

Searching for the Unexpected: Evolution through Surprise



Daniele Gravina
Institute of Digital Games
University of Malta

A thesis submitted for the degree of
Doctor of Philosophy
May, 2019



L-Università
ta' Malta

University of Malta Library – Electronic Thesis & Dissertations (ETD) Repository

The copyright of this thesis/dissertation belongs to the author. The author's rights in respect of this work are as defined by the Copyright Act (Chapter 415) of the Laws of Malta or as modified by any successive legislation.

Users may access this full-text thesis/dissertation and can make use of the information contained in accordance with the Copyright Act provided that the author must be properly acknowledged. Further distribution or reproduction in any format is prohibited without the prior permission of the copyright holder.

Abstract

In this dissertation we present a new approach called *surprise search* that realises the concept of surprise for the serendipitous discovery in a computational search space. Inspired by the notion of surprise in computational creativity, surprise search seeks unconventional solutions and equips computational creators with the ability to search for unexpected outcomes. This new approach contrasts the traditional paradigm of rewarding progress towards the objective, and rewards unexpected discoveries to handle hard and deceptive problems.

According to the literature in computational creativity, surprise is a key element for the discovery of highly creative and unconventional solutions. Furthermore, theories of intrinsic motivation situate surprise, along with novelty, as primary factors for the elicitation of interest, for the enhancement of learning, and for enabling discovery. This thesis tests the hypothesis that surprise can be an effective drive for the discovery of solutions in hard and deceptive testbeds and it also examines how surprise may complement other forms of divergent search such as novelty and quality diversity algorithms. The main contributions of this work include: (1) the introduction of surprise search; (2) the validation of surprise search for problem-solving; (3) the exploration of how surprise can be effectively coupled with novelty search; (4) and the testing of the effectiveness of surprise as a reward for quality diversity. The findings of this thesis support the idea that deviation from expected behaviours can be a powerful alternative for divergent search and quality diversity with key benefits over state-of-the-art evolutionary approaches.

Acknowledgements

This thesis has been made possible thanks to my supervisors, Georgios N. Yannakakis and Antonios Liapis, who have continuously believed in me and guide me towards this difficult but incredibly satisfying goal. I deeply appreciate all the help they gave me during these years.

I would like to say thank you to all the people I shared an office with along these years, Daniel Karavolos, Phil Lopes and David Melhart. It's been a pleasure to share so many good moments and have fun during our boardgame sessions.

A deep thank you goes to all the people at the Institute of Digital Games, who have made this journey somewhat easier, in particular, Costantino Oliva, Stefano Gualeni and Jasper Schellekens.

I would also to give a big thank you to the Institute of Digital Games and the University of Malta, for all the support.

A big big thank you goes to my mother, who has always supported me in my entire life by pushing me to do my best and letting me follow my dreams and desires. I'll never be able to thank her enough.

Last but not least, Cetty, who—as a hero in the moments of needs—has helped me to get through this. Her support was fundamental to finish this major step of my life.

Statement of Originality

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified. (Daniele Gravina)

To my mom, my family and Cetty.

Contents

1	Introduction	1
1.1	Searching for Surprise: a Computational Creativity View	2
1.2	Searching for Surprise: a Bio-inspired View	4
1.3	Searching for Surprise: an Evolutionary Approach	6
1.4	Research Questions	6
1.5	Contributions	7
1.6	List of publications	7
1.7	Structure of the Thesis	8
1.8	Summary	9
2	Related Work	11
2.1	Computational Creativity	11
2.1.1	Surprise	12
2.1.2	Novelty	13
2.1.3	Novelty vs. Surprise	14
2.1.4	Value	15
2.2	Evolutionary Computation	16
2.2.1	Multi-objective Evolution	18
2.2.2	Neuroevolution: NEAT and CPPNs	20
2.3	Quality Diversity and Divergent Search	23
2.3.1	EC-Hardness and Deception	23
2.3.2	Ignoring the Objective: Divergent Search	24
2.3.3	Beyond Divergence: Diverse and Good solutions	27
2.3.4	Curiosity-based Reinforcement Learning	30
2.4	Domains	32
2.4.1	Maze Navigation	32
2.4.2	Soft Robots	34
2.5	Summary	35
3	Surprise Search: the Approach	37
3.1	Surprise Search	37
3.1.1	The Surprise Search Algorithm	38
3.2	Coupling Novelty and Surprise	40
3.2.1	Novelty-Surprise Search Algorithm	41
3.2.2	Novelty Search-Surprise Search Algorithm	42
3.3	Surprise for Quality Diversity	42
3.3.1	Surprise Search with Local Competition	43

3.3.2	Novelty-Surprise Search with Local Competition	44
3.3.3	Novelty Search–Surprise Search–Local Competition	44
3.4	Summary	44
4	Surprise Search: Experiments	47
4.1	Maze Navigation Test Bed	47
4.1.1	Domain	48
4.1.2	Algorithms	49
4.1.3	Experiments and Analysis	52
4.1.4	Generality	57
4.2	Soft Robot Test Bed	61
4.2.1	Domain	61
4.2.2	Algorithms	61
4.2.3	Experiments and Analysis	63
4.3	Discussion	70
4.4	Summary	71
5	Fusing Novelty and Surprise: Experiments	73
5.1	Maze Navigation Testbed	73
5.1.1	Algorithms	74
5.1.2	Experiments and Analysis	75
5.1.3	Generality	79
5.2	Soft Robot Testbed	84
5.2.1	Algorithms	85
5.2.2	Experiments and Analysis	86
5.3	Discussion	89
5.4	Summary	92
6	Surprise for Quality Diversity: Experiments	93
6.1	Maze Navigation Testbed: First Set	93
6.1.1	Algorithms	93
6.1.2	First Set of Generated Mazes	95
6.1.3	Experiments and Analysis	97
6.2	Maze Navigation Testbed: Second Set	104
6.2.1	Algorithms	105
6.2.2	Experiments and Analysis: Authored Mazes	105
6.2.3	Experiments and Analysis: Second Set of Generated Mazes	109
6.3	Discussion	113
6.4	Summary	115
7	Discussion and Conclusions	117
7.1	Limitations	118
7.1.1	Limitations of Surprise Search	118
7.1.2	Limitations of Novelty-Surprise Search	121
7.1.3	Limitations of Surprise for Quality Diversity	122
7.1.4	Limitations of the Domains	123
7.2	Extensibility: Beyond Mazes and Soft Robots	125
7.2.1	Surprise Search for Procedural Content Generation	125

7.2.2 Surprise Search for Reinforcement Learning	130
7.3 Summary	132
Appendices	133
A Experimental Parameters	135
A.1 Parameters for Maze Navigation Experiments	135
A.2 Parameters for Soft Robot Evolution Experiments	136
B Maze Navigation Generated Mazes	139
C Soft Robot Behavioural and Structural Analysis	143
D Computational Effort	147
Acronyms	149

List of Figures

1.1	A high-level representation of the differences between the notions of value, novelty and surprise. We argue that the three notions are orthogonal, as they operate on different dimensions. While value and novelty can be considered <i>static</i> concepts, surprise is <i>dynamic</i> , as it can be viewed as a form of “temporal novelty”. Image inspired by (Maher and Fisher, 2012) and reproduced for the purposes of the thesis.	3
1.2	A card game example to illustrate the difference between novelty and surprise. The cards are ordered from left to right in order of appearance. In Fig. 1.2a the rightmost card is novel, as the shape and colors of this card are different from the ones shown in the other cards; however, we can see that every card in the sequence is novel, and therefore we can predict that the next card will be novel as well. In the second sequence (Fig. 1.2b) the fourth and rightmost card are <i>surprising</i> . A player would expect to see an unseen card after the sequence of the first three cards, but, instead, the fourth card shows a green circle as the first card. Also, the last card is indeed surprising, as it shows a completely new shape and colors instead of another green curved figure, as one would expect by seeing the previous sequence of the first two cards. Image by Yannakakis and Liapis (2016). Authors granted copyright permission.	4
1.3	An evolutionary-inspired analogy to the way novelty and surprise search may operate synergistically for generating diverse individuals. We use parts of the phylogenetic tree of dinosaurs as an example. Blue (dashed line) branches refer to feathered dinosaurs, versus non-feathered (orange, continuous line). Note that this schematic, is a simplification (with notable sample species) of specific phenotypic (feathers) and behavioral (flying) traits and does not necessarily follow phylogenetic lineages often studied by paleontologists. However, the traits investigated are largely in line with the generally agreed origin story of birds (Godefroit et al., 2013).	5
2.1	Graph model example: an agent is visiting a graph made of nodes connected in a fixed configuration. Every step, an agent follows a particular model to decide which node to visit next. In (a) the agent visit only unvisited nodes (to maximise the novelty score), while in (b) the agent is trying to maximise the surprise reward, by deviating from the predictions made with the prediction model.	14
2.2	Evolutionary Loop: a high-level representation of an Evolutionary Algorithm.	16

2.3	A representation of three possible crossover operators.	19
2.4	NEAT genotype and phenotype example. Image inspired by Stanley and Miikkulainen (2002) and reproduced for the purposes of the thesis. . .	21
2.5	Novelty Search. A high-level diagram of the Novelty Search algorithm. The novelty score is computed for every individual in the population by averaging the distance from the closest neighbors collected from the current population and an archive. Image inspired by Liapis (2014) and reproduced for the purposes of the thesis.	26
2.6	Quality Diversity. A high-level diagram of the Quality Diversity paradigm. Given a behaviour characterization (or space descriptor), here depicted as a one-dimensional space, and the desired quality, the goal is to find a set of diverse and high-performing solutions (red points). Image inspired by Cully and Demiris (2018) and reproduced for the purposes of the thesis.	26
2.7	Global vs. Local Competition. A visualization of the resulting exploration in the search space obtained with global competition and local competition. Local competition enables to exploit the explorative capabilities of a divergent algorithm and at the same time to <i>illuminate</i> interesting (i.e., high quality) regions of the search space, that would be instead out-shadowed by the global competition variant.	29
2.8	Maze Navigation. The maze testbeds that appear in (Lehman and Stanley, 2011a) (Fig. 2.8a and 2.8b). The filled circle is the robot’s starting position and the empty circle is the goal.	32
2.9	Robot controller for the maze navigation task. Fig. 2.9a shows the network’s inputs and outputs. Fig. 2.9b 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.	33
2.10	Soft Robot Evolution. Sample of soft robots evolved with a CPPN representation.	34
3.1	Surprise Search. 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_{SS} behavioral characteristics per generation, to predict the expected k_{SS} behaviors of generation t . The surprise score of individual i is the deviation of the behavior of i from a subset of these k_{SS} expected behaviors.	39
3.2	Surprise-based Quality Diversity. The flow chart illustrates a high level representation of the three introduced QD algorithms: SS-LC, NSS-LC, NS-SS-LC. All algorithms are employed as a steady state EA (left flow chart). The model of surprise is initialized after the generation of the initial population and then updated every N offspring generations (right flow chart). The evaluation of individuals (middle flow chart) goes through the calculation of local competition, novelty, and surprise scores before those are assigned to the corresponding algorithm. Algorithmic loops are depicted as gray boxes. The introduced surprise-based components of the algorithm are depicted in orange: light orange refers to the surprise components and dark orange refers to the novelty-surprise components. For the interested reader, references are made to the equations and sections.	43

4.1	Authored Mazes. The maze testbeds that appear in (Lehman and Stanley, 2011a) (Fig. 4.1a and 4.1b) and new mazes introduced (Fig. 4.1c and 4.1d 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 the medium maze and 200×200 units for the other mazes.	48
4.2	Novelty Search: Sensitivity Analysis. Selecting the nearest neighbor for novelty search: the figure depicts the average number of evaluations (normalized by the total number of evaluations allocated) obtained out of 50 runs (of 300 generations for the medium and hard maze, of 1000 generations for the very and extremely hard maze) by varying n_{NS} between 5 and 30 for all four authored mazes examined. The error bars represent the 95% confidence interval of the average.	50
4.3	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 behavioral clusters ($k_{SS} = 10$) in this example. Robots' final positions are depicted as green squares; cluster centroids and prediction points are depicted as empty red and solid blue circles, respectively.	51
4.4	Surprise Search: Sensitivity Analysis. Selecting k_{SS} for surprise search: the figure depicts the average number of evaluations (normalized by the total number of evaluations allocated) obtained out of 50 runs (of 300 generations for the medium and hard maze, of 1000 generations for the very and extremely hard maze) by varying k_{SS} between 20 and 240 for all four mazes examined. The error bars represent the 95% confidence interval of the average.	52
4.5	Efficiency: number of evaluations on average to find a solution for each maze considered. Error bars denote 95% confidence intervals. The maximum number of evaluations is $75 \cdot 10^3$ for the medium and hard maze, $250 \cdot 10^3$ for the very hard and extremely hard maze.	53
4.6	Robustness comparison. The graphs depict the evolution of algorithm successes in solving the maze problem over the number of evaluations.	55
4.7	Maze generator: Sample generated mazes (200×200 units) created via recursive division, showing the starting location (grey filled circle) and the goal location (white circle).	58
4.8	Efficiency: number of evaluations on average by aggregating all the runs of the 60 generated mazes for each algorithm (i.e., 3000 runs per algorithm). Error bars denote 95% confidence intervals; the maximum number of evaluations is $150 \cdot 10^3$	60
4.9	Robustness: algorithm successes in solving all the generated mazes over the number of evaluations for each considered method.	60
4.10	Representation: a CPPN describes the materials of a $5 \times 5 \times 5$ lattice.	62
4.11	Behavior Characterization: behaviour characterization used for objective (the euclidean distance between starting point and ending point) and novelty (the average distance of two trajectories sampled at the same rate; here only 5 samples are shown).	62
4.12	Surprise Behavior Characterization: The key phases of the surprise search algorithm. Surprise search uses a history of two generations ($h = 2$) and 15 clusters ($k_{SS} = 15$) in this example. One cluster's centroid in generations $t-2$ and $t-1$ as well as their predictions are depicted, respectively, as red, dark red and blue lines.	63

4.13	Visualization of the k-means calculation for surprise search: the thick red and green lines are, respectively, two example centroid trajectories obtained by clustering the dotted red and dotted green robot trajectories.	64
4.14	Robustness: number of successes across different thresholds for the 8 lattice's resolutions considered. The graphs depict the number of successes for different performance thresholds.	66
4.15	Structural variety: average number of explored bins for all feature maps. Each bar is normalized by the maximum number of possible bins and error bars display the 95% confidence interval of the average shown.	67
4.16	Feature maps: sample feature maps produced by the three methods, for a single evolutionary run on a resolution of $5 \times 5 \times 5$. White bins do not have any robots, while colored bins denote the fitness of the best individual (blue for low fitness, red for high fitness).	67
4.17	Velocity: mean of velocities over time for the resolution $5 \times 5 \times 5$ (95% confidence interval as error bar).	69
4.18	Walk cycles: walk cycles of different soft robots	69
5.1	Sensitivity analysis: selecting λ for NSS. The figure depicts the average number of evaluations obtained out of 50 runs (of $75 \cdot 10^3$ evaluations for the medium and hard maze, of $250 \cdot 10^3$ evaluations for the very hard and extremely maze). Error bars represent the 95% confidence interval.	76
5.2	Evaluations. Number of evaluations on average to solve the three mazes for each algorithm. Error bars denote the 95% confidence interval. The maximum number of evaluations is $75 \cdot 10^3$ for the medium and hard maze, $250 \cdot 10^3$ for the very hard and extremely hard maze.	77
5.3	Robustness comparison: number of successes in solving the maze problems over the number of evaluations.	78
5.4	Genotypic Space: Connections. Metrics of genomic complexity of the final evolved ANNs, averaged from successful runs. Values in parentheses are 95% confidence intervals.	81
5.5	Genotypic Space: Hidden nodes. Metrics of genomic complexity of the final evolved ANNs, averaged from successful runs. Values in parentheses are 95% confidence intervals.	82
5.6	Efficiency: algorithm successes in solving all the generated mazes over the number of evaluations for each considered method. Error bars denote 95% confidence intervals; the maximum number of evaluations is $150 \cdot 10^3$	83
5.7	Robustness: algorithm successes in solving all the generated mazes over the number of evaluations for each considered method.	83
5.8	Complexity: number of connections and hidden nodes on average for evolved ANNs which solve the mazes per approach. Error bars denote 95% confidence intervals.	84
5.9	Sensitivity analysis of λ. The figure depicts the final average fitness of the fittest individuals obtained from 90 runs across nine λ values in the resolution $5 \times 5 \times 5$. Error bars display the 95% confidence interval of the average shown.	85
5.10	Relation between the resolution of the robots and the maximum fitness of each approach (averaged from 90 runs).	88
5.11	Robustness: number of successes, cumulated on all resolutions, for different performance thresholds.	88

