 21,249$). We observe the same comparative advantage in the hard maze as surprise search solves the problem in $23,566$ evaluations on average ($\sigma = 15,925$) whereas novelty search, SS_{np}, and objective search solve it in $28,493$ ($\sigma = 19,939$), $47,550$ ($\sigma = 25,524$) and $73,643$ ($\sigma = 7,542$) evaluations, respectively. Most importantly surprise search is significantly faster ($p < 0.05$) than novelty search in the more deceptive problems: on average surprise search finds the solution in $76,261$ evaluations ($\sigma = 52,385$) in the very hard maze and in $154,794$ evaluations ($\sigma = 84,733$) in the extremely hard maze, whereas novelty search requires $115,600$ evaluations ($\sigma = 81,091$) and $178,045$ evaluations ($\sigma = 86,410$), respectively. Furthermore surprise search is significantly faster ($p < 0.01$) than SS_{np}, which requires $117,560$ ($\sigma = 74,430$) and $200,190$ ($\sigma = 75,569$) evaluations in the very hard and the extremely hard maze, respectively.

The findings from the above experiments indicate that, in terms of maximum fitness obtained, surprise search is comparable to novelty search and far more efficient than objective search in deceptive domains. We can further argue that the deviation from the predictions (which are neither random nor omitted) is beneficial for surprise search as indicated by the performances of SS_r and SS_{np}. The performance of this baseline appears to be similar to novelty search, especially in harder mazes; this is not surprising as SS_{np} is conceptually similar to novelty search, as noted in Section VI-D. It is also clear that, on average, surprise search finds the solution faster than any other algorithm in all mazes.

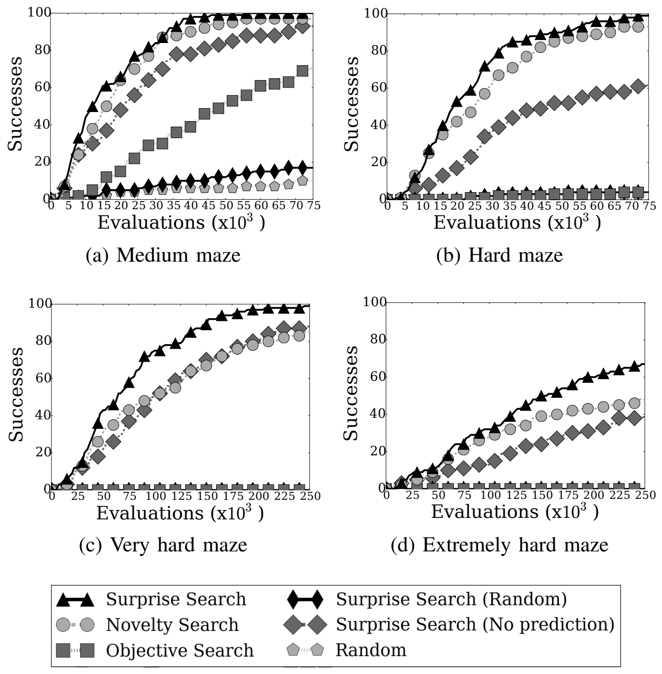


Fig. 7: **Robustness** comparison for the four mazes in Fig. 3. The graphs depict the evolution of algorithm successes in solving the maze problem over the number of evaluations.

B. Robustness

Robustness is defined as the number of successes obtained by the algorithm across time (i.e. evaluations). In Figure 7 we compare the robustness of each approach across the four mazes, collected from 100 runs. In the medium maze (Fig. 7a), surprise search is more successful than novelty search in the first 20,000 evaluations; moreover, surprise search finds, on average, the 100 solutions in fewer evaluations compared to the other approaches. As noticed in the previous section, in the first 20,000 evaluations novelty search has a comparable or higher efficiency in Fig. 6a; this points to the fact that while some individuals in surprise search manage to reach the goal, others do not get as close to it as in novelty search. On the other hand, objective search fails to find the goal in 29 runs, because of the several dead-ends present in this maze. The control algorithm SS_{np} finds the goal 93 times out of 100, but it's slower compared to novelty and surprise search. Few solutions are found by the baseline random search and SS_r, and they are significantly slower than the other approaches. Fig. 7b shows that, in the hard maze, novelty search attains more successes than surprise search in the first 10,000 evaluations but the opposite is true for the remainder of the evolutionary progress. As in the previous maze, this behaviour is not reflected in the efficiency graph (Fig. 6b): this can be explained by how surprise search evolves individuals, as they change their distance to the goal more abruptly, while novelty search evolves behaviours in smooth incremental steps. On the other hand, SS_{np} finds fewer solutions in this maze, 62 out of 100. Finally, the deceptive properties of this maze are exemplified by the poor performance of objective search and the two random baselines.