5.12	Structural variety: average cumulative number of explored bins for all feature maps. Each bar is normalized by the maximum number of possible bins and error bars display the 95% confidence interval of the average shown.	89
5.13	Fittest robots evolved in the first run of each approach for the resolutions 3x3x3, 5x5x5, 8x8x8 and 10x10x10 (from top to bottom). Four simulation frames are depicted for each robot.	91
6.1	Maze generation: 10 easiest generated mazes created via recursive division, sorted by NS-LC successes. The starting position (blue filled circle) is at the bottom left corner; the goal position (black empty circle) is at the top right corner. In the caption, R is the number of subdivisions, L is the A* length and S is the number of successes of NS-LC ($n_{LC} = 15$).	96
6.2	Sensitivity analysis: Average number of evaluations across the 10 easiest mazes with four local competition sizes (5, 10, 15, 20), with five values of λ (0.4, 0.5, 0.6, 0.7, 0.8).	99
6.3	Sensitivity to algorithmic components: Average number of evaluations across the 10 easiest mazes for five QD approaches (NS-LC, NSS-LC, SS-LC, SSA-LC, NS-SS-LC) and four divergent algorithms (NS, SS, NSS, NS-SS).	101
6.4	Robustness: number of successes over evaluations by aggregating all the runs of the 60 generated mazes for each approach.	102
6.5	Number of evaluations on average by aggregating all the runs of the 60 generated mazes for each algorithm (i.e., 3000 runs per algorithm). Error bars denote 95% confidence intervals.	103
6.6	Complexity: number of connections and hidden nodes on average for evolved ANNs which solve the mazes per approach. Error bars denote 95% confidence intervals.	104
6.7	Efficiency comparison for the four mazes in Fig. 2.8. The graphs depict the evolution of algorithm successes in solving the maze problem over the number of evaluations. The maximum number of evaluations is $75 \cdot 10^3$ for the medium and hard maze, $250 \cdot 10^3$ for the very hard and extremely hard maze.	107
6.8	Robustness comparison for the four mazes in Fig. 2.8. The graphs depict the evolution of algorithm successes in solving the maze problem over the number of evaluations.	108
6.9	Genotypic Space: Connections. Metrics of genomic complexity of the final evolved ANNs, averaged from successful runs. Values in parentheses are 95% confidence intervals.	109
6.10	Genotypic Space: Hidden Nodes. Metrics of genomic complexity of the final evolved ANNs, averaged from successful runs. Values in parentheses are 95% confidence intervals.	110
6.11	Efficiency: algorithm successes in solving all the generated mazes over the number of evaluations for each considered method.	111
6.12	Robustness: algorithm successes in solving all the generated mazes over the number of evaluations for each considered method.	112
6.13	Complexity: number of connections and hidden nodes on average for evolved ANNs which solve the mazes per approach. Error bars denote 95% confidence intervals.	113

7.1	Deviation in maze navigation: 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	121
7.2	A mockup of Surprise-based MAP-Elites: the idea is to use two different descriptor spaces: the first one is computed as in Mouret and Clune (2015) while the second space acts as a probabilistic distribution model of surprise (<i>Surprise Map</i>) that can be used to evaluate the unexpectedness of the solutions in a probabilistic fashion.	124
7.3	Dynamic maze navigation: a dynamic maze navigation problem, where the obstacles between the starting point and the goal are placed during the simulation with the wheeled robot, e.g., in this example at the timestep 100 and timestep 200.	124
7.4	Surprise-based FI-2pop. Diagram of the two-population genetic algorithm augmented by surprise search used in (Gravina et al., 2016a).	128
7.5	An example of the prediction model of constrained surprise search. The first two sets of heatmaps are computed in the last two generations, H_{t-2} and H_{t-1} ; the death location density is always normalized per floor. 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 truncated to $[0, 1]$. An individual's death locations are mapped to H_t to calculate the surprise score.	129
7.6	Two example weapons created by constrained surprise search. The weapons achieve quality diversity as they both respect balance constraints (quality) and, at the same time, maximize their surprise score (diversity). The weapon on the left creates 'mines' around the map: its bullets are extremely slow, with a large blast area (explosive, high collision radius) and they can also bounce on walls or the level's floor. Moreover, these 'mines' are fired in clusters (high shot cost) thus costing a lot of ammo (of which the weapon has little): the first weapon requires its wielder to move around the level, laying 'mines' in chokepoints when the other player is nearby. Meanwhile, the weapon on the right is very similar to a rifle: high-damage fast bullets which shoot straight (trivial gravity effects) with a very low collision radius, thus requiring precise aiming. Unlike traditional rifles, however, the weapon's bullets have some explosive qualities.	130
7.7	Surprise-based Reinforcement Learning: the flowchart illustrates the two key phases involved in reinforcement learning through surprise search. The figure include the general principles of the exploration-exploitation trade-off (bold) and the algorithmic contributions of this thesis (<i>in italics</i>).	131

List of Tables

2.1	Novelty vs Surprise. A comparison of the key differences between Novelty and Surprise. Table by Barto et al. (2013)	14
4.1	Behavioral 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: $H = -(1/\log C) \sum_i \{(v_i/V) \log(v_i/V)\}$; 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.	56
4.2	Genotypic Space. Metrics of genomic complexity of the final evolved ANNs, averaged from successful runs. Values in parentheses are 95% confidence intervals.	57
4.3	Algorithms tournament: Percentage of 60 generated mazes for which the algorithm in a row has a strictly greater (≥ 1) number of successes compared to the algorithm in a column. Last row and last column are respectively the average of each column and the average of each row.	59
4.4	Efficiency: average efficiency computed from 90 independent runs (95% confidence interval in parentheses). Bold values are significantly different from all the other approaches.	65
4.5	Behavioural analysis: behavioural performance metrics (resolution $5 \times 5 \times 5$) as the mean values of 90 independent runs (95% confidence interval in parentheses). Bold values are significantly different from all other methods.	68
5.1	Behavioural Space. Typical successful runs solved after a number of evaluations (E) on the three 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: $H = -(1/\log C) \sum_i \{(v_i/V) \log(v_i/V)\}$; 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.	80
5.2	Algorithms tournament: Percentage of 60 generated mazes for which the algorithm in a row has a strictly greater (≥ 1) number of successes compared to the algorithm in a column. Last row and last column are respectively the average of each column and the average of each row.	84

5.3	Efficiency: distance covered on average by the fittest individuals collected from 90 independent runs for each method (95% confidence interval in parentheses). Bold values are significantly different from all the other approaches.	87
5.4	Feature maps: feature maps produced by the four methods, by aggregating all the individuals evolved on a resolution of $4 \times 4 \times 4$. White bins do not have any robots, while colored bins denote the fitness of the best individual (blue for low fitness, red for high fitness).	90
6.1	Distribution of the 60 selected mazes per number of subdivisions and corresponding average length of the shortest path computed with the A* pathfinding algorithm.	96
6.2	Final parameters for QD algorithms , based on a sensitivity analysis. Shown is the average number of evaluations (and 95% confidence intervals) across the 10 easiest mazes of Fig. 6.1.	99
6.3	Algorithms tournament: Percentage of 60 generated mazes for which the algorithm in a row has a strictly greater (≥ 1) number of successes compared to the algorithm in a column. Last row and last column are respectively the average of each column and the average of each row.	102
6.4	Algorithms tournament: Percentage of 60 generated mazes for which the algorithm in a row has a strictly greater (≥ 1) number of successes compared to the algorithm in a column. Last row and last column are respectively the average of each column and the average of each row.	112
A.1	NEAT parameter setting. This table shows the NEAT parameters used for the maze navigation experiments conducted in Chapter 4, Chapter 5 and Chapter 6.	136
A.2	Maze navigation parameters. Parameters used for the experiments reported in Chapter 4.	136
A.3	Maze navigation parameters. Parameters used for the experiments reported in Chapter 5.	137
A.4	Maze navigation parameters. Parameters used for the experiments reported in Chapter 6; n_{LC}^1 is used for NS-LC, NSS-LC and NS-SS-LC, n_{LC}^2 is used for SS-LC.	137
A.5	Soft Robot Evolution parameter setting.	138
B.1	60 generated mazes: set introduced in Chapter 4. The starting position (grey filled circle) is at the bottom left corner; the goal position (black empty circle) is at the top right corner.	140
B.2	60 generated mazes: set introduced in Chapter 6. The starting position (grey filled circle) is at the bottom left corner; the goal position (black empty circle) is at the top right corner.	141
C.1	Feature maps: feature maps produced by the four methods, across the 8 resolutions considered. White bins do not have any robots, while colored bins denote the fitness of the best individual (blue for low fitness, red for high fitness).	144
C.2	Behavioural analysis: behavioural performance metrics as the mean values of 90 independent runs (95% confidence interval in parentheses). Bold values are significantly different from all other methods.	145

C.3	Behavioural analysis: behavioural performance metrics as the mean values of 90 independent runs (95% confidence interval in parentheses). Bold values are significantly different from all other methods.	146
D.1	Maze Navigation Computational Effort. CPU time (in seconds) per generation of one indicative run in the maze navigation domain.	147
D.2	Soft Robot Computational Effort: CPU time (in seconds) per generation of one indicative run in the soft robot domain.	148

Chapter 1

Introduction

Over the last 50 years, evolutionary computation (EC) has obtained significant achievements across several complex tasks, ranging from numerical to behavioural optimization. While there are many possible interpretations of what evolutionary computation is, a standard set of components usually involves one or more populations competing for limited resources, a dynamically changing population, a degree of variational inheritance and the ability of the individual to survive and reproduce (De Jong, 2006). The most common optimization approach in artificial evolution is via an *objective function*, which rewards solutions based on their ‘goodness’ (Goldberg and Holland, 1988), 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. One of the implicit assumptions is that the fitness measure, i.e., the driver of the search process, has to be directly correlated with the final objective of the problem. This is a common approach used in the machine learning field (Mitchell et al., 1997): to improve a given solution, the search should follow the gradient towards the objective.

Despite the success of such approaches in a multitude of tasks (Goldberg and Holland, 1988; Michalski et al., 2013), they are challenged in **deceptive** fitness landscapes (Goldberg, 1987), where the necessary stepping stones towards the global optimum are hidden 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 dissertation follows Whitley (1991) and views deception as the *intuitive definition of problem hardness*. The notion of objective function has been inspired by the concept of survival of the fittest in natural evolution, which intuitively selects those individuals that perform better. However, a selection pressure towards quality may cause convergence to a single solution, in contrast to the diversity experienced in the natural evolution. Many algorithms have been proposed to tackle the problem of deception, primarily revolving around diversity preservation (Goldberg et al., 1987; Hu et al., 2005; Hornby, 2006), which deters premature convergence while still rewarding proximity to the objective, and divergent search (Lehman and Stanley, 2011a; Stanton and Clune, 2016; Smith et al., 2016) which abandons objectives in favour of rewarding diversity in the population.

Moreover, there are problems which lack an easily defined objective—or a gradient to reach it. For instance, open-ended evolution studies within artificial life (Channon et al., 2001) do not have a goal state and instead prioritize e.g., survival (Yaeger, 1994; Adami

et al., 2000). In evolutionary art, music or design, a large body of research in computational creativity and generative systems (Boden, 2004; Ritchie, 2007; Wiggins, 2006) focuses on the *creative* capacity of search rather than on the objectives. Ritchie (2007) considers computational creativity on two core properties of a produced solution: *value* and *novelty* (Ritchie, 2007). 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 such as *novelty search* (NS) (Lehman and Stanley, 2011a; Lehman et al., 2013) can be considered as a metaphor of searching for novelty, as it rewards solutions which exhibit dissimilar behaviour from those in the current and the previous populations. An effective use of both in EC can lead to highly novel and valuable at the same time outcomes (Liapis et al., 2013), thus realizing *quality diversity* (Pugh et al., 2016). Quality diversity (QD) algorithms attempt to balance between their individuals' quality and their population's diversity. Quality can be assessed via an objective function, assuming a problem space where this is possible to compute. Diversity, on the other hand, can be assessed in different ways: for instance, *MAP-Elites* (Mouret and Clune, 2015) compartmentalizes the search space beforehand based on two or more behavioral characterizations, while *novelty search with local competition* (NS-LC) (Lehman and Stanley, 2011b) is a the multi-objective approach that pushes for *novelty* as a second objective.

Novelty as a form of search is generally assessed as the distance from the behaviorally closest neighbors in the current population and in an archive of past novel individuals (Lehman and Stanley, 2011a). In that sense, novelty is the deviation from current and past solutions. In natural evolution, the novelty of a behavioral trait such as flying, coupled with a competitive advantage (such as the improved flying ability of Archaeopteryx against flying reptiles of the same period) can lead to massive shifts as these behaviors become dominant. However, according to Grace et al. (2015), novelty and value are not sufficient for the discovery of highly creative and unconventional solutions to problems. Novelty faces many limitations as a measure of diversity; in particular, it lacks a *temporal* dimension in terms of the trends that evolution is following from one generation to the next. While novelty can be considered a static notion, surprise considers the temporal properties of the search, a critical dimension to assess the creativity of the generated solution (Grace et al., 2015; Maher, 2010). Further studies in general intelligence and decision making support the importance of unexpectedness for problem-solving (Barto et al., 2013).

1.1 Searching for Surprise: a Computational Creativity View

Driven by the notion of computational surprise for the purpose of creatively traversing the search space towards unexpected or serendipitous solutions, this thesis introduces the notion of *surprise* as a mechanism for 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 novelty. Surprise search (SS) is built upon the novelty search (Lehman and Stanley, 2011a) 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 (Lorini and Castelfranchi, 2007): as such, it is different to novelty which rewards deviation from past and current behaviours.

Based on the large volume of work in computational creativity (Grace et al., 2015, 2014;

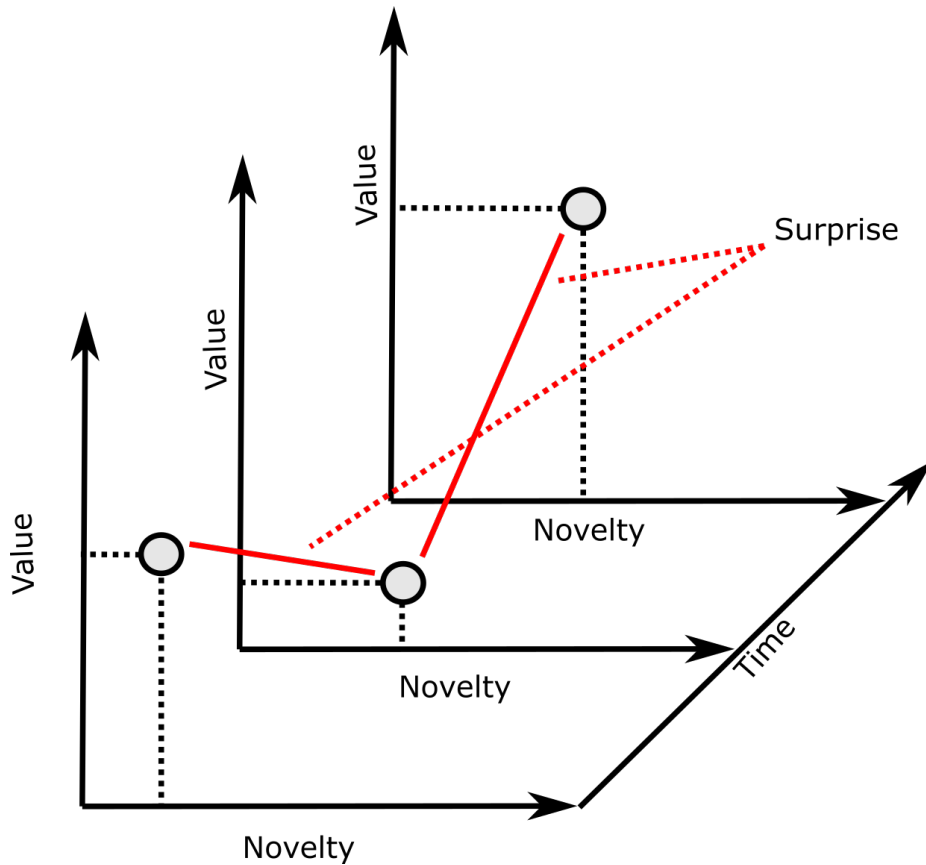


Figure 1.1: **A high-level representation of the differences between the notions of value, novelty and surprise.** We argue that the three notions are orthogonal, as they operate on different dimensions. While value and novelty can be considered *static* concepts, surprise is *dynamic*, as it can be viewed as a form of “temporal novelty”. Image inspired by (Maher and Fisher, 2012) and reproduced for the purposes of the thesis.

(Maher et al., 2013; Macedo and Cardoso, 2002; Macedo et al., 2009; Macedo and Cardoso, 2001) we can claim that surprise is different from novelty and value. A novel and valuable outcome may not be necessarily surprising, e.g., we *expect* something novel based on the historical trends of the previous outcomes. Surprise is indeed tailored by novelty, but it arises from breaking the expectations (Maher et al., 2013) and not merely by new events. We can envision that surprise operates on a different space compared to novelty (see Fig. 1.1): expectations necessarily imply a temporal dimension and hence surprise can be viewed as form of temporal novelty. Novelty, however, works on the space of already seen or existing results (Yannakakis and Liapis, 2016). To exemplify the difference between the notions of novelty and surprise, Yannakakis and Liapis (2016) use 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 (Fig. 1.2). The novelty of the next card is the highest if all past revealed cards are different. The surprise value of the same card, however, is low as the player has grown to expect a new, unseen, card every time. However, if seen cards are revealed after a while then the novelty of next cards decreases, but surprise increases as the game deviates from the expected behavior which calls for a new card every time.

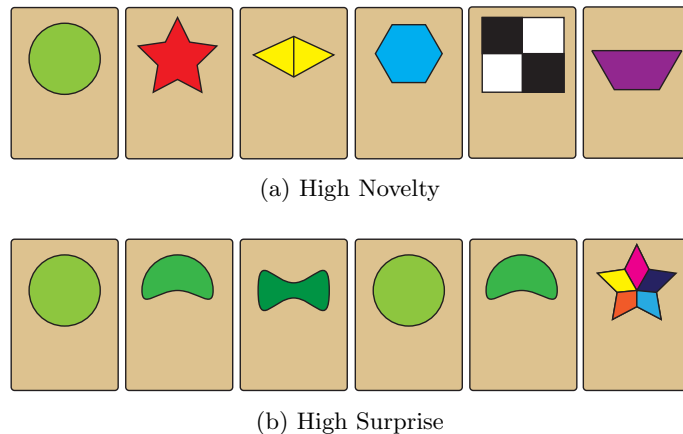


Figure 1.2: **A card game example to illustrate the difference between novelty and surprise.** The cards are ordered from left to right in order of appearance. In Fig. 1.2a the rightmost card is novel, as the shape and colors of this card are different from the ones shown in the other cards; however, we can see that every card in the sequence is novel, and therefore we can predict that the next card will be novel as well. In the second sequence (Fig. 1.2b) the fourth and rightmost card are *surprising*. A player would expect to see an unseen card after the sequence of the first three cards, but, instead, the fourth card shows a green circle as the first card. Also, the last card is indeed surprising, as it shows a completely new shape and colors instead of another green curved figure, as one would expect by seeing the previous sequence of the first two cards. Image by Yannakakis and Liapis (2016). Authors granted copyright permission.

1.2 Searching for Surprise: a Bio-inspired View

Drawing an evolutionary analogy for novelty and surprise, one may view those processes through the lens of *phylogenetics* (Wiley and Lieberman, 2011); the study of the evolutionary history and relationships among organisms. Both algorithms may represent evolutionary lineages that operate synergistically on the behavioral (rather than on the genetic) space of a phylogenetic tree. While both processes can be seen as *behavioral lineages* of evolution, on the one hand, novelty search rewards diversity by aggregating the entire evolutionary history into a novelty archive, on the other hand, surprise search considers the recent historical trends to make predictions and deviate from them. Placing the processes within an evolutionary tree (see Fig. 1.3), novelty is most likely to reward extensions of current branches in the tree whereas surprise is expected to benefit higher branching factors over time. Drawing again from natural evolution in paleontology, the shift towards smaller, flying dinosaurs in the Cretaceous period (Lee et al., 2014) pointed to a trend towards ever-smaller fliers, as evident in most birds in following epochs: such birds would not be deemed surprising. Differently, some birds evolved into man-sized flightless bipedal predators (Bertelli et al., 2007) (“terror birds”): this breaks expectations and phenetic tendencies; therefore terror birds are deemed surprising. However, terror birds are not behaviorally novel since similar-sized bipedal predators abounded in earlier periods (i.e., most carnivorous dinosaurs) (Paul, 1988).

This analogy doesn’t imply that natural evolution actively uses the process of surprise, as arguably it is hard to imagine how nature could generate a prediction for future generations

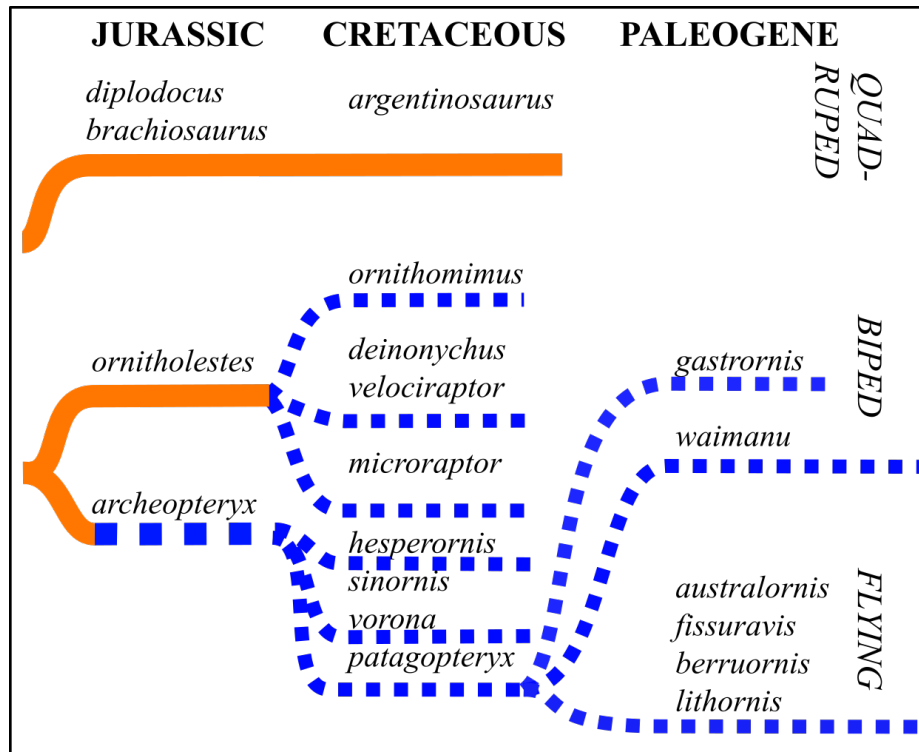


Figure 1.3: **An evolutionary-inspired analogy to the way novelty and surprise search may operate synergistically for generating diverse individuals.** We use parts of the phylogenetic tree of dinosaurs as an example. Blue (dashed line) branches refer to feathered dinosaurs, versus non-feathered (orange, continuous line). Note that this schematic, is a simplification (with notable sample species) of specific phenotypic (feathers) and behavioral (flying) traits and does not necessarily follow phylogenetic lineages often studied by paleontologists. However, the traits investigated are largely in line with the generally agreed origin story of birds (Godefroit et al., 2013).

and then deviate from it. We argue that the notion of surprise could be used as a *lens* through which we can understand how to reproduce certain lineages of natural evolution. Nonetheless, studying how natural evolution works is beyond the scope of this thesis, and we leave the question on whether surprise search resembles natural evolution open for future investigations.

Another interesting analogy to take in consideration is human design evolution. In (Grace et al., 2014) it is argued that human design creativity can be fully understood only by taking into consideration the notion of surprise, along with novelty and quality. An example of this can be found by analyzing the trend in length of mobile phones in the mid-2000s (Grace et al., 2014). If we analyze this trend, it is possible to notice that in the previous decade mobile phones were getting smaller and smaller; this trend was reversed with the introduction of multi-touch and large-screen mobile phones, breaking the expectation to see even smaller devices. Therefore based on the trends observed in the recent history of mobile phones, we can consider this new direction *surprising*. As in the previous analogy, we don't claim that this process was actively searching for unexpected events. On the contrary, surprise is more the ending result of unknown processes within these evolutionary systems; we propose to use the notion of unexpectedness to replicate the

serendipitous discovery obtained by those processes.

1.3 Searching for Surprise: an Evolutionary Approach

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. Unlike novelty, surprise accounts for behavioral trends in recent generations and uses them to predict future behaviors: if a new individual breaks those expectations, then its behavior is surprising and the individual is favored for evolution. While surprise is tied to human emotions (Ekman, 1992), in the context of EC it is more broadly defined as *deviation from expected behaviors*. Based on that definition, surprise can be considered both orthogonal and complementary to novelty: the latter deviates from past behaviors, while the former deviates from predicted future behaviors. With the theoretical argument of the importance of surprise as a diversity mechanism, we suggest that surprise may constitute 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.

1.4 Research Questions

This dissertation aims to test the hypothesis that searching for unexpected solutions is highly beneficial for the purposes of evolutionary search. Essentially, we can summarize the main research question as:

“How does the notion of surprise affect evolutionary search?”

As this question is quite general and can be interpreted in several ways, we need to break it down to more specific questions:

- Q1. How can we formalize effectively the notion of surprise for use in computational search?
- Q2. When and under which circumstances can surprise solve optimization problems?
- Q3. How can surprise be interweaved more efficiently with other quality and diversity measures?

In order to answer the above questions, in this thesis we introduce several surprise-based algorithms which are tested in a number of experiments. We address Q1 in Chapter 3, where we devise a general formalization of the concept of surprise in the context of evolutionary search, and, building on this formalization, we set out several extensions of surprise-based search. In Chapter 4 we answer Q2, by performing a number of experiments that test the optimization capabilities of surprise in deceptive domains. Finally, we complete our investigation by addressing Q3 in Chapter 5 and Chapter 6. In particular, in Chapter 5,

we test how surprise can be coupled with novelty for evolutionary divergence; in Chapter 6, we test different extensions of surprise search with other quality diversity algorithms.

1.5 Contributions

This thesis, in our attempt to test our key hypothesis, contributes to the current state of the art in the following ways:

- **Surprise Search:** The introduction and formalization of surprise as a quantitative measure. By modeling surprise, not only we do attempt to advance our knowledge in understanding the phenomenon, but we equip artificial creators with capacities to search for surprising outcomes.
- **Optimization through Surprise:** The implementation and validation of surprise search in two different and difficult domains: maze navigation and soft-bodied morphology design.
- **Coupling Novelty and Surprise:** The formalization and implementation of the combination of two diverging and orthogonal algorithms, surprise search and novelty search. Supported by the theoretical orthogonality of the two notions, this idea is implemented in two formulations (novelty-surprise search and novelty search-surprise search), tested in two domains, and compared against their base components.
- **Novelty, Surprise and Value:** The introduction of surprise as a quality diversity mechanism. Three new extensions of surprise search, namely surprise search with local competition, novelty-surprise search with local competition and novelty search-surprise search-local competition are introduced and compared to state-of-the-art quality diversity algorithms.

1.6 List of publications

In our attempt to answer the research questions introduced in Section 1.4, we achieved results that were published as a series of papers, listed below:

1. Daniele Gravina, Antonios Liapis and Georgios N. Yannakakis: “Quality Diversity Through Surprise”. In *IEEE Transactions on Evolutionary Computation*. *IEEE*, (2019).
2. Daniele Gravina, Antonios Liapis and Georgios N. Yannakakis: “Fusing Novelty and Surprise for Evolving Robot Morphologies”. In *Proceedings of the Genetic and Evolutionary Computation Conference*. *ACM*, 2018.
3. Daniele Gravina, Antonios Liapis and Georgios N. Yannakakis: “Coupling Novelty and Surprise for Evolutionary Divergence”. In *Proceedings of the Genetic and Evolutionary Computation Conference*. *ACM*, 2017.
4. Daniele Gravina, Antonios Liapis and Georgios N. Yannakakis: “Exploring Divergence in Soft Robot Evolution”. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. *ACM*, 2017.

5. Daniele Gravina, Antonios Liapis and Georgios N. Yannakakis: “Constrained Surprise Search for Content Generation”. In *Proceedings of the IEEE Conference on Computational Intelligence and Games. IEEE, 2016.*
6. Daniele Gravina, Antonios Liapis and Georgios N. Yannakakis: “Surprise Search: Beyond Objectives and Novelty”. In *Proceedings of the Genetic and Evolutionary Computation Conference. ACM, 2016.*

1.7 Structure of the Thesis

This dissertation is organized as follows:

- **Chapter 2:** In this chapter, we review relevant studies and discuss the research areas relevant to this work. Specifically, we extensively describe related work on computational creativity and evolutionary computation. Furthermore, we provide in-depth and detailed descriptions of deceptive problems, divergent search and quality diversity.
- **Chapter 3:** In this chapter, we focus on how the notion of surprise can be formalized quantitatively. A description of the surprise search algorithm and its main components and details on the prediction model and deviation are given. Moreover, five variants of surprise search are presented. Two of them test the idea of fusing novelty and surprise; the remaining three interweave the notion of surprise with quality diversity algorithms. The following chapters refer to the algorithms described in this chapter and provide the implementation details depending on the domain tested.
- **Chapter 4:** In this chapter, we test the surprise search algorithm on two deceptive domains, maze navigation and soft robot evolution. We compare the performance of this new algorithm against two approaches, novelty search and objective search, and we collectively test its capabilities against four authored mazes, 60 generated mazes and eight instances of the soft robot evolution problem.
- **Chapter 5:** In this chapter, we introduce two extensions of surprise search (novelty-surprise search and novelty search-surprise search) and we then investigate the effectiveness of coupling novelty and surprise in one unique formulation. As in the previous chapter, these two introduced algorithms are tested in the maze navigation and soft robot evolution domains.
- **Chapter 6:** In this chapter, we explore how the notion of surprise impacts the performance of state-the-art quality diversity approaches, though three introduced surprise-based algorithms. Furthermore, this chapter presents a new methodology to test quality diversity approaches in a challenging set of maze navigation problems. The three algorithms that are introduced and extensively tested are named: surprise search with local competition, novelty-surprise search with local competition and novelty search-surprise search-local competition.
- **Chapter 7:** This chapter summarizes the main findings of this dissertation and discusses the main limitations found in the proposed approaches, domains and experiments. In order to address these limitations, a number of possible directions are proposed for future work.

Many contributions of this thesis are based on previous publications listed in Section 1.6. In particular, the algorithms of Chapter 4 have been presented in publications 3, 6, and 1. The maze navigation results described in Chapter 5 have been reported in publication 6; the soft robots results described in the same chapter have been reported in publications 2 and 4. In Chapter 6 we report an analysis of results that have been partially reported in publication 3, while the soft robots results have been presented in publication 2. The experiments conducted in Chapter 7 have been partially presented in publication 1. In Chapter 6 we report original results obtained with novelty search-surprise search in the domain of soft robot evolution. The same applies for the results of the surprise-based quality diversity algorithms in the mazes introduced in Chapter 5. Finally the analysis of the limitations and extensions in Chapter 8 has been discussed across all the listed publications and significantly extended.

1.8 Summary

Motivated by the importance of unexpectedness as a mechanism to find unconventional and diverse solutions, in this chapter we introduced the main contribution of this dissertation: *surprise as a viable and effective reward for evolutionary search*. Several studies in computational creativity stress the importance of unexpectedness as a mechanism for finding unconventional and diverse solutions. Supported by this theoretical work, we introduced a new algorithm that implements surprise in the form of evolutionary search and we named it surprise search. Surprise is introduced as evolutionary divergence and as quality diversity measure, and tested in two domains, maze navigation and virtual creature evolution. Our findings support the idea that surprise is an effective approach in deceptive domains and its orthogonality with other diversity and quality measures make surprise search a useful evolutionary tool. The remainder of this work is dedicated to answer extensively the questions arisen in this chapter.