The capacity of surprise search is more evident in the very hard maze (see Fig. 7c) where the difference in terms of robustness becomes even larger between surprise and novelty search. While in the first 50,000 evaluations novelty and surprise search attain a comparable number of successes, the performance of surprise search is boosted for the remainder of the evolutionary run. Ultimately, surprise search solves the very hard maze in 99 out of 100 times in just 160,000 evaluations whereas novelty search manages to obtain 85 solutions by the end of the 250,000 evaluations. With 88 solutions out of 100, SS_{np} performs similarly to novelty search in this maze but is generally slower compared to surprise search. Objective search and the two random baselines, as expected, do not succeed in solving the maze.

Similarly, in the extremely hard maze (see Fig. 7d) the benefits of surprise search over the other algorithms are quite apparent. While surprise and novelty obtain a similar number of successes in the first 100,000 evaluations, surprise search obtains more successes in the remaining evaluations of the run. At the end of 250,000 evaluations in the most deceptive map examined, surprise search finds solutions in 67 runs versus 48 runs of novelty search. SS_{np} finds 39 solutions and it is generally slower than novelty and surprise search. As in the very hard maze, the remaining algorithms fail to find a single solution to this maze.

C. Analysis

As an additional comparison between surprise and novelty search, we study the behavioural and genotypical characteristics of these two approaches. The behavioural space is presented in a number of typical runs collected from the four mazes, while the genotypical space is inspected through the metrics computed from the final ANNs evolved by these two algorithms. Objective search and the other baselines are not further analysed in this section to emphasise on comparisons between surprise and novelty search.

1) Behavioural Space: Typical Examples: Table I shows pairs of typical surprise and novelty search runs for each of the four mazes; in all examples illustrated the maze is solved at different number of evaluations as indicated at the captions of the images. The typical runs are shown as heatmaps which represent the aggregated distribution of the robots' final positions throughout all evaluations. Moreover, we report the entropy (H) of those positions as a measure of the populations' spatial diversity in the maze. Surprise search seems to explore more uniformly the space, as revealed by the final positions depicted in the heatmaps. The corresponding H values further support this claim, especially in the more deceptive mazes.

2) Genotypic Space: Table II contains a set of metrics that characterize the final ANNs evolved by surprise and novelty search obtained from all four mazes, which quantify aspects of genomic *complexity* and genomic *diversity*. For genomic complexity we consider the number of connections and the number of hidden nodes of the final ANNs evolved, while genomic diversity is measured as the average pairwise distance of the final ANNs evolved. This distance is computed with the compatibility metric, a linear combination of disjoint and excess genes and weight difference, as defined in [28]. As noted in [8], novelty search tends to evolve simpler networks in terms of connections when compared to objective search. Surprise search, on the other hand, seems to generate significantly more densely connected ANNs than novelty search (based on the number of connections). It also evolves slightly larger ANNs than novelty search (based on the number of hidden nodes). Most importantly, surprise search yields population diversity — as expressed by the compatibility metric [8] — that is significantly higher than novelty search. This difference seems to be mostly due to the disjoint factor, which counts the number of mismatching genes between two genomes, depending on whether their genes are within the innovation numbers of the other genome [28]. This suggests that ANNs evolved with surprise search are more diverse in terms of evolutionary history. In the more deceptive mazes, differences in genomic complexity and diversity become significantly larger. In the very hard maze the average number of connections for surprise search grows to 101.94 ($\sigma = 52.45$) while novelty search evolves ANNs with 42.03 connections ($\sigma = 14.53$), on average; the number of hidden nodes used by surprise search is significantly larger (6.94; $\sigma = 3.62$) compared to novelty search. Moreover the diversity metric (compatibility) is around three times that of novelty search. A similar trend can be noticed in the extremely hard maze, where again surprise search evolves denser, larger and more diverse ANNs. As mentioned earlier, handling more complex and larger ANNs has a direct impact on the computational cost of surprise search since it takes more time to simulate new networks across generations. It should be noted that creating larger networks does not imply that this behaviour is beneficial, it is however an indication that surprise search operates differently to novelty search.

VIII. Surprise Search in Generated Mazes

In the previous sections we showed the power of surprise search in four selected instances of deceptive problems. While surprise search outperforms novelty and objective search both in terms of efficiency and robustness in four human-designed mazes, an important concern is whether these results are general enough across a broader set of problems.

In order to assess how surprise search generalises in any maze navigation task, we follow the methodology presented in [49] and test the performance of surprise, novelty and objective search as well as the baselines across numerous mazes generated through an automated process. Moreover, the parameters of k and n which were fine-tuned for the problem at hand in each maze of Section VII are now kept the same, enabling us to observe if a particular parameter setup for surprise search can perform well in unseen problems of varying complexity.