Chapter 2

Related Work

This dissertation discusses the notion of surprise as a potential form of divergent search for computational creativity which is manifested as unconventional problem-solving. For that purpose in this chapter we first draw inspiration from the literature in computational creativity and attempt to define surprise; we then compare it against the notions of novelty and value (Ritchie, 2007) which arguably define the most popular criteria of creativity assessment for computational outcomes (Section 2.1). A second crucial domain from which this work is built upon is evolutionary computation: in Section 2.2 we offer a detailed description of artificial evolution and we present some recent advances in the field. Section 2.3 describes the main challenges faced by fitness-based evolutionary approaches when handling deceptive problems. Divergent search and quality diversity paradigms are then introduced as solutions for these problems and we describe why they are important for the purposes of this thesis. Given the necessity to test the proposed algorithms in a testbed, this chapter ends with a description of the domains against which the performance of surprise-based search is validated (Section 2.4).

2.1 Computational Creativity

Computational creativity focuses on the challenge of creating unconventional computational outcomes, that can be recognized by humans as creative (Boden, 2004). However, creativity, being such a non-trivial concept to define precisely, has led to different definitions, starting from search heuristics to criteria for the assessment of the outcome of an artificial creator. A common definition shared within the field is to consider computational creativity along two core properties of a produced solution: *value* and *novelty* (Ritchie, 2007). 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. Boden (2004) and Ritchie (2007) agree that in order to assess creativity novelty and value are essential, and Wiggins (2006) argues that value and novelty are key features of creativity. However Grace and Maher (2014) argue that novelty and value are not sufficient for the discovery of highly creative and unconventional solutions to problems, and *surprise* might be an essential aspect to consider to solve difficult problems (Kulkarni and Simon, 1988). Another valuable contribution on this topic is from Boden (2004), where she defines creativity as the ability to come up with ideas or artefacts that are new, surprising and valuable.

The distinction between novel and surprising outcomes comes from the observations that high valuable artefacts are not merely novel but also unexpected. The concept of

novelty, however, does not cater for the temporal dimensions involved in the process of unexpectedness, which in turn can be defined as temporal novelty (Maher and Fisher, 2012). Grace et al. (2014) and Maher and Fisher (2012) suggest that surprise should be a core assessment of a creative outcome, while Macedo and Cardoso (2001) and Macedo et al. (2009) argue that surprise is also involved in the process of generating something creative. Based on the literature on computational creativity, it is possible to argue that surprise, novelty and value are essential concepts that grant a computational agent to be creative. In order to use them efficiently in an algorithm, we need to formalize these three concepts computationally. Therefore, this section will briefly review the relevant literature related to these three concepts and clarify their differences.

2.1.1 Surprise

The study of surprise has been central in neuroscience (Donchin, 1981), psychology (Ekman, 1992), and cognitive science (Ortony and Partridge, 1987; Kulkarni and Simon, 1988), and to a lesser degree in computational creativity and computational search. In neuroscience, particular event-related brain potentials that can be attributed to unexpected events have been recognized and thus can be used as predictors of unexpectedness and event memorability (Donchin, 1981). In the literature of affective modeling and psychology, surprise defines one of the six basic emotion of Ekman (1992). In cognitive science studies, it is possible to find several definitions of surprise: surprise has been viewed as a violation of a belief (Ortony and Partridge, 1987), a temporal-based cognitive process of the unexpected (Meyer et al., 1997; Lorini and Castelfranchi, 2007), or a reaction to novelty (Wiggins, 2006). Within computational creativity, coupled with novelty and value, surprise is often attributed to a core component of a creative outcome. For instance, in (Grace et al., 2015; Maher, 2010; Maher and Fisher, 2012; Maher et al., 2013) surprise is associated to a creative output of a computational designer, or used as an incentive for creative exploration in agent modeling (Macedo and Cardoso, 2002; Macedo et al., 2009; Macedo and Cardoso, 2001). Computational models of surprise have also been employed to recognize important features in images (Itti and Baldi, 2006) or to create novel data visualization techniques (Correll and Heer, 2017), to guide automatic computational scientific discovery (Kulkarni and Simon, 1988).

Several types of surprise have been suggested in the literature. For instance, one important distinction has been made by Ortony and Partridge (1987) and Grace et al. (2015), where surprise is described as either *active* or *passive*. Active surprise involves an intentional act of prediction: an explicit expectation about an event is violated, and therefore it elicits the emotional response. On the contrary, passive surprise is a stimulus given by assumptions deriving from earlier experiences, which resembles more the concept of novelty, as it will be described in the following subsection. Inspired by the relevant literature on the subject, in this thesis we view surprise as the degree to which expectations about a solution are violated through observation (Grace et al., 2014). In particular, we introduce the notion of surprise in the context of evolutionary computation (Yannakakis and Liapis, 2016; Gravina et al., 2016b, 2017c). We will give a detailed description of surprise-based evolutionary search in Chapter 3.

2.1.2 Novelty

Similarly to surprise, also the concept of *novelty* has been central in several disciplines. For instance, a first distinction given by Berlyne (1960) is between short, long-term or complete novelty. *Complete novelty* is something that has never been experienced before, *long-term novelty* is something that has not been experienced for some time, and finally *short-term novelty* is when something does not occur in the last few minutes. A further possible distinction is between absolute novelty and relative novelty (Berlyne, 1960). *Absolute novelty* happens when its features have never been encountered before, whereas *relative novelty* is something that is novel but with familiar features, i.e., a novel combination of known elements (Berlyne, 1960).

In order to model a novelty detector computationally, we are required to formalize the above distinctions. A conventional approach is to model the probability density of the underlying distribution and then label as novel those inputs that fall in the low-probability area of the distribution space. Also known as anomaly detection, these techniques try to identify if a particular input varies significantly from the usual distributions of observations. Many techniques exist to estimate the probability distribution given a set of finite samples, such as parametric methods (R.O. and P.E., 1973) or non-parametric (Markou and Singh, 2003). The shared principle across all the statistical methods is simple: novelty is proportional to the inverse of the probability of incidence.

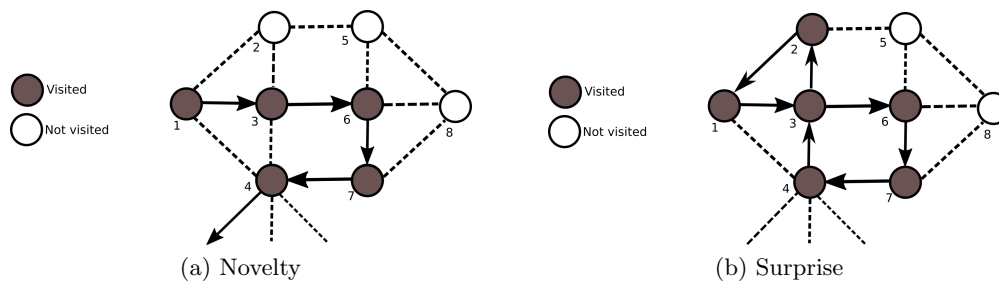
A different view on novelty involves the use of *memory* to archive the past experiences. If something has never been experienced before, it will not appear in the memory, and therefore it can be considered novel. A fading memory, i.e., the oldest elements in the memory are progressively deleted, can represent either short or long-term memory, depending on the fading rate. However, such novel detection system can only return a boolean value and it does not consider the complexity resulting from the use of a memory, such as the metric to be used when storing an experience in memory or the computational cost of using an archive that can potentially grow to infinity. A more sophisticated system may consider adopting a similarity measure, which enables to evaluate different degrees of novelty. An example could be using outlier detection through a clustering algorithm (Markou and Singh, 2003) or a novelty detector that employs an unsupervised learning algorithm such as self-organizing feature maps (Kohonen, 2012) or autoencoders (Liapis et al., 2013).

In evolutionary computation studies, novelty has been defined as the degree to which a specific artefact differs from previous outcomes (Lehman and Stanley, 2011a). In particular, in the novelty search algorithm, novelty is measured based on the distance from the nearest (behaviourally) neighbouring individuals existent in the current population and an archive, which represents the memory of the past experiences. More details about the use of novelty within EC are given in Section 2.3.2, where we present a detailed definition of novelty search.

Admittedly, both the memory-based and stochastic definitions of novelty might seem similar to the notion of surprise. In particular, when the computation of novelty involves a model to estimate the probability of an event, this can be considered a prediction of the occurrences of the event. This resembles the formalization of surprise as described in Section 2.1.1. The next subsection will shed lights on the main differences between these two notions.

Table 2.1: **Novelty vs Surprise.** A comparison of the key differences between Novelty and Surprise. Table by Barto et al. (2013)

	Novelty	Surprise
<i>Type of knowledge store</i>	Memory, memory recall	Predictor, prediction
<i>Time</i>	Time not a key factor: items in memory are always available	Incoming data usually compared with a temporalized prediction
<i>Processes for novelty/surprise triggering</i>	Experience does not match memory	Two phases: (1) Formulation of prediction, (2) Prediction is violated

Figure 2.1: **Graph model example:** an agent is visiting a graph made of nodes connected in a fixed configuration. Every step, an agent follows a particular model to decide which node to visit next. In (a) the agent visit only unvisited nodes (to maximise the novelty score), while in (b) the agent is trying to maximise the surprise reward, by deviating from the predictions made with the prediction model.

2.1.3 Novelty vs. Surprise

Novelty and surprise are different notions by definition as it is possible for a solution to be both novel and/or surprising to various degrees. Following the core principles of Lehman and Stanley (2011a) and Grace et al. (2015), 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 behaviors. Prior information is required to predict what is expected; hence a *prediction of the expected* (Maher, 2010) is a necessary component for modeling surprise computationally. By that logic, surprise can be viewed as a *temporal novelty* process or as novelty on the prediction (rather than the behavioral) space. Table 2.1 summarizes the key differences between the two notions.

The difference between novelty and surprise can be exemplified by considering an agent travelling a graph made of nodes connected in a fixed configuration, starting from the node labelled as 1. Every step the agent has to decide which node to visit next, and based on a certain model it will get a reward based on the decision made. The objective of the agent is to maximise the immediate reward of every step, while the final objective is to visit as many different nodes as possible. Two different models are used, where respectively the agent is rewarded when it visits a novel node i or an unexpected node i . In the first

case, an agent would probably try to maximise novelty by looking for novel discovery in every step it takes in the graph: this can easily lead to a situation as in Fig. 2.1a. In this case, the sequence of discoveries is $1 \rightarrow 3 \rightarrow 6 \rightarrow 7$: in this configuration, the agent has to decide to visit either node 4 or node 8, but as they are both novel, the agent will get the same reward no matter what it decides. Therefore, if it decides to visit node 4, the agent will never visit nodes 2, 5 and 8, as backtracking (i.e., visiting already visited nodes) is highly discouraged by the novelty reward. Instead, a surprise approach would try to deviate from the patterns learned while visiting the nodes of the graph. In this case, the surprise model has higher chance to visit nodes not visited in the previous example, as backtracking could be the result of a self-surprising behaviour. If we take the same sequence as before ($1 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 4$), surprise would encourage visiting already seen nodes (for example node 3); if its prediction model has learned to expect a new node in every step, visiting an already seen node is unexpected. Therefore, a surprising approach might lead to visit nodes not explored by novelty, thanks to backtracking. Given the new sequence, after visiting node 3, the expectation for the next node changes, and the prediction might be to visit a *seen* node, e.g., 1. In this new state visiting node 2 has higher unexpectedness, and the agent would be encouraged to visit a node that in the previous example has never been reached. However, the drawback of this model is that a surprising agent can easily get stuck visiting the same nodes in a loop, as backtracking can cause a “circular behaviour” (e.g., it will forever visit already visited points) as pointed out in Fig. 2.1b.

Since novelty and surprise can be considered different notions of diversity, we can imagine combining the two for the purpose of evolutionary search. This idea, introduced in (Gravina et al., 2017a) and extensively tested in (Gravina et al., 2018, 2019b) will be described in Chapter 3.

2.1.4 Value

Ritchie (2007) defines *value* as the degree to which a produced item is of high quality in its domain. In computational art and aesthetics value is by definition a subjective trait, which can be estimated by using human ratings given by experts in the field. However, for creative problem solving, value can be measured more formally, and usually it captures the quality of the solution of the problem. For instance, in an optimization scenario, a conventional way to define an objective is to compute the distance from the desired goal through a cost function, which in evolutionary computation is commonly known as objective function, i.e., the function value of the objective is proportional to the quality of the solution. Recent findings, however, have questioned this conventional formalization of the objective function, as in deceptive problems (Whitley, 1991) the direct use of the distance to the goal might be counterproductive to reach the global optimum of the problem, as we will describe in more detail in Section 2.3.

Generally speaking, novelty, surprise and value can be considered orthogonal concepts. In fact, while novelty and surprise share some common traits highlighted in the previous subsection, it is possible to use them as three separated dimensions: value stands for the direct assessment of the global goal, novelty for the distance to the nearest behaviours, and surprise for the distance from the expectations. By employing these three concepts in computational search, we enhance the creativity capacity of search algorithms. Particular benefits of the approach can be found on deceptive problems or problems where the objective is not easily definable, as in open-ended evolution or computational creativity. There are ways of integrating value in divergent search e.g., via constraints that accepted artifacts should

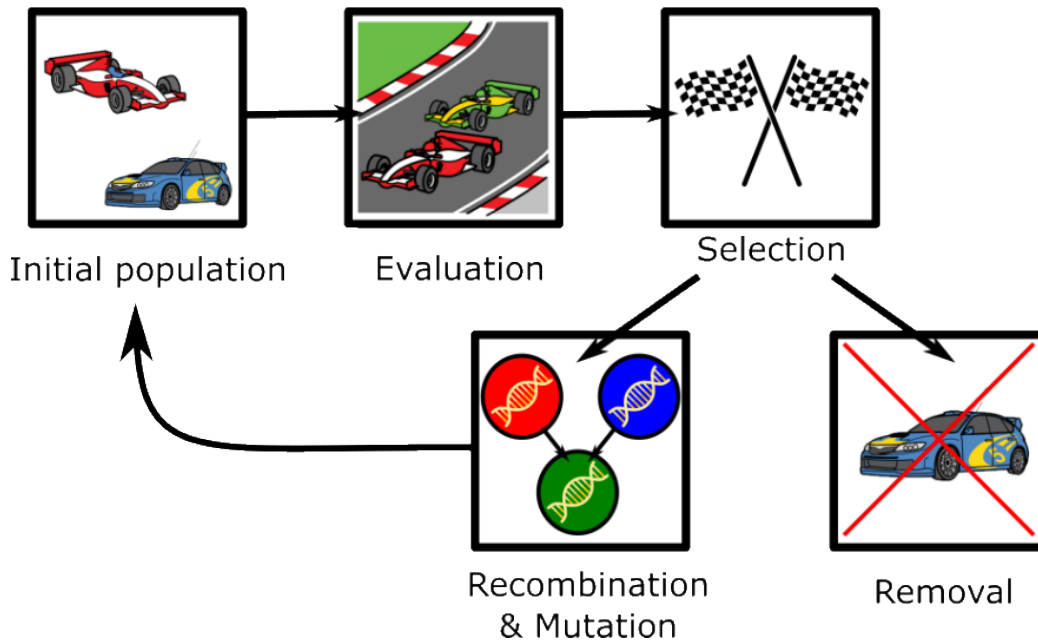


Figure 2.2: **Evolutionary Loop:** a high-level representation of an Evolutionary Algorithm.

have a minimum value (i.e. fitness score) while individuals satisfying these constraints can evolve towards divergence. Examples of constrained novelty search, in particular, have been proposed by Liapis et al. (2015) and Lehman and Stanley (2010) for problem solving tasks (level generation and maze navigation respectively), as well as Vinhas et al. (2016) for evolutionary art and Liapis et al. (2013) for novel game object generation. Quality diversity (Pugh et al., 2016), for instance, is a novel paradigm that tries to successfully combine the converging behaviour of the objective with the diverging properties of divergent search synergistically. Surprise can similarly be combined with value and novelty, for instance by linear aggregation (Gravina et al., 2017a), multiobjective evolution (Gravina et al., 2019b) or by constraint satisfaction (Gravina et al., 2016a). In this thesis, we first introduce an approach that aggregates novelty, surprise and value in Chapter 3 and then we test it in Chapter 6.