TABLE I: **Behavioural Space.** Typical successful runs solved after a number of evaluations (E) across the four mazes examined. Heatmaps illustrate the aggregated numbers of final robot positions across all evaluations. Note that white space in the maze indicates that no robot visited that position. The entropy ($H \in [0,1]$) of visited positions is also reported and is calculated as follows: $H = (1/logC)\sum_i\{(v_i/V)log(v_i/V)\}$; where v_i is the number of robot visits in a position i, V is the total number of visits and C is the total number of discretized positions (cells) considered in the maze.

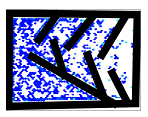
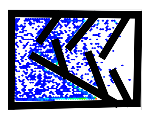
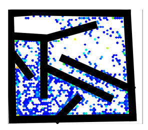
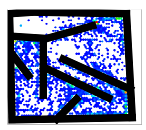
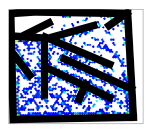
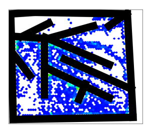
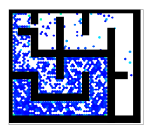
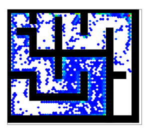
Medium Maze ($E = 25,000$)		**Hard Maze** ($E = 25,000$)		**Very Hard Maze** ($E = 75,000$)		**Extremely Hard Maze** ($E = 75,000$)	
Novelty	**Surprise**	**Novelty**	**Surprise**	**Novelty**	**Surprise**	**Novelty**	**Surprise**
							
0	200	0	110	0	330	0	370
$H = 0.63$	$H = 0.67$	$H = 0.61$	$H = 0.67$	$H = 0.63$	$H = 0.69$	$H = 0.64$	$H = 0.68$

TABLE II: **Genotypic Space**. Metrics of genomic complexity and diversity of the final ANNs evolved using NEAT, averaged across successful runs. Values in parentheses denote standard deviations.

Maze	**Algorithm**	**Genomic Complexity**		**Genomic Diversity**			
		Connections	Hidden Nodes	Compatibility	Disjoint	Weight Difference	Excess
Medium	Surprise	33.76 (15.08)	2.46 (1.53)	42.52 (19.80)	27.40 (16.27)	1.22 (0.25)	11.44 (11.45)
	Novelty	29.08 (6.10)	2.2 (1.0)	32.55 (7.90)	24.97 (7.05)	1.09 (0.26)	4.28 (3.41)
Hard	Surprise	52.34 (28.57)	3.84 (2.66)	73.24 (36.82)	51.60 (27.76)	1.25 (0.24)	17.86 (20.99)
	Novelty	32.55 (9.84)	2.48 (1.29)	39.35 (12.05)	31.06 (11.32)	1.19 (0.28)	4.71 (4.66)
Very Hard	Surprise	101.94 (52.45)	6.94 (3.62)	160.31 (67.01)	121.56 (57.11)	1.26 (0.22)	34.96 (37.34)
	Novelty	42.03 (14.53)	3.27 (1.90)	56.53 (19.66)	46.65 (19.25)	1.16 (0.27)	6.38 (7.71)
Extremely Hard	Surprise	158.16 (89.65)	10.63 (5.85)	260.52 (113.08)	207.97 (107.2)	1.31 (0.23)	48.62 (54.20)
	Novelty	41.79 (11.77)	3.25 (1.53)	54.05 (14.44)	45.30 (14.55)	1.15 (0.24)	5.28 (4.93)

A. Experiment Description

To compare the capabilities in navigation policies of surprise, novelty and objective search in increasingly complex maze problems, we test their performance against 60 randomly generated mazes. These mazes are created by a recursive division algorithm [50], which starts from an empty maze and divides it into two areas by adding a vertical or a horizontal wall with a randomly located hole in it. This process is repeated until no areas can be further subdivided, because doing so would make the maze untraversable or because a maximum number of subdivisions is reached. In this experiment, the starting position and the ending position of the maze have been fixed in the lower left and upper right corner respectively, while the generated mazes have a number of subdivisions chosen randomly between 2 and 6. These values have been chosen empirically to avoid generating mazes that are too easy (solvable by all three methods in few generations) or impossible to solve (because of too many subdivisions). Examples of the mazes generated are shown in Fig. 8. The parameters of surprise search and novelty search are fixed based on well-performing setups with mazes of Section VII: surprise search uses $k = 200$ and $n = 2$ (used in the very hard maze) and novelty uses $n = 15$ (used in medium, hard and very hard mazes). Each generated maze was tested 50 times for each of three methods, measuring the number of successes (i.e. once the agent reaches the goal) in each maze. The number of simulation timesteps is set to 200 and the number of generations to 600.

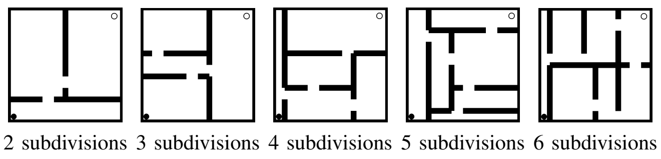


Fig. 8: **Maze generator:** Sample generated mazes (200x200 units) created via recursive division, showing the starting location (black circle) and the goal location (white circle).