2.2 Evolutionary Computation

Evolutionary computation is a computational process inspired by natural evolution that optimizes a given problem through *combination* and *mutation* of a set of solutions, which are *evaluated* and then *selected* based on a performance measure. Artificial evolution was initially introduced in the 1950s to translate the malleability of the natural evolution process into a computational algorithm, and from then a vast number of variants have been introduced (Holland, 1992; Koza, 1994; Rechenberg, 1973). Although there are several variations of EC in literature, it is possible to extract the common characteristics shared by this family of algorithms (Mitchell, 1998). This section offers an overview of the key features and the principal components of evolutionary algorithms, while Chapter 3 will provide the details of the evolutionary algorithms proposed in this dissertation.

The *representation* of the individuals is the first component to take into consideration, i.e., how to codify a solution that can be used and modified by the algorithm. Depending

on the problem, designing a suitable representation may be an arduous task to accomplish. A good representation should be able to consider all the possible solutions in the search space and at the same time it should be computationally efficient and easy to alter; furthermore, it needs to show robustness and resilience to modifications, as it might undergo several mutations over the course of the evolutionary process. Traditional nomenclature in EC distinguishes the representation of an individual with two different perspectives: the *Genotype* is the low-level representation of the individual in the search space, while the *Phenotype* is the image of the genotype in the solution space, where the evaluation of individual's performance happens. This mapping between the search space and the solution space should be consistent, in order to preserve the *locality* of the mutations, i.e., preserving the relative distance between two genomes and their respective images in the solution space. Further attention should be given at the phenomenon of *epistasis* (Davidor, 1991); the interaction between the genes should be kept as low as possible, as we will explain in Section 2.3. In literature a vast number of representations have been proposed, such as traditional representations made of a finite sequence of bits or floats to more advanced and complex representations such as syntax trees (Koza, 1994) or artificial neural networks (Stanley and Miikkulainen, 2002).

The second fundamental aspect that drives the behaviour of the evolutionary algorithm is the reproduction mechanism, or how the genetic information is exchanged in the search space between different individuals. Reproduction is achieved through two evolutionary operators, traditionally named *crossover* and *mutation*. The first operator, crossover, merges the genome of two individuals to produce a novel solution for the successive generations. Several crossover implementations can be conceived, depending on the problem and representation chosen. A traditional implementation firstly segments the genotype of a pair of selected parents from the current population; it then recombines the segments of the first parent with the complementary segments taken from the other parent, originating two new offspring. Different segmentation techniques are possible. For instance, a genotype can be divided by using one (One-Point Crossover) or several cutting points (N-Point Crossover) as Fig. 2.3a and Fig. 2.3b show. An alternative implementation of this operator selects the genes randomly from the two parents and swap them according to a predefined probability (Fig. 2.3c). Furthermore, the offspring may go through an additional operator named *mutation*. Mutation is an attenuate modification of the genome, which yields new points in the solution space. Like crossover, we can find several implementations of the mutation operator, depending on the representation used and the application. In the case of a vector of real-valued numbers, a genotype can be modified by one, multiple or all its genes based on a stochastic distribution. A debate on the importance of these two evolutionary operators is still ongoing (Spears, 1993), as their contribution to the exploration of the search space is different: mutation pushes for random diversity in the population, while crossover promotes potential emergent behaviours already present in the population (Spears, 1993). However, most of the researchers agree that their usefulness depends on the problems' properties, the representation used and the evolutionary implementation chosen.

Measuring the value of each individual in the population is commonly known as *evaluation*. The evaluation process assigns a fitness value to the individuals, which affect its probability of selection. This process guarantees that the positive traits are safeguarded across the generations, ensuring progress towards the desired goal. Traditionally, individuals are evaluated objectively, meaning that their fitness is evaluated independently of the population they are inserted in. However, recent work has started to question the validity of this approach, especially in the case of *deceptive* problems. EC-hardness and deception

will be covered extensively in Section 2.3.

Another critical component of EC is the *selection* and *replacement* mechanism. Two strategies can be adopted for selection, deterministic or probabilistic, but the stochastic process is preferred as it ensures higher individual diversity and exploration. Stochastic selection helps the evolutionary process to escape local optima and to explore more effectively the search space. A number of selection mechanisms have been described in the literature, such as *elitist selection*, which select only the n best individuals from the population, *tournament selection*, which runs a tournament between n randomly selected individuals, or *roulette-wheel selection*, which selects individuals proportionally to their fitness compared to the global fitness of the population. Replacement decides how to remove and insert individuals in the population based on the offspring generated through the reproduction. There are two primary replacement methods: generational and steady state. *Generational replacement* replaces the entire population with the offspring, while *steady-state* replaces only one individual, e.g., the worst individual in the population. On top of the generational replacement, usually, also *elitism* is applied, which preserves from the selection mechanism n fittest individuals, in order to preserve the best solutions in the population from the environmental selection.

We summarize the evolutionary loop in Fig. 2.2. The starting point is the creation of an initial population of individuals, either randomly or deterministically. Then the created individuals are evaluated and, accordingly to their fitness, selected based on the selection method chosen. The selected individuals undergo the reproduction phase, and the generated offspring replace a number of individuals of the population. Then, the process restarts from the evaluation phase until the termination condition is reached, which depends on the one's requirements. Typical termination conditions used are either a fixed number of generations or the achievement of a satisfactory solution.

Building on the notion of evolutionary search, in this thesis we propose to search for unexpected solutions. This approach challenges the traditional view of the objective as main drive for evolutionary computation by introducing a new way of searching for high-performing solutions. Being a critical aspect to take in consideration for the approaches introduced in this thesis, we will provide a detailed description of the evolutionary components of surprise-based algorithms in the following chapter (Chapter 3).

2.2.1 Multi-objective Evolution

As their name suggests, multi-objective evolutionary algorithms (MOEAs) (Deb, 2001) aim to find the best trade-off between two or more objectives. These algorithms are appropriate for problems which have more than one desirable properties, each of which could be considered objectives in their own right. Often, in these problems one objective may be in conflict with another, so optimizing for one objective would end up lowering the score of others, hence dominating them. When objectives are conflicting it is natural to expect that any solutions found cannot optimize all objectives at the same time. Instead MOEAs attempt to find solutions which are *not dominated* in at least one dimension; in particular, a solution A it is said to dominate (*Pareto dominance*) another solution B if and only if Eq. (2.1) and Eq. (2.2) are true. These solutions are called Pareto non-dominated solutions.

$$\forall i : A_i \geq B_i \tag{2.1}$$

$$\exists i : A_i > B_i \tag{2.2}$$

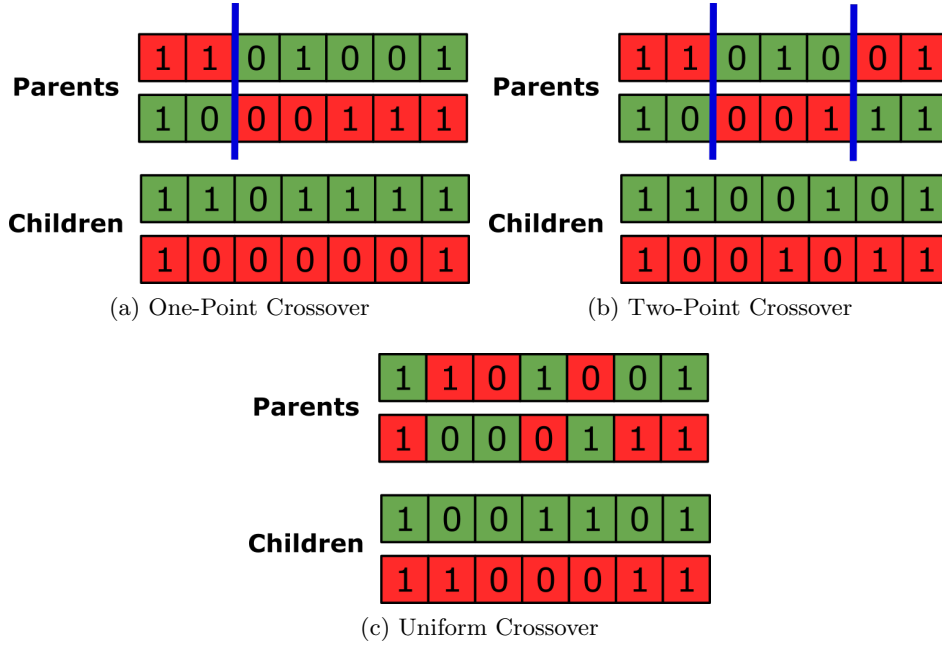


Figure 2.3: A representation of three possible crossover operators.

where, with an abuse of notation, A_i and B_i are the i -th objective of solution A and B respectively.

Several approaches have been proposed within the area of multi-objective evolutionary search (Coello et al., 2007) including ranking selection, aggregation-based selection of objectives and Pareto front optimization. In one of the most naive approaches, aggregation-based selection combines the desired properties of the solution into a weighted sum. This simple approach has a core limitation: as the search space must be explored for finding the best weights for each objective particular solutions may be missed as the Pareto front might not be convex. In other words, aggregation-based selection can discover solutions only on the linear fronts defined by the selected weights. On the other hand, ranking selection favors solutions according to a user-defined priority of the objectives. For example, if we want to optimize the cost and velocity of a car, we may want to select a solution that does not exceed a certain cost limit, and then opt for the vehicle that optimizes the performance from the selected subset. Based on the notions of Pareto dominance, a dominance-based approach can be used instead to drive evolution. NSGA-II by Deb et al. (2002) is one of the most adopted of such approaches (and for MOEAs in general). In this thesis, we use extensively NSGA-II to combine effectively surprise search with other evolutionary approaches (Gravina et al., 2019b). These approaches will be described in Chapter 3 and validated in Chapter 5 and Chapter 6.

Non-dominated Sorting Genetic Algorithm II

The Non-dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al., 2002) is generic MOEA based on the original formulation of NSGA. The algorithm produces a population of competing individuals and sorts them based on their non-domination ranking, which is assigned as fitness to every individual. It then uses evolutionary operators (crossover and

mutation) to generate the offspring, which are combined with the parents before splitting the resulting population into fronts. To aid the exploration the fitness landscape, NSGA-II employs a measure called *crowding distance* to preserve the diversity within the same ranking and used as a tie-breaker in the selection process.

The algorithm starts by initializing a population and dividing it into non-dominated fronts; for each individual, a rank is assigned based on the front it belongs to. In addition, a crowding distance measure is assigned to every individual based on how close it is to its neighbors in the same front. Parents are selected based on the rank and the crowding distance and the offspring is generated by means of crossover and mutation. The current population and the generated offspring are joined and sorted again based on non-dominated sorting, and the best N individuals are selected for the next generation.

2.2.2 Neuroevolution: NEAT and CPPNs

In evolutionary computation, the choice of representation is often considered of “vital importance” (Schoenauer, 1996). Representation encoding influences the genotype space (Michalewicz and Hartley, 1996), which in turns influences the phenotype space based on the chosen mapping between the two (Bentley and Kumar, 1999). As noted in the previous section, a standard approach involves a direct mapping between the genotype and phenotype and the use of a bit arrays, which theoretically can encode all the possible phenotypes that can be stored on computer memory. However, for longer encodings, there is an exponential growth of the possible combinations of bits which makes the search for the desired solutions challenging and ultimately impractical. In practice, it is often desirable to design a genotype of the minimal size that can encode a large space of phenotypes. Formally, we would like to maximise the *pleiotropy*, i.e., how a single gene can influence multiple phenotypic traits, and minimize the *polygeny*, i.e, how a single phenotypic trait can be influenced by interactions of multiple genes (Fogel, 1995). However, such compressed information in the genotype requires a complex function to decode the information into a phenotype, if even such a function exists. Furthermore, complex representations are often accompanied by complex genetic operators (crossover and mutation) which can make the computation burden too high compared to the benefits of a powerful representation. Another requirement for indirect mapping is to minimize the possible augmentation of the effects due to the genome mutations: a minimal change in the genotype should have a similar impact in the phenotypic space. Therefore it is required to carefully design the genetic operators, especially in the case of complex indirect representation (Bentley and Kumar, 1999). Several representations have been proposed in the literature, such as syntax trees in Genetic Programming, where it is possible to evolve directly pieces of code, or artificial neural networks (ANNs), a subfield of EC that goes under the name of neuroevolution.

Inspired by the structure of the natural brain, artificial neural networks are densely connected nodes (Mitchell et al., 1997), where each node is commonly referred to as “neurons”. These neurons can take as input multiple real-valued inputs and return a single real number value that acts as output. The output is the result of a combination of a weighted sum of the inputs with a non-linear activation function, such as a rectifier or a sigmoid function. Neuroevolution (NE), the evolution of artificial neural networks, has shown great promise in several complex task (Gomez and Miikkulainen, 1999). Neuroevolution searches in the encoded space of ANNs for a network that performs well in a specific task. Several algorithms have been proposed through the years to evolve ANNs, from evolving the weights of a fixed topology (Whitley et al., 1990) to evolve at the same time weights and structure

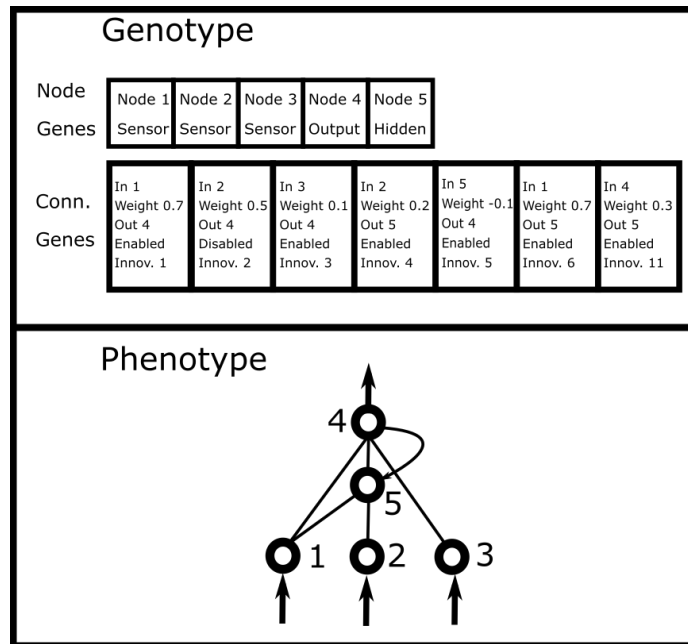


Figure 2.4: **NEAT genotype and phenotype example.** Image inspired by Stanley and Miikkulainen (2002) and reproduced for the purposes of the thesis.

of the ANNs (Gruau et al., 1996). While a debate on the advantages of evolving both the weights and the structure is an open area of research (Gomez and Miikkulainen, 1999), a number of challenges have to be addressed.

A range of solutions has been proposed to evolve weights and the structures of the ANNs simultaneously. The first problem is how to encode the networks efficiently. The encoding can be direct or indirect: the former explicitly list all the connection between nodes and the list of nodes; the latter instead specify rules to infer the phenotypes of the ANN. Direct encodings are usually simpler to implement, but they can easily blow up with the number of nodes and connections between them. On the other hand, indirect encodings are more compact, but they can be unpredictable, as small changes in the encodings can influence the search in unpredictable ways (Braun and Weisbrod, 1993). The second most significant issue is related to the problem of competing conventions. Competing convention means that two different encodings represent the same artificial neural network, i.e., two different encodings maps to the same phenotype. This can lead to damaged phenotypes after the recombination of two networks, and consequently to lose critical information (Stanley and Miikkulainen, 2002). The third problem is related to the protection of innovation. In neuroevolution, innovation takes place when a network changes its structure through mutation, e.g., adding a connection between two genes. This often leads to a decrease in performance, as the ANN needs time to find the optimal weight for the newly added connection. Therefore, if not explicitly protected, novel networks will not survive through evolutionary selection because of their initial suboptimal performance (Stanley and Miikkulainen, 2002).