B. Results

As a first analysis on the results obtained on the 60 mazes, we focus on which of the evolutionary approaches finds strictly more successes. Table III shows that surprise search has more successes than novelty search in 40% of mazes, while novelty achieves more successes than surprise search in 8% of the generated mazes. Comparing the results of these two approaches against objective search, surprise search outperforms objective search in more mazes (56%) than novelty search (40%). If we look at the baselines, SS_{np} reaches comparable performance to novelty search, but surprise search remains the most successful algorithm, as it outperforms SS_{np} in 45% of the considered mazes. Finally, SS_r and random search evidently perform poorly compared to the other approaches.

An important question to put to the test is how novelty search and surprise search perform with respect to maze deceptiveness. Intuitively, we can say that the deceptiveness (or difficulty) of the maze can be determined by the number of successes obtained by objective search: the more deceptive

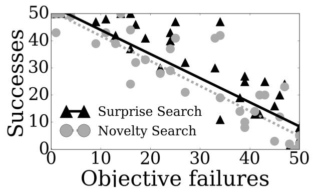


Fig. 9: **Linear regression:** relation between the failures of objective search and successes of surprise and novelty search.

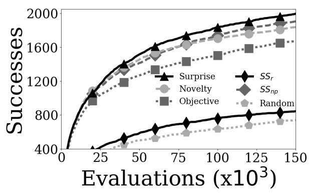


Fig. 10: **Robustness:** algorithm successes in solving all the generated mazes over the number of evaluations for each considered method.

the maze, the more often objective search would fail to find a solution. Figure 9 shows the number of successes obtained by novelty and surprise search against the number of failures obtained by objective search. Unsurprisingly, surprise search and novelty search find mazes where objective search failed more difficult as well, since their successes are highly correlated with the failures of objective search (adjusted $R^2 > 0.85$ for each method, $p < 0.001$). Looking at the trends of the linear regression lines, surprise search constantly achieves a greater number of successes, based on the intercept values of the linear regression models which are significantly different according to an ANCOVA test ($p < 0.05$). Based on the angle of the linear regression line, it also seems that surprise search scales better for more deceptive mazes.

As a final analysis, we report the robustness obtained by aggregating all the runs of the 60 generated mazes for each approach, i.e. a total of 3000 runs. From Fig. 10 we can observe that surprise search is faster, on average, than novelty and objective search in reaching the goal from $112,500$ evaluations onward ($p < 0.05$). Surprise, novelty and objective search require on average $71,865$ ($\sigma = 63,007$), $76,225$ ($\sigma = 64,961$) and $84,398$ ($\sigma = 65,081$) evaluations for each success, respectively. As Fig. 10 shows, some maze problems are easy to solve as all three methods fare similarly in the first $20,000$ evaluations but as the problems become more complex, surprise and novelty search become faster than objective search and eventually surprise search surpasses novelty search in terms of successes. Furthermore, surprise search shows a significant improvement compared to its baseline variants. Respectively, SS_{np}, SS_r and random search obtain $75,531$ ($\sigma = 63,820$), $118,668$ ($\sigma = 53934$) and $123,558$ ($\sigma = 50,568$) evaluations; all results are significantly different from the performance of surprise search ($p < 0.05$).

IX. Discussion

This paper identified the notion of *surprise*, i.e. deviation from expectations, as an alternative measure of divergence to the notion of novelty and presented a general framework for incorporating surprise in evolutionary search in Section IV. In order to highlight the differences between surprise search and other divergent search techniques (such as novelty search) or baselines (such as random search or search with inaccurate predictions), an experiment in the robot maze navigation testbed was carried out and comparisons between algorithms were made on several dimensions. The key findings of these experiments suggest that surprise search yields comparable efficiency to novelty search and it outperforms objective search. Moreover it finds solutions faster and more often than any other algorithm considered. In a broader range of mazes, generated via recursive subdivision, surprise search was also shown to be more robust and generalise well, as it had more successes than novelty search in 40% of generated mazes.

The comparative advantages of surprise search over novelty search are inherent to the way the algorithm searches, attempting to deviate from predicted *unseen* behaviours instead of prior *seen* behaviours. Compared to novelty search, surprise search may also deviate from expected behaviours that exist in areas that have been visited in the past by the algorithm. The novelty archive operates in a similar fashion; however it contains positions (instead of prediction points) and these positions are always considered for the calculation of the novelty score. In surprise search, instead, the prediction points are derived from clusters that characterise *areas* in the behavioural space.

The findings in the maze navigation experiments show a clear difference between novelty and surprise, both behaviourally and genotypically. Surprise search has greater exploratory capabilities, which are more obvious in later generations. Surprise search also creates genotypically diverse populations, with larger and denser ANNs. It is likely this combination of diverse populations, larger and denser networks and a higher spatial diversity that gives surprise search its advantage over novelty search.