NEAT

In order to address these problems, Stanley and Miikkulainen (2002) proposed an approach called Neuroevolution of Augmenting Topologies (NEAT). NEAT bypasses the difficulties

mentioned above by starting from simple networks and complexifying them via recombination and mutation throughout evolution. The networks are encoded as a linear representation of nodes and connections between them (see Fig. 2.4). Each genome has a list of nodes (input, hidden and outputs) and a list of connections, which specify their in-node, out-node, and connection weight. To allow a meaningful crossover, NEAT introduces the concept of historical markings, which keep track of every new structural component. Every time a new structure is created, this is marked by a unique identification. Furthermore, historical markings address the third problem, the protection of innovation. They enable the use of speciation, as the species are computed based on the number of shared genes with the same identification. Speciation is used to maintain genetic diversity and to give some room for improvement to newer and more complex networks.

Mutation in NEAT can modify the weights of the connections or add new structures, i.e., new connections between different nodes or new nodes. Crossover instead uses the historical markings to line up the genes from two different genomes. When lined up, the offspring randomly select the genes from one of the two parents, and it always selects non-matching genes from the fittest parent. Historical markings are fundamental for the process of complexification, which finds simple ANNs at the beginning of the evolutionary process, and then it builds more complex network starting from them. The complexification process establishes order to the complexity of the ANNs evolved, as the simpler ones, i.e., with fewer nodes and connections, will appear first in the evolutionary history of the algorithm. This means that different fitness definitions can influence the size of the final evolved ANNs, as the following sections will show. NEAT has been validated through several works, and it is widely applied across several domains (Stanley and Miikkulainen, 2004; Stanley et al., 2005a; Aaltonen et al., 2009; Reisinger et al., 2007; Stanley et al., 2005b). In this thesis, we use NEAT to evolve controllers able to navigate complex mazes and to evolve soft robot morphologies. We describe these two domains in the next sections.

While neuroevolution through NEAT is a robust and widely validated approach, it can only encode neural networks activated by sigmoids. However, a more powerful encoding might unleash the power of evolution as a seeker of complex and interesting structure or morphologies. An open area of research revolves around the concepts of *developmental encodings* (Stanley, 2007), which tries to find an efficient way to encode information capable of expressing an extensive range of phenotypes, as DNA does in nature. In the spirit of this research field, Stanley (2007) has proposed a new encoding, called Compositional Pattern Producing Networks (CPPNs), that tries to describe the complex relationships that arise in nature using functions compositions. A CPPN is an artificial neural network with nodes of different activation functions (e.g., sine, sigmoid, Gaussian, etc.) which allow regularities, repetitions and other patterns to emerge. Being CPPNs structurally similarly to ANNs, they can be evolved by existing neuroevolution methods, such as NEAT (Stanley and Miikkulainen, 2002). CPPNs have demonstrated their representational power in multiple domains, such as image generation (Secretan et al., 2008), three-dimensional object generation (Clune and Lipson, 2011), robots controllers (Risi and Stanley, 2013) and procedural content generation (Liapis et al., 2013; Hastings et al., 2009). In the several experiments conducted in this dissertation, we evolve artificial neural networks acting as robotic controllers or designers of complex shapes.

2.3 Quality Diversity and Divergent Search

In this section we describe the notions of deception, divergent search and quality diversity. In particular, we review what are the major source of hardness for evolutionary computation and why deception is one of the most difficult to address. We then describe two recent paradigms that have addresses deception in evolutionary computation, divergent search and quality diversity.

2.3.1 EC-Hardness and Deception

The term *deception* in the context of evolutionary computation was introduced by Goldberg (1987) 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. Whitley (1991) argues that “the only challenging problems are deceptive”. However, a debate on what makes a problem difficult is still ongoing, and the role of deception contentious. For instance, Mitchell et al. (1992) argue that deception is only one of the many components that can make fitness landscape difficult, and Grefenstette (1993) points out that deception is neither a necessary or a sufficient condition to define a problem hard, as it depends on the dynamic of the sampling strategy. Admittedly, it is difficult to define what makes a problem hard. Given a generic description of the algorithm and the instance of the problem, a priori measure of its performance does not exist, and the only way to measure the algorithm’s effectiveness is by running it (Rice, 1953).

For instance, other measures of EC-hardness are the sampling error (Liepins and Vose, 1990) and a rugged fitness landscape (Kauffman, 1989). In combinatorial optimization problems, the fitness landscape can affect optimization when performing local search. Such algorithmic process assumes a high correlation between the fitness of neighboring points in the search space and independent genes in the chromosome. The latter assumption is commonly referred as *epistasis* (Davidor, 1991) 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 (Davidor, 1991). As noted, epistasis is evaluated from the perspective of the fitness function and thus is susceptible to deception; Naudts and Verschoren (1999) argue that deceptive functions cannot 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 neighbors’ in the search space, or epistasis, i.e., the degree of interaction among genes’ effects. Another interesting take on the EC hardness is from Borenstein and Poli (2004). They argue that the analysis of the difficulty for a certain problem can be improved by assessing the properties of the problem’s fitness distribution. The problems’ classification is often based on properties given by human definitions: this entails a human bias, while the analysis of the difficulty should be less dependent from the choices of the designer. Therefore, by sampling over a set of different representation, it is possible to isolate the problem’s properties from a particular representation. Jones and Forrest (1995) propose instead to use a *fitness distance correlation*, i.e., the relationship between the fitness function and an ideal goal, but this can lead to some mispredictions as the correlation measure may be too simplistic. Guo and Hsu (2003) advance three key aspects to iden-

tify a hard problem. The first one is when the defined fitness lacks any information about the goal (e.g., needle-in-the-haystack problem), and therefore an objective-based approach cannot be useful. The second reason is when the search algorithm does not exploit helpful information given by the fitness function intentionally. Even if this might be considered a non-optimal use of the algorithm (it would perform worse than random search on a set of GA-easy problems), it can be regarded as a solution for deceptive problems, the third cause of problem hardness. In this case, the fitness function returns conflicting information regarding the goal and ignoring the objective could be beneficial. In conclusion, as noted by Lehman and Stanley (2011a), it seems that 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.

2.3.2 Ignoring the Objective: Divergent Search

Deception actively leads search away from the global optimum, often by converging prematurely to local optima in the search space. Numerous approaches have been proposed to discourage this behavior, as surveyed by Lehman et al. (2013). Many diversity maintenance techniques, such as speciation (Stanley and Miikkulainen, 2002) and niching (Wessing et al., 2013) enforce local competition among similar solutions. Similarity can be measured on the genotypical level (Goldberg et al., 1987), on the fitness scores (Hu et al., 2005), or on the age of the individuals (Hornby, 2006). An alternative way of exploring the search is coevolution, where the calculation of fitness is dependent on the current population (Angeline and Pollack, 1994); competition between individuals in the same population ideally leads to an arms race towards better solutions and finds a better gradient for search. However, coevolution runs the risk of causing mediocre stalemates where all competitors perform poorly and cannot improve, or that one competitor is so much better than the others that no gradient can be found (Ficici and Pollack, 1998). An interesting alternative approach is proposed in Hutter and Legg (2006), where the selection pressure is distributed uniformly across sparsely populated fitness regions. Such a method has the advantage not to bias the search towards highly fitted areas of the search space, and therefore it can counter the effect of deception. Another possible way to counter deceptive landscapes is to use hand-crafted (or hand-shaped) objective function. The idea, named *incremental learning* (Elman, 1993), is to design multiple objective functions from simpler to harder. The algorithm will learn the task from the easier objectives, and then it will progressively learn more complex and interesting tasks. However, this approach requires a strong human involvement and an in-depth knowledge of the task to solve, but, given a black-box scenario, this is not always possible. Techniques from multi-objective evolutionary algorithms can, at least in theory, explore the search space more effectively by evaluating individuals in more than one measure of quality (Knowles et al., 2001) and thus avoid local optima by attempting to improve other objectives; however, multi-objective optimization cannot guarantee to circumvent deception (Deb, 1999). Moreover, optimizing for multi-objectives objectives does not ensure a more manageable problem (Brockhoff et al., 2007), as the number of non-dominated solutions can grow with the number of the objectives (Purshouse and Fleming, 2007). Further, multi-modal function optimization can be employed (Goldberg et al., 1987; Mahfoud, 1995). For instance, the niching algorithm NEA2 by Preuss (2015) uses nearest-better clustering to spread several parallel populations over the multiple local optima that are present in the fitness landscape. However, in Lehman et al. (2013), it is argued that in perversely deceptive problems, such as maze navigation and biped robot locomotion, genotypic diversity is

not sufficient, as all the possible “right” innovation steps towards the global optimum are punished by the optimization process.

Divergent search tackles the problem directly by rewarding diversity at the phenotypical level, and a shift of paradigm is proposed where rewarding *behavioral* diversity becomes the predominant driver of evolution. It is argued that divergence, therefore, can tackle the problem of a deceptive fitness landscape or a deceptive fitness characterization by awarding diverse behaviors (Lehman and Stanley, 2011a) which may eventually lead to optimal results. As a popular example of divergent search methods, *novelty search* by Lehman and Stanley (2011a) is inspired by open-ended evolution and rewards behaviors that have not been seen previously during search. As well as novelty search, surprise-search approaches can be framed within this paradigm (Yannakakis and Liapis, 2016; Gravina et al., 2016b, 2017c). In particular, we can see surprise as a proxy measure to guide the search towards the stepping stones needed to find the global solution of a given (deceptive) problem.

Novelty Search

Novelty search (Lehman and Stanley, 2011a) differs from previous approaches at handling deceptive problems as it explicitly ignores the objective of the problem it attempts to solve. While traditional convergent approaches provide control mechanisms, modifiers or alternate objectives which complement the gradient search towards a better solution, novelty search 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 different than a random walk, however, as it explicitly provides higher rewards to more diverse solutions and also because it maintains a memory of the areas of the search space that it has previously explored. The latter is accomplished via a *novelty archive* of past novel individuals, with individuals with a high novelty score being added continuously to this archive. Each individual’s novelty score is the average distance from a number of closest neighbors in the behavioral space; neighbors can be members of the current population or the novelty archive (see Fig. 2.5). The distance measure is problem-dependent and can also bias the search (Lehman and Stanley, 2011a) and thus affect the performance and behavior of the novelty search algorithm: examples include the agents’ final positions in a two-dimensional maze solving task, the position of a robot’s center of mass (Lehman and Stanley, 2011a), properties of images such as brightness and symmetry (Lehman and Stanley, 2012), machine-learned encodings (Liapis et al., 2013), or the amount of collected reward (Risi et al., 2009, 2010).

The novelty score is computed through Eq. (2.3), i.e., the average distance with the n_{NS} closest individuals in the current population or an archive of novel solutions (see Fig. 2.5). In every generation, individuals with a novelty score above a fluctuating threshold are added to a novelty archive which persists throughout the evolutionary run. The distance d_n , which is used to assess the novelty score as well as what constitutes an individual’s closest neighbors, is based on the difference of behaviors (rather than genotypes) between individuals. This allows novelty search to explore a diverse set of behaviors without explicitly favoring behaviors closer to the desired behavior, i.e., the solution of a problem.

$$n(i) = \frac{1}{n_{NS}} \sum_{j=0}^{n_{NS}} d_{NS}(i, \mu_j), \quad (2.3)$$

where d_{NS} is the behavioral distance between two individuals and depends on the domain

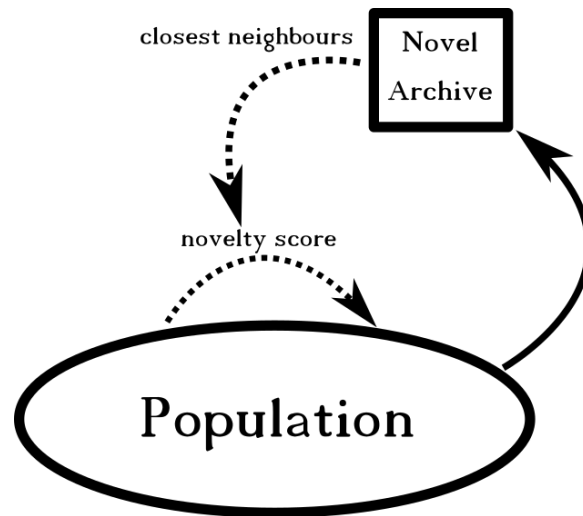


Figure 2.5: **Novelty Search**. A high-level diagram of the Novelty Search algorithm. The novelty score is computed for every individual in the population by averaging the distance from the closest neighbors collected from the current population and an archive. Image inspired by Liapis (2014) and reproduced for the purposes of the thesis.

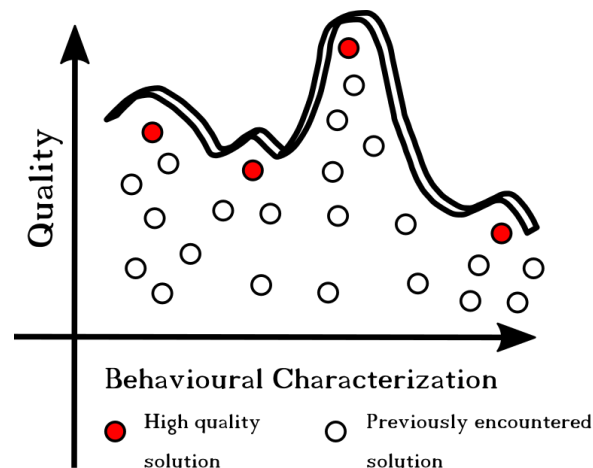


Figure 2.6: **Quality Diversity**. A high-level diagram of the Quality Diversity paradigm. Given a behaviour characterization (or space descriptor), here depicted as a one-dimensional space, and the desired quality, the goal is to find a set of diverse and high-performing solutions (red points). Image inspired by Cully and Demiris (2018) and reproduced for the purposes of the thesis.

under consideration (e.g., the Euclidean distance between two robots’ final positions in a maze navigation task), μ_j is the j -th nearest neighbor and $n(i)$ is the novelty score for the individual i . The current population and the novelty archive are used to find the nearest neighbors.

Novelty search has demonstrated its effectiveness in domains such as maze navigation and robot control (Lehman and Stanley, 2011a; Risi and Stanley, 2013), image generation (Lehman and Stanley, 2012), data clustering (Naredo and Trujillo, 2013), symbolic regression (Martínez et al., 2013) and game content generation (Liapis et al., 2013, 2015).

2.3.3 Beyond Divergence: Diverse and Good solutions

While diversity alone can be beneficial to discover the optimal solution of a deceptive problem, for particular search spaces this is not enough. When the search space is boundless or when the diversity measure is entirely uncorrelated to desired behaviors, some push towards quality is needed. Several solutions have been proposed to solve this issue, such as combining novelty with objectives (Mouret, 2011; Gomes et al., 2015), the minimal criteria novelty search algorithm by Lehman and Stanley (2010), or the constrained novelty search algorithm by Liapis et al. (2015). However, a more aligned combination of divergence and convergence might enable evolutionary search to discover high-performing and diverse solutions in the same run. In fact, the solutions mentioned above try to combine novelty with a *global* competition for the objective, which may contrast with the inspiration of divergent search and make the convergence preminent again.

Nature can be source of inspiration to solve this problem. Natural evolution has discovered an impressive number of diverse solutions (i.e., organisms) that can adapt to different conditions and constraints. If we take for example the problem of moving, nature has evolved different ways to ambulate: crawling, swimming, walking, etc. Moreover, an organism can be composed of different legs. It becomes apparent that each configuration of the above can lead to an effective yet different solution to movement. An interesting parallelism can be traced back to the rise of dinosaurs. Recent theories (Langer et al., 2017) have highlighted the fact at the beginning of the Triassic period (at the early stage of their evolutionary history), dinosaurs were not as widespread as believed up to now, but they were overtaken by other species that were larger and more diversified. The dinosaurs continued to live as a marginal species till the end of the Triassic period when a random event caused the extinction of their direct competitors. Starting from the the Jurassic period, dinosaurs began to spread and outperform any other species in terms of agility, dimensions and strength. Therefore it seems that protecting promising niches in the search space can lead to advantages regarding global performance, as they can save interesting and potential characteristics that can enable further *evolvability* (Lehman and Stanley, 2011c).

Such an approach can lead to several advantages if transferred to an optimization scenario. Given a set of solutions for a problem, an algorithm can use a *repertoire* of alternative solutions in case the selected solution does not perform as expected, i.e., in case the simulated results do not transfer well to reality. This problem, known as *reality gap* (Koos et al., 2013), can be solved efficiently by keeping several good solutions in an archive and employ backup solutions when needed (Cully et al., 2015). In evolutionary robotics, an archive of diverse and good solutions grants the ability to evolve controllers able to solve multiple tasks, which is a more efficient solution compared to evolve a separate controller for each task. A further advantage is that diversity and quality can influence each other synergistically towards an even better solution; rewarding diversity might help to find the necessary stepping stones towards higher-performing areas of the search space, as they circumvent potential deceptive traps of the fitness landscape.

Inspired by these advantageous properties, Pugh et al. (2016) propose the quality diversity challenge. This family of algorithms search 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 (Ritchie, 2007). Such evolutionary algorithms have been named *quality diversity* (QD) algorithms (Pugh et al., 2015, 2016) and aim to find a maximally diverse population of high-performing individuals. Examples of such algorithms include novelty search with local competition by

Lehman and Stanley (2011b) and MAP-Elites by Mouret and Clune (2015); Cully et al. (2015) 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* by Liapis et al. (2015).

It is worth mentioning that this new paradigm stresses the importance of the diversity of the solutions, while the performance becomes of secondary importance, in contrast with the traditional formulation of EC (Pugh et al., 2016). While many solutions have been proposed in the literature to search for good and diverse solutions, they usually focus first on the performances of the solutions, and not on their diversity. For instance, in multi-modal function optimization (Goldberg et al., 1987; Mahfoud, 1995; Preuss, 2015), the objective is to find the local optima of a function. MMFO accomplishes this task by spreading the solutions over the search space employing, e.g., clustering, but usually, the diversity is maintained in the genotypical space. This has the disadvantage that possible good solutions cannot be reached if the fitness function is mono-modal. Behavioural diversity is in general different from the genotypical diversity, especially when there is an indirect encoding between the genotype and the phenotype Stanley and Miikkulainen (2002); a possible phenomenon, for instance, it is the *genotypic aliasing*, which can make different genotypes acting in the same way in the phenotypic space.

Another possible approach to the quality diversity challenge are the multi-objective evolutionary algorithms (Deb, 2001) (explained earlier in Section 2.2.1). MOEAs try to optimize more than one objective, and they return a selection of high-performing solutions called *Pareto* front. However, MOEAs are *intrinsically convergent* (Cully and Demiris, 2018), as their primary use is for optimization of multiple objectives, while divergence is used only as an aid for the exploration of the fronts (e.g., crowding distance). Admittedly, the quality diversity paradigm has been inspired by the idea of solving single-objective optimization problems via multi-objective algorithms. Such approaches have been successfully applied to different problems in the literature. For instance, Gong et al. (2015) proposes using MOEAs to optimize two conflicting objectives, the reconstruction error and the sparsity of deep artificial neural networks. In Li et al. (2016), *multi-objective self-paced learning* decomposes a hard problem into simpler problems which can be optimized in a more accessible way. In Qian et al. (2015a), a multi-objective approach is proposed for ensemble learning to obtain the best performance with the fewest learners. In Qian et al. (2015b), feature selection is performed through evolutionary Pareto optimization to select the best possible subset of features for a deep neural network (feature learning). Finally, in Qian et al. (2017) MOEAs are used for influence maximization in a constrained scenario.

A shift towards divergent algorithms is necessary. The argument is similar to the one used previously for divergent search: given a definition of objective function, a bias is introduced in the optimization process that leads to the problems described in Section 2.3. Searching for diversity and then archiving the best solutions helps the search process to be independent of poorly designed fitness functions. However, the definition of the diversity measure can influence the performance of a QD algorithm as well. As shown in (Pugh et al., 2016; Preuss et al., 2014), depending on the diversity measure employed, specific algorithms can be more or less performant, based on the *alignment* between the selected behaviour characterization and the performance measure. While the diversity measure affects the performance of the QD algorithm, another critical aspect to take into consideration is *how* this diversity information is exploited. *Novelty with local competition* by Lehman and Stanley (2011b) uses the diversity measures defined in Eq. (2.3) and *MAP-Elites* by Mouret and Clune (2015) subdivides the behavioral space in bins of predefined size. In this

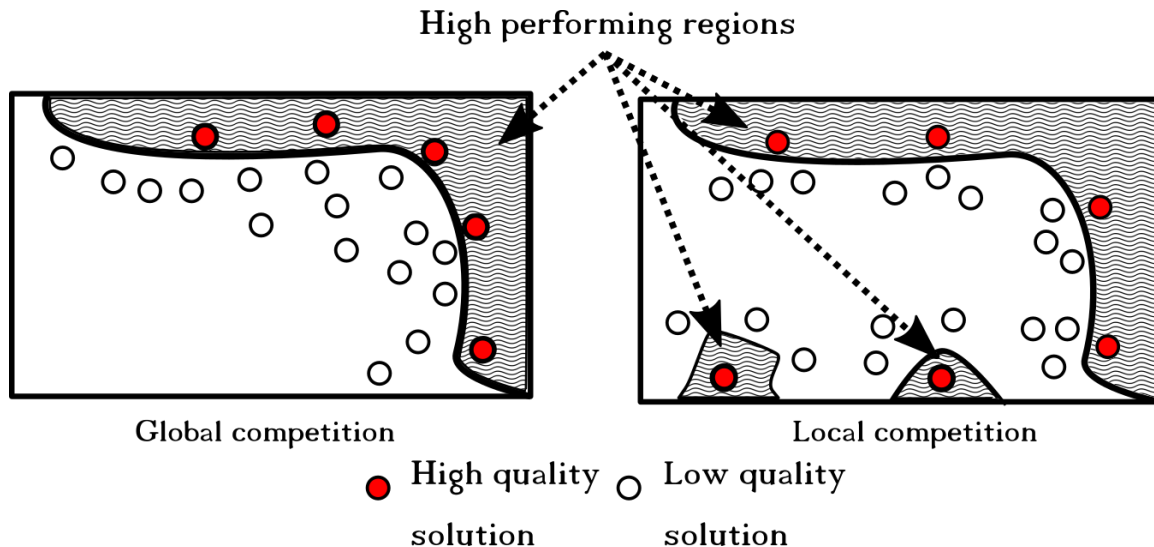


Figure 2.7: **Global vs. Local Competition.** A visualization of the resulting exploration in the search space obtained with global competition and local competition. Local competition enables to exploit the explorative capabilities of a divergent algorithm and at the same time to *illuminate* interesting (i.e., high quality) regions of the search space, that would be instead out-shadowed by the global competition variant.

thesis, we argue that a QD algorithm will discover more highly-fit solutions if we extend the divergence pressure across multiple and orthogonal dimensions beyond novelty—such as surprise. Following this view, in Chapter 3 we introduce three algorithms that explore how both surprise and a combination of novelty and surprise might help to cover the solution space more advantageously.

Novelty Search with Local Competition

Considered to be the first QD algorithm, novelty search with local competition (Lehman and Stanley, 2011b) is an algorithm that combines the divergence of novelty search with the localized convergence obtained through a local competition. In NS-LC, a multi-objective algorithm, NSGA-II by Deb et al. (2002), searches for non-dominated solutions across two dimensions: novelty and local competition. Novelty attempts to maximize a novelty score computed in Eq. (2.3), i.e., the average distance of the individual’s behavior with the behavior of the closest neighbors in the current generation or the novelty archive. Local competition is also calculated based on the closest individuals in the current generation and the novelty archive (see Algorithm 1).

Local competition is introduced as a way to protect from selection the sub-performing niches present in the population. If the evolutionary pressure is based only on global competition and novelty, this will push the population towards one non-dominated front and interesting niches of the search space will be discarded (Lehman and Stanley, 2011b; Mouret, 2011). However, in a quality diversity setting, we are interested in those areas of the solution space that are diverse and sufficiently good. Following the idea of protecting potentially attractive solutions, we limit the competition for the objective only in the behavioural neighborhood of the solutions (see Fig. 2.7).

Algorithm 1 outlines the key steps involved in the local competition computation. The

Algorithm 1: Calculation of Local Competition in a population.

```

input : population  $Pop$ , novelty archive  $A$ , local competition  $n_{LC}$ , objective
         function  $f$ , behavioral distance  $d$ 
output: population  $Pop$ 
1 Initialize  $T \leftarrow \emptyset$ ;           // temporary vector of individuals  $T$ 
2  $T \leftarrow Pop \cup A$ ;
3 foreach  $i \in Pop$  do
4   Sort  $T$  ascending based on distance  $d$  to  $i$ ;
5    $lc(i) \leftarrow n_{LC}$ ;
6   for  $j : 0 \rightarrow n_{LC}$  do
7     if  $f(i) < f(T_j)$  then
8        $lc(i) \leftarrow lc(i) - 1$ ;
9 return population  $Pop$ 

```

reward for local competition $lc(i)$ of the individual i is proportional to the number of solutions outperformed in terms of the objective function f (see line 11 of Algorithm 1). This creates a pressure towards those solutions that are good within their (behavioral) niche, even if they globally underperform compared to the general population Pop . The blend between novelty search and local competition allows NS-LC to pursue and optimize many different behaviors in the hope that one of these directions would eventually lead to globally optimal solutions. NS-LC has shown performance advantages over novelty search in the domains of maze navigation (Pugh et al., 2015, 2016) and robot evolution (Lehman and Stanley, 2011b; Cully and Demiris, 2018).

2.3.4 Curiosity-based Reinforcement Learning

While divergent search and quality diversity are gaining momentum in the field of EC, the idea of using pure exploration as the core objective has been transferred in other fields as well, such as in Reinforcement Learning (RL) (Sutton and Barto, 2018). Reinforcement Learning is a machine learning approach, where an RL agent receives a training signal from the environment by interacting with it. In RL, the objective is to maximize the expected return of policy, i.e., maximise the sums of rewards (Sutton and Barto, 2018).

Like evolutionary computation, rewarding the main objective of the problem works well in a number of cases, especially when the rewards distribution is dense and it is statistically easy to find a random sequence of actions that lead to high rewards. However, there are several cases where the reward scheme is sparse and finding a valid sequence of actions is hard if not highly improbable (Mnih et al., 2015; Bellemare et al., 2016). To address this challenge, exploration of the environment becomes a fundamental key to solve hard problems where extrinsic rewards are sparse or completely missing. A vast number of solutions have been proposed, such as using ϵ -greedy, Boltzmann exploration (Mnih et al., 2015) or Gaussian Noise (Schulman et al., 2015). However, these heuristics rely on random walks, which can be either inefficient or extremely computationally expensive for problems with large state spaces.

Curiosity and Intrinsic motivation

In *curiosity-based* learning, an RL agent explores novel observation led by intrinsic motivation rewards (Kaplan and Oudeyer, 2007). The *reduction of cognitive dissonance* states that to motivate agents intrinsically they should be incentivised to learn cognitive models that successfully predict novel sensory input (Festinger, 1962). We can formalize this theory in the context of RL by rewarding agents that observe novel, surprising or curious events based on their internal model of the environment (Schmidhuber, 2010). For instance, it is possible to reward the discrepancies between what is usually perceived and what is currently perceived, i.e., the prediction error of the learning model (Gordon and Ahissar, 2012).

Inspired by intrinsic motivation theories, a number of different solutions have been proposed in recent years. *Variational Information Maximizing Exploration* (Houthoofd et al., 2016) is an exploration strategy that maximises the information gain of the agent’s model of the environment. Based on the principle of curiosity, agents are encouraged to visit those states that cause a substantial update in their dynamic model distribution. Building on the concept of information gain, Bellemare et al. (2016) propose to use *prediction gain*, which simulates in the continuous domain the concept of count-based exploration in a tabular setting. Another count-based generalization has been proposed by Tang et al. (2017). The authors propose to discretize the continuous state space in a hash-table, and then compute an exploration bonus based on the visitations count. In Fu et al. (2017), the authors propose to train a classifier to distinguish new states from others seen previously, and compute a novelty score based on how easy is to classify a state as novel. A self-supervised inverse model is employed in (Pathak et al., 2017), where the curiosity score is computed as the error between the prediction made using the agent’s next action and the resulting state. In Burda et al. (2018) the authors introduce an easy and computationally preferred solution for sparse rewards problems, where an exploration bonus aids a deep reinforcement learning algorithm to solve difficult problems. This exploration bonus is proportional to the error of a random neural network trained to predict the features of the observations, in order to reward the *novelty* of the observed states. Justesen and Risi (2018) propose to reward the *rarity* of the event experienced by the agent. In particular, the events that are experienced more often are considered not interesting, while higher rewards are assigned to rare events. The *Rarity of the Events* method pushes the agent to progressively explore rare behaviours starting from the more common ones, with the hope that it will learn more complex and potentially exciting tasks. Another solution comes from Shyam et al. (2018), named *Model-Based Active Exploration*. In that work, the authors propose to actively search for novelty, by computing several surrogate models and measuring the amount of error between the models. The agent’s reward is proportional to this error which accounts for the most interesting future possible states. In Savinov et al. (2018) it is proposed to compute a *curiosity* metric by using an episodic memory. The approach uses a neural network to approximate the number of environment steps to reach two observations. Given this reachability metric, novelty is computed as the number of steps it takes to reach a selected observation starting from the ones in memory.

Artificial curiosity and intrinsic motivation differ from surprise-based evolutionary computation as the latter is based on evolutionary divergent search and motivated by open-ended evolution, similarly to novelty search. Specifically, as we will describe in Chapter 3, surprise search approaches does not keep a persistent world model as Schmidhuber (2010) does; instead it focuses on the current trajectory of search using the latest points of the

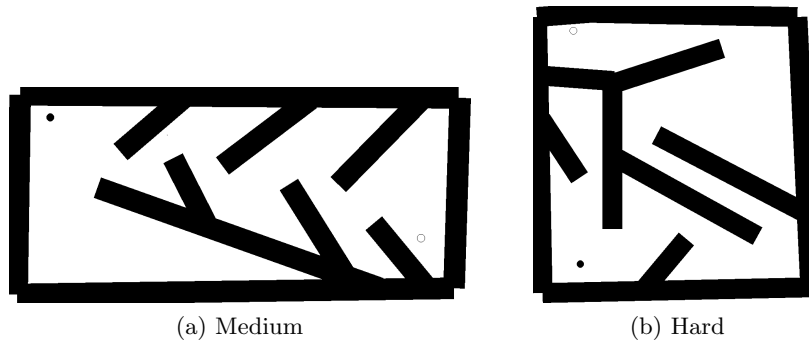


Figure 2.8: **Maze Navigation.** The maze testbeds that appear in (Lehman and Stanley, 2011a) (Fig. 2.8a and 2.8b). The filled circle is the robot’s starting position and the empty circle is the goal.

search space it has explored (and ordering them temporally). Additionally, it rewards deviations from expected behaviors 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.

2.4 Domains

This section describes the domains used in this dissertation to validate the proposed algorithms. Section 2.4.1 offers a detailed description of the first domain used to validate the introduced algorithm: the maze navigation task. Maze navigation is a well-known task to test divergent and quality diversity approaches, thanks to the relatively cheap computational resources required to run the experiments and the facility to generate deceptive environment procedurally. The second domain involves the evolution of soft robot morphologies: evolving virtual creatures is a more complex task compared to the maze navigation domain and it is an ideal testbed to assess the potential creativity of evolutionary algorithms.

2.4.1 Maze Navigation

One of the most popular testbeds for divergent search and quality diversity algorithms is the maze navigation problem. First proposed by Lehman and Stanley (2011a), the maze navigation problem has several properties that make it suitable for testing divergent and quality diversity algorithms and, hence, the algorithms proposed in this dissertation.