Finally, the comparative analysis of surprise search against random search suggests that surprise search is not random search. Clearly it outperforms random search in efficiency and robustness. Furthermore, the poor performance of the two surprise search variants — employing random predictions and omitting predictions — suggests that the prediction of expected behaviour is beneficial for divergent search.

By now we have enough evidence for the benefits of surprise search and enough findings suggesting that surprise search is a different and more robust algorithm compared to novelty search in this domain. Furthermore, through our analysis, we have identified qualitative characteristics of the algorithm that gave us critical insights on the way the algorithm operates. However, we still lack empirical evidence on the reasons the algorithm manages to perform that well compared to other divergent search algorithms. An intuition by the anonymous reviewers of a previous paper about the comparative benefits of surprise search is that the algorithm allows search to *revisit* areas in the behavioural space. Such a behaviour, in contrast, is penalised in novelty search. This difference in how the two algorithms operate leads to the assumption that surprise search is more willing to revisit points in the behaviour space — in a form of *backtracking* or cyclical manner. As a result of this shifting selection pressure in the behavioural space a different strategy is adopted every time a particular area is revisited as each time the ANN controller is different and potentially larger. Such an algorithmic behaviour appears to be beneficial for search and might explain why ANNs get significantly larger in surprise search.

TABLE III: **Successes:** Percentage of generated mazes for which the algorithm in the row has a strictly greater number of successes than the algorithm in the column. The last row and the last column are respectively the average of each column and the average of each row.

	Objective	Novelty	Surprise	SS_{np}	SS_r	Random	Total
Objective	-	15	5	6	71	78	35
Novelty	40	-	8	20	75	81	44.8
Surprise	56	40	-	45	78	85	**60.8**
SS_{np}	51	35	11	-	78	85	52
SS_r	1	1	1	1	-	36	8
Random	0	0	0	0	20	-	4
Total	29.6	19	**5**	16.5	62.8	71.8	-

More importantly than a performance comparison between algorithms, however, is the introduction of surprise as a drive for divergent search. As will be discussed in Section X, the general framework of surprise search can be used with other behaviour characterizations and in other domains. Moreover, while some properties such as the model of prediction are inherent to surprise search, properties such as the clustering of behaviours as well as findings regarding the ability of surprise search to backtrack can be re-used in other divergent search algorithms; examples include a variant of novelty search which considers neighboring cluster centroids rather than neighboring individuals in the behavioural space, or a novelty archive which is pruned (and reduces in size) over time to allow backtracking. We can only hypothesise that the way surprise search operates may result in increased *evolvability* [51], which is an individual's capacity to generate future phenotypic variation, or alternatively, the potential for further evolution. Naturally, all these hypotheses need to be tested empirically in future studies as outlined in the next section.

X. Extensions and Future Work

While this study already offers evidence for the advantages of surprise as a form of divergent search, further work on several directions needs to be performed.

Behaviour Characterization: The surprise search algorithm currently characterises a behaviour merely as a point (i.e. the final robot position on the maze after the simulation time elapses). While such a decision was made in order to compare our findings against the initial results of [8], we currently have no evidence suggesting that surprise search would be able to generalise well in behaviours that are characterised by higher dimensions (e.g. a robot trail). To envision the behaviour of surprise search in a high dimensional space, Fig. 11 shows a possible implementation for surprise search with robot trails, sampled over time. Following the implementation described in Section VI-C, we show the three key phases of the algorithm: the robots' trails are clustered at generation $t-2$ via k-means (see Fig. 11a), then at generation $t-1$ we seed the clustering algorithm with the ones computed in the previous generation and we find the new trajectories' centroids (Fig. 11b). Finally the prediction at generation t is computed via linear interpolation of each point on the computed (centroid) trajectories at generations $t-2$ and $t-1$ (Fig. 11c). Preliminary experiments have shown that predictions of robot trails do not affect the performance of surprise search compared to results in this paper; future studies, however, should investigate the impact of higher dimensional behaviours across several domains, especially on how they affect the predictive model of the algorithm.

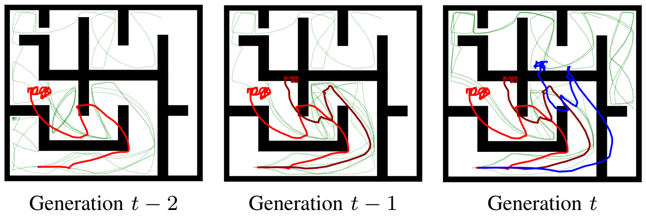


Fig. 11: **Behaviour Characterization:** The key phases of the surprise search algorithm on the maze navigation task, characterizing behaviour via 200 samples of robot positions over time. Surprise search uses a history of two generations ($h=2$) and 10 clusters ($k=10$) in this example (for the sake of visualization, only one cluster is shown). Robot trails are depicted as green lines. Cluster centroids in generations $t-2$ and $t-1$ as well as their predictions are depicted, respectively, as red, dark red and blue lines.

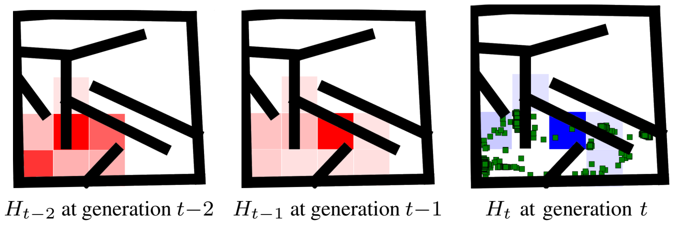


Fig. 12: **Deviation:** Surprise search using heatmaps, at generation t. The first two heatmaps are computed in the last two generations by using the final robot positions, H_{t-2} and H_{t-1}. Using linear interpolation, the difference $H_{t-1} - H_{t-2}$ is computed and applied to H_{t-1} to derive the predicted current population's H_t. The surprise score penalizes a robot if its position (green point) is on a high concentration cell on the predicted heatmap H_t.

Deviation: The current algorithm allows for any degree and type of deviation from the expected behaviour. Inspired by novelty search, this paper only investigated a linear deviation from expectations — i.e. the further a behaviour is from the prediction the better. There exist, however, several ways of computing deviation in a non-linear or probabilistic fashion, e.g. as per [18]. In a maze navigation environment for instance, we can alternatively consider a non-distance-based deviation by using heatmaps of the chosen behaviour characterization. Figure 12 shows the key phases of this implementation: in generation $t-2$ and $t-1$ we compute the heatmaps H_{t-2} and H_{t-1} which map the final positions of all robots ($k=1$), and we use a linear interpolation to compute the predicted heatmap at generation t. The surprise score is then computed by mapping the individual's position on the predicted heatmap: $1 - H_t(x, y)$, where x, y is the final position of the robot mapped onto the heatmap.

Complexity and generality: To a degree, the experiment with 60 generated mazes tests how generalizable surprise search is without explicit parameter tuning. Since results from that experiment indicated that surprise search scales better to more deceptive problems, we need to further test the algorithm's potential within the maze navigation domain through more deceptive and complex environments. The capacity of surprise search will then need to be tested in other domains such as robot locomotion or procedural content generation. As an example, the potential of surprise search has been explored for the generation of unexpected weapons in a first person shooter game [38]. In that work, the considered behaviour characterization is a weapon tester agent's death location: as we have described above, we can employ a heatmap as a probability distribution of the population's behaviour and predict the next generation's heatmap by means of linear interpolation of each singular cell. Surprise search has shown its capacity to generate feasible and diverse content, thereby achieving quality diversity.

Surprise search has also been successfully implemented to evolve soft robot morphologies [52] constrained by a fixed lattice. The goal of this experiment is to create robots able to travel as far as possible from a starting position, within a number of simulation steps. In this task, surprise search was shown to be as efficient as novelty search; moreover, it evolved more diverse morphologies. When computing behavioural distance for this problem, the surprise score is computed by predicting an entire robot trace in all simulation steps, based on past behaviours (traces). This further supports the claim that surprise is unaffected by the dimensionality of the behaviour characterization. The experiments presented in this paper, in [38] and [52] already demonstrate the generalizability of surprise search across three rather diverse domains: maze navigation, game content generation and robot design.

Model of expected behaviour: When it comes to designing a model of expected behaviour there are two key aspects that need to be considered: *how much prior information the model requires* and *how is that information used to make a prediction*. In this paper the prediction of behaviour is based on the simplest form of 1-step predictions via a linear regression. This simple predictive model shows the capacity of surprise search given its performance advantages over novelty search. However we can envision that better results can be achieved if machine learned or non-linear predicted models are built on more prior information ($h > 2$). A possible way of considering more extensive history is to apply linear interpolation of past centroids over time. It is thus possible to compute a 3-dimensional line over the three dimensions considered (x, y, t) and compute the next predicted position over the line by taking the interpolated position at generation t. Linear regression can be easily replaced by a quadratic or cubic regression or even an artificial neural network model or a support vector machine.

Prediction locality of surprise search: The algorithm presented in this paper allows for various degrees of *prediction locality*. We define prediction locality as the amount of local information considered by surprise search to make a prediction. This is expressed by k which is left as a variable for the algorithm designer. Prediction locality can be derived from the behavioural space (as in this paper) but also on the genotypic space. Future investigations should investigate the effect of locality for surprise search. Experiments in this paper (see Fig. 5b) already showcase that algorithm performance is not sensitive with respect to k (as long as k is sufficiently high for the problem at hand).