The problem of maze navigation can be simply formulated as follows: a robot starting at a specific position in the maze must reach the goal position in a maze using local (incomplete) information. The robot is equipped with six range finder sensors which indicate the distance from the closest wall and four pie-slice sensors broadly indicating the direction of the goal. During simulation, the sensors’ data is provided as input to an artificial neural network (ANN) which controls the movement of the robot, i.e., its velocity and turning angle as two outputs. As per the problem formulation, the *de facto* objective is to reach the goal position, and the most intuitive objective/fitness function is to select individuals based on their Euclidean distance from the goal—ignoring the presence of walls. The fact that the maze topology is not known in advance (local information) results in a deceptive fitness

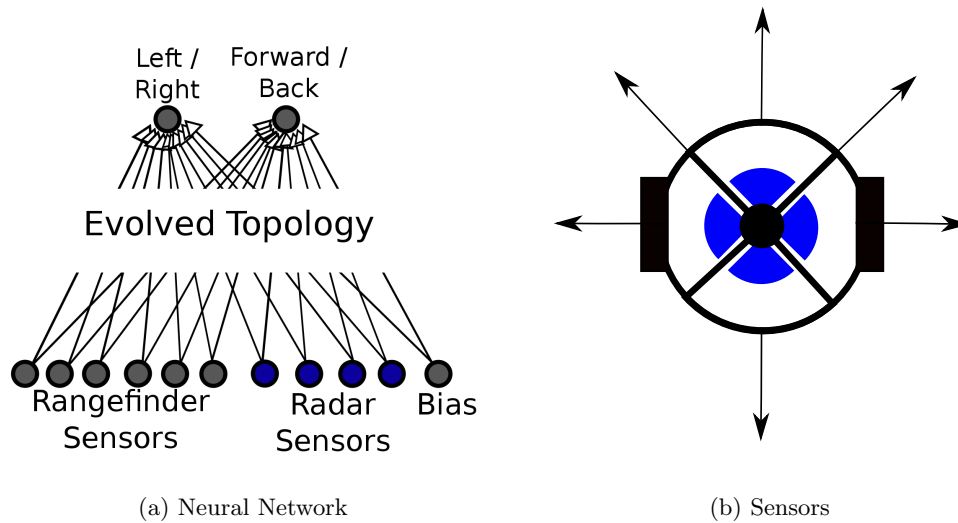


Figure 2.9: **Robot controller for the maze navigation task.** Fig. 2.9a shows the network’s inputs and outputs. Fig. 2.9b shows the layout of the sensors: the six black arrows are rangefinder sensors, and the four blue pie-slice sensors act as a compass towards the goal.

landscape as the robot may end up in a dead-end which is locally optimal but is unable to bring the robot any closer to the goal. In order to find the goal, in most cases the robot must go through areas of lower fitness before the goal becomes accessible. Deceptive mazes are easy to identify visually, as they feature dead ends along the direct line between starting and goal position. Beyond a visually interpretable search space, which largely coincides with the physical space of the maze, an additional property of this testbed is the relatively lightweight simulations it affords; this allows extensive tests to be run, featuring evolutionary runs with large population sizes and multiple re-runs. Indicatively, this dissertation performs 50 independent evolutionary runs per maze (or more, considering a sensitivity analysis), and tests more than 100 mazes in this fashion. Such a computational burden is prohibiting in more complex simulations such as evolving robot morphologies (Lehman and Stanley, 2011b) or generating game content and testing it in games (Cardamone et al., 2011). One of the core contribution of this dissertation is the introduction of two new authored deceptive mazes and, most importantly, a new methodology to generate procedurally extremely difficult mazes. We evaluate and compare the introduced algorithms—and all other baseline algorithms examined—comprehensively across 4 authored mazes and 120 procedurally generated mazes of varying complexity and degrees of deceptiveness. This analysis offers a broad assessment of the capacity, efficiency and robustness of surprise-based approaches in this domain. While the generality and extent to which we evaluate the algorithms are not an innovation per se, they consist a decisive step towards establishing a methodology for evaluating divergent and QD algorithms based on procedurally generated content. The large number of runs and the extensive comparisons with baselines and variations of the algorithms proposed similarly enhance the validity and generality of the findings of this thesis.

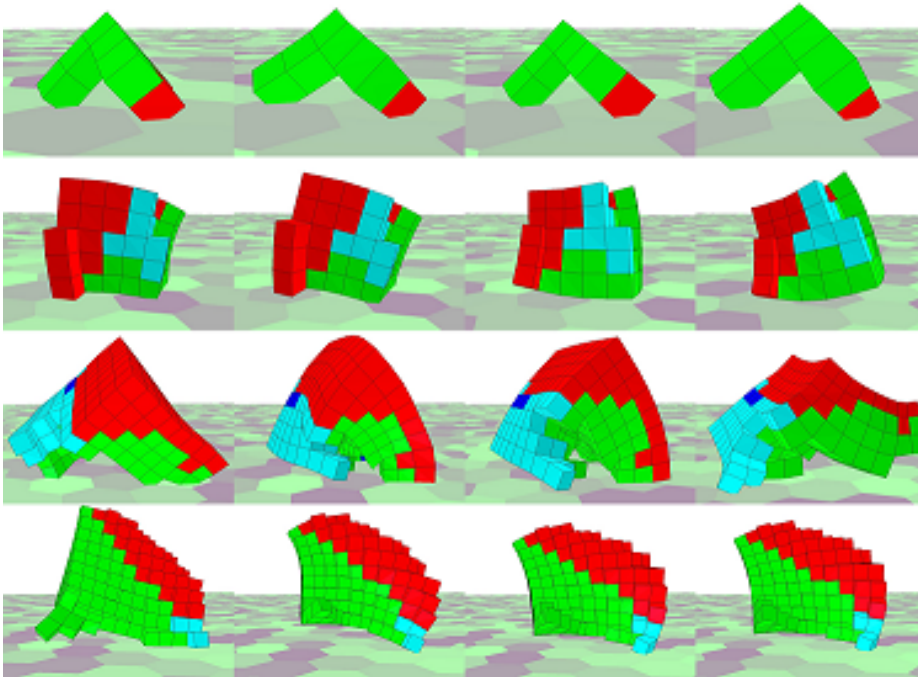


Figure 2.10: **Soft Robot Evolution.** Sample of soft robots evolved with a CPPN representation.

2.4.2 Soft Robots

Evolving virtual creatures is a popular testbed used to assess the potential creativity of evolutionary algorithms, and in the last years several environments have been proposed, ranging from evolving rigid bodies (Sims, 1994; Lehman and Stanley, 2011b) to evolving soft body morphologies (Hiller and Lipson, 2012). While it has been proven that evolving interesting and efficient artificial creatures is feasible, these works still fall short when compared to the complexity found in nature. Cheney et al. (2013) propose to achieve this ambitious objective by evolving the robots’ morphologies by means of different materials, creating “soft” robots composed of voxels with different properties.

The domain of evolutionary robotics traditionally focuses on artificially evolving the structures of virtual creatures. The robots’ numerous degrees of freedom and the difficulty of the task makes EC ideal for tackling this problem. Previous work has focused on evolving rigid bodies (Sims, 1994; Lehman and Stanley, 2011b), as they are simpler and less computationally expensive to simulate. On the one hand, the few degrees of freedom can limit the dexterity of rigid bodies, unless an excessive number of joints is used. On the other hand, soft bodies have a distributed deformation that permits theoretically infinite degrees of freedom, allowing these “soft” robots to reach any point in the space with an infinite number of configurations. Moreover, soft bodies can conform to obstacles, as they generate little resistance to external forces (Trivedi et al., 2008).

Relatively few attempts have been made to evolve soft robots, as this problem comes with a high computational cost in simulating these materials and a large parameter space, especially if different materials are applied. (Hiller and Lipson, 2012) introduced the use of a soft-voxel simulator (VoxCad) to simulate the statics and the dynamics of soft bodies within reasonable computational budgets. The lattice of the soft robot has a predefined

resolution, and multiple materials are chosen as building blocks to compose the robot, both active (as they can contract and expand following an external signal) and passive (e.g., not actuated). Within the domain of evolving virtual creature morphologies, it has been shown that direct encodings tend to lead to poorly structured and dysfunctional robot architectures (Cheney et al., 2013). For that purpose, several indirect encodings have been proposed, including L-systems by Hornby et al. (2001), hierarchical nested graphs by Sims (1994), and gene regulatory networks by Bongard and Pfeifer (2003). Cheney et al. (2013) propose to evolve soft morphologies by evolving Compositional Pattern Producing Networks (Stanley, 2007), given their high evolvability and expressive range capacities. CPPNs evolve using the neuroevolution of augmenting topologies algorithm (Stanley and Miikkulainen, 2002). As shown in (Cheney et al., 2013), the CPPN representation allows soft robots to exhibit several locomotion strategies and morphologies (Fig. 2.10).

Compared to the maze navigation domain, evolving soft robot morphologies is (a) more challenging and (b) requires a more complex behaviour characterization (e.g., the movement trail of the evolved robot). Given these properties, we argue that this is an ideal testbed to investigate the generality of the results obtained in maze navigation, as it is a complementary deceptive domain. Towards that end, we test the surprise-based approaches thoroughly across eight different lattice resolutions and we perform an in-depth analysis of the structural and behavioural characteristics of the individuals evolved by the surprise search approaches.

2.5 Summary

This chapter has presented the areas of research involved in this dissertation and the key algorithms that have been used for comparative purposes or have inspired this work. Two interwoven areas are necessary to frame this dissertation: computational creativity and evolutionary computation. We offered an extensive literature review on computational creativity, which focused on three central concepts that are considered fundamental in order for a machine to generate creative outcomes through a computational process: novelty, surprise and value. A description of evolutionary computation followed, where its main components were surveyed. Furthermore, we described in detail two sub-domains important for the purposes of this thesis: multiobjective optimization and neuroevolution.

The main motivation of this work is to solve difficult problems in unconventional ways by means of unexpected solutions. Towards that end, we introduced one of the major source of hardness for evolutionary search, i.e., deceptive landscapes. Furthermore, we described in detail two recent paradigms that address deceptive problems and inspired the algorithms introduced in this work: divergent search and quality diversity. We then introduced two state-of-the-art implementations of divergent search and quality diversity, respectively novelty search and novelty search with local competition.

This chapter ended with a broad description of the domains used in this dissertation. Inspired by divergent search and quality diversity paradigms, the next chapter outlines the family of algorithms (*surprise search*) introduced in this work.

Chapter 3

Surprise Search: the Approach

As already stated in Chapter 1, this dissertation introduces a new general search algorithm inspired by the concept of surprise for unconventional discovery. Surprise search is a new approach that is framed in the divergent search paradigm and built on the principles of evolutionary computation. This chapter formalizes the notion of surprise and builds the algorithm within this formalization. Towards that end, we first give a general definition of surprise and we then describe the surprise search algorithm, by offering an overview of each component of the algorithm (Section 3.1).

Given a formalization of surprise search as an evolutionary algorithm, it is straightforward to expand with other approaches proposed in the literature. As noted in Chapter 2, surprise search is inspired by novelty search, but each algorithm rewards solutions differently: novelty search rewards solutions which exhibit dissimilar behaviour from those in the current and in previous populations, while surprise search rewards solutions which diverge from expected behaviours based on past trends. With the theoretical argument for the orthogonal nature between novelty and surprise, we argue that by coupling the search for unseen (novelty) with the search for unexpected (surprise) solutions we will end up with an algorithm that explores unseen points in the search space and at the same time also rewards deviations from predicted search trends. To test our hypothesis, we introduce five additional surprise-based algorithms. In particular, Section 3.2 describes formally how to combine effectively the two concepts of novelty and surprise in the context of divergent search: a linear combination of novelty and surprise, novelty-surprise search (NSS), and a multi-objective approach, novelty search-surprise search (NS-SS). Section 3.3, instead, describes how surprise can be used as an effective reward in the quality diversity paradigm and it introduces three new quality diversity algorithms. Inspired by novelty search with local competition (see Chapter 2), the new algorithms replace novelty search with surprise search (SS-LC), or combine measures of novelty and surprise linearly (NSS-LC) or as separate objectives (NS-SS-LC).

3.1 Surprise Search

In Chapter 2 we offered an overview of the different formalizations of the notion of surprise. In this section, we take inspiration from the aforementioned perspectives in computational creativity and we attempt to give a general definition of surprise. The primary overarching element of surprise across any of its taxonomies described earlier is the *degree to which observation is expected*. Thus, independently of the various definitions across the disciplines

that study surprise as a phenomenon, we can safely derive a general definition of surprise that satisfies the critical characteristics of that notion. For the purposes of this dissertation, we define surprise as the *deviation from the expected* and we use the notions *surprise* and *unexpectedness* interchangeably due to their highly interwoven nature (Reisenzein, 2000): unexpectedness being the approximate cognitive appraisal cause of surprise. Given this definition, we can introduce the idea of surprise search and propose a general evolutionary algorithm that realizes it. The proposed 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 behavior. The predictive model of expected solutions is based on historical trails of where the search has been and local information about the search space. 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 (Grace et al., 2015). Our hypothesis is that if modeled appropriately, surprise may enhance divergent search and complement or even surpass the performance of traditional forms of divergent search such as novelty. Furthermore, as novelty search (Lehman and Stanley, 2011a) has been defined as a proxy for the intermediate stepping stones towards the objective of the problem, our hypothesis is that surprise stands as an alternative proxy, that can be coupled with novelty and value for divergent search and quality diversity.

3.1.1 The Surprise Search Algorithm

This section discusses the principles of designing a surprise search algorithm for any task or search space. To realize surprise as a search mechanism, an individual should be rewarded when it *deviates from the expected behavior*, 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 (Stanley and Miikkulainen, 2002).

Surprise search can be decomposed into two tasks: *prediction* and *deviation*. At the highest descriptive level, surprise search uses local information from past generations to predict behavior(s) of the population in the current generation; observing the behaviors of each individual in the actual population, it rewards individuals that deviate from predicted behavior(s): this is summarized in Figure 3.1 and Algorithm 2. The following sections will describe the phases of prediction and deviation at high level, and will introduce the parameters which are associated with each phase; in Chapter 4 we will elaborate on their domain-specific implementation.

Prediction

As shown in Fig. 3.1, the predictive model uses local information from previous generations to estimate (in a quantitative way) the expected behavior(s) in the current population. Formally, predicted behaviors (\mathbf{p}) are created via Eq. (3.1), where m is the predictive model that uses a degree of local (or global) behavioral information (expressed by k_{SS}) from h previous generations (behavior vector b_g of size k_{SS} ; refer to line 2 in Algorithm 2). The choice of each of these parameters (m , h , k_{SS}) 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.

$$\mathbf{p} = m(h, k_{SS}) \tag{3.1}$$

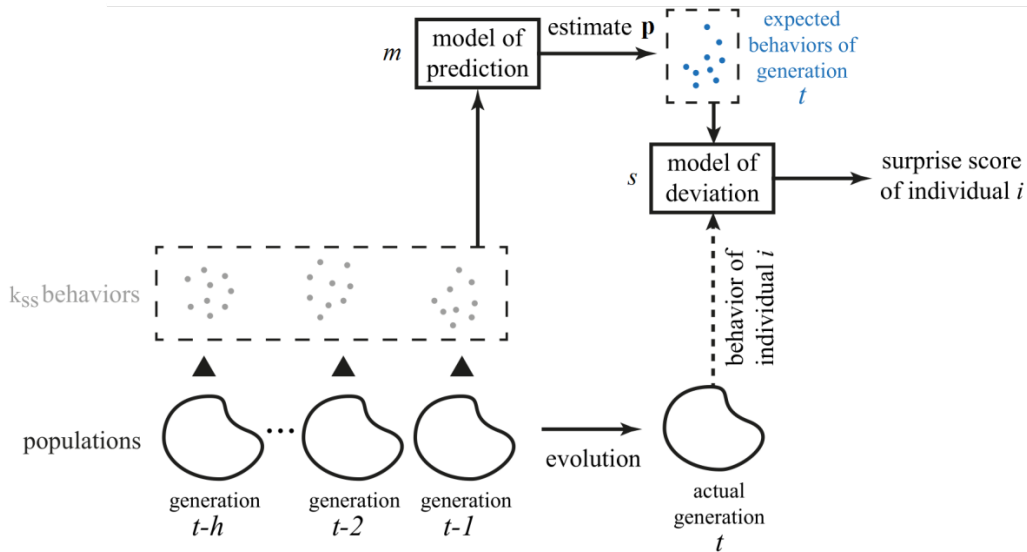


Figure 3.1: **Surprise Search.** High-level overview of the surprise search algorithm when evaluating an individual i in a population at generation t . The h previous generations are considered, with respect to k_{SS} behavioral characteristics per generation, to predict the expected k_{SS} behaviors of generation t . The surprise score of individual i is the deviation of the behavior of i from a subset of these k_{SS} expected behaviors.

How much history of prior behaviors (h) should surprise search consider