Clustering: Surprise search, in the form presented here, requires some form of behavioural clustering. While k-means was investigated in the experiments of this paper for its simplicity and popularity, any clustering algorithm is applicable. Comparative studies between approaches need to be investigated, including different ways of dealing with (or taking advantage of) empty clusters.

XI. Conclusions

In this paper, we argue that surprise is a concept that can be exploited for evolutionary divergent search, we provide a general definition of the algorithm that follows the principles of searching for surprise and we test the idea in a maze navigation task. Results show that surprise search has clear advantages over other forms of evolutionary divergent search, i. e. novelty search, and outperforms traditional fitness search in deceptive problems. In particular, surprise search has shown a comparable efficiency to novelty search and, most importantly, to be more successful and faster in finding the goal. Moreover, a detailed analysis of the behaviours and the genomes evolved by surprise search has revealed a more diverse population and a higher exploratory capacity. Finally, the capacity of surprise search to generalize in tasks of increasing complexity is evidently higher when surprise drives the search process, as tested in randomly generated mazes of increasing deceptive properties. These findings support the idea that deviation from expected behaviours can be a powerful alternative to divergent search with key benefits over novelty or objective search.

Acknowledgment

This work has been supported in part by the FP7 Marie Curie CIG project AutoGameDesign (project no: 630665).

References

[1] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, no. 2, 1988.

[2] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[3] D. E. Goldberg, "Simple genetic algorithms and the minimal deceptive problem," in *Genetic Algorithms and Simulated Annealing, Research Notes in Artificial Intelligence*. Morgan Kaufmann, 1987.

[4] L. D. Whitley, “Fundamental principles of deception in genetic search,” in *Foundations of Genetic Algorithms*. Morgan Kaufmann, 1991.

[5] D. E. Goldberg and J. Richardson, “Genetic algorithms with sharing for multimodal function optimization,” in *Proceedings of the Second International Conference on Genetic Algorithms*, 1987.

[6] J. Hu, E. Goodman, K. Seo, Z. Fan, and R. Rosenberg, “The hierarchical fair competition (hfc) framework for sustainable evolutionary algorithms,” *Evolutionary Computation*, vol. 13, no. 2, 2005.

[7] G. S. Hornby, “Alps: the age-layered population structure for reducing the problem of premature convergence,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2006.

[8] J. Lehman and K. O. Stanley, “Abandoning objectives: Evolution through the search for novelty alone,” *Evolutionary computation*, vol. 19, no. 2, 2011.

[9] A. Channon, “Passing the alife test: Activity statistics classify evolution in geb as unbounded,” in *Advances in Artificial Life*. Springer, 2001.

[10] L. Yaeger, “Poly world: Life in a new context,” *Proc. Artificial Life*, vol. 3, 1994.

[11] C. Adami, C. Ofria, and T. C. Collier, “Evolution of biological complexity,” *Proceedings of the National Academy of Sciences*, vol. 97, no. 9, 2000.

[12] M. A. Boden, *The Creative Mind: Myths and Mechanisms*. Routledge, 2004.

[13] G. Ritchie, “Some empirical criteria for attributing creativity to a computer program,” *Minds and Machines*, vol. 17, no. 1, 2007.

[14] G. A. Wiggins, “A preliminary framework for description, analysis and comparison of creative systems,” *Knowledge-Based Systems*, vol. 19, no. 7, 2006.

[15] J. Lehman, K. O. Stanley, and R. Miikkulainen, “Effective diversity maintenance in deceptive domains,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2013.

[16] A. Liapis, H. P. Martínez, J. Togelius, and G. N. Yannakakis, “Transforming exploratory creativity with DeLeNoX,” in *Proceedings of the International Conference on Computational Creativity*, 2013.

[17] J. K. Pugh, L. B. Soros, and K. O. Stanley, “Quality diversity: A new frontier for evolutionary computation,” *Frontiers in Robotics and AI*, vol. 3, p. 40, 2016.

[18] K. Grace, M. L. Maher, D. Fisher, and K. Brady, “Modeling expectation for evaluating surprise in design creativity,” in *Design Computing and Cognition*, 2014.

[19] M. L. Maher, “Evaluating creativity in humans, computers, and collectively intelligent systems,” in *Proceedings of the 1st DESIRE Network Conference on Creativity and Innovation in Design*, 2010.

[20] A. Barto, M. Mirolli, and G. Baldassarre, “Novelty or surprise?” *Frontiers in Psychology*, vol. 4, 2013.

[21] E. Lorini and C. Castelfranchi, “The cognitive structure of surprise: looking for basic principles,” *Topoi*, vol. 26, no. 1, 2007.

[22] G. N. Yannakakis and A. Liapis, “Searching for surprise,” in *Proceedings of the International Conference on Computational Creativity*, 2016.

[23] D. Gravina, A. Liapis, and G. N. Yannakakis, “Surprise search: Beyond objectives and novelty,” in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2016.

[24] G. E. Liepins and M. D. Vose, “Representational issues in genetic optimization,” *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 2, no. 101, 1990.

[25] S. A. Kauffman, “Adaptation on rugged fitness landscapes,” in *Lectures in the Sciences of Complexity*. Addison-Wesley, 1989.

[26] Y. Davidor, “Epistasis variance: A viewpoint on ga-hardness,” in *Foundations of Genetic Algorithms*. Morgan Kaufmann, 1991.

[27] B. Naudts and A. Verschoren, “Epistasis and deceptivity,” *Bulletin of the Belgian Mathematical Society*, vol. 6, no. 1, 1999.

[28] K. O. Stanley and R. Miikkulainen, “Evolving neural networks through augmenting topologies,” *Evolutionary Computation*, vol. 10, no. 2, 2002.

[29] S. Wessing, M. Preuss, and G. Rudolph, “Niching by multiobjectivization with neighbor information: Trade-offs and benefits,” in *Proceedings of the Evolutionary Computation Congress*, 2013.

[30] P. J. Angeline and J. B. Pollack, “Competitive environments evolve better solutions for complex tasks,” in *Proceedings of the International Conference on Genetic Algorithms*, 1994.

[31] S. Ficici and J. B. Pollack, “Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states,” in *Proceedings of the International Conference on Artificial Life*, 1998.

[32] J. D. Knowles, R. A. Watson, and D. W. Corne, “Reducing local optima in single-objective problems by multi-objectivization,” in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2001, pp. 269–283.

[33] K. Deb, “Multi-objective genetic algorithms: Problem difficulties and construction of test problems,” *Evolutionary Computation*, vol. 7, 1999.

[34] J. Lehman and K. O. Stanley, “Beyond open-endedness: Quantifying impressiveness,” in *Proceedings of the International Conference on Artificial Life*, 2012.

[35] ——, “Evolving a diversity of virtual creatures through novelty search and local competition,” in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 2011, pp. 211–218.

[36] J.-B. Mouret and J. Clune, “Illuminating search spaces by mapping elites,” *arXiv preprint arXiv:1504.04909*, 2015.

[37] A. Liapis, G. N. Yannakakis, and J. Togelius, “Constrained novelty search: A study on game content generation,” *Evolutionary Computation*, vol. 23, no. 1, 2015.

[38] D. Gravina, A. Liapis, and G. N. Yannakakis, “Constrained surprise search for content generation,” in *Proceedings of the IEEE Computational Intelligence and Games Conference*. IEEE, 2016.

[39] P. Ekman, “An argument for basic emotions,” *Cognition & emotion*, vol. 6, no. 3-4, 1992.

[40] W.-U. Meyer, R. Reisenzein, and A. Schützwohl, “Toward a process analysis of emotions: The case of surprise,” *Motivation and Emotion*, vol. 21, no. 3, 1997.

[41] A. Ortony and D. Partridge, “Surprisingness and expectation failure: what’s the difference?” in *Proceedings of the Joint conference on Artificial intelligence*, 1987.

[42] L. Macedo and A. Cardoso, “Modeling forms of surprise in an artificial agent,” in *Proceedings of the nnual Conference of the Cognitive Science Society*, 2001.

[43] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner, “Intrinsic motivation systems for autonomous mental development,” *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 2, 2007.

[44] J. Schmidhuber, “Formal theory of creativity, fun, and intrinsic motivation (1990–2010),” *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 3, 2010.

[45] F. Kaplan and V. V. Hafner, “Information-theoretic framework for unsupervised activity classification,” *Advanced Robotics*, vol. 20, no. 10, 2006.

[46] D. Gravina, A. Liapis, and G. N. Yannakakis, “Coupling novelty and surprise for evolutionary divergence,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017.

[47] J. A. Hartigan and M. A. Wong, “Algorithm as 136: A k-means clustering algorithm,” *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[48] G. N. Yannakakis, J. Levine, J. Hallam, and M. Papageorgiou, “Performance, robustness and effort cost comparison of machine learning mechanisms in flatland,” in *Proceedings of the Mediterranean Conference on Control and Automation*, 2003.

[49] J. Lehman and K. O. Stanley, “Novelty search and the problem with objectives,” in *Genetic Programming Theory and Practice IX*. Springer, 2011, pp. 37–56.

[50] A. Reynolds, “Maze-solving by chemotaxis,” *Physical Review E*, vol. 81, no. 6, p. 062901, 2010.

[51] J. Lehman and K. O. Stanley, “Improving evolvability through novelty search and self-adaptation,” in *2011 IEEE Congress of Evolutionary Computation (CEC)*. IEEE, 2011, pp. 2693–2700.

[52] D. Gravina, A. Liapis, and G. N. Yannakakis, “Exploring divergence in soft robot evolution,” in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2017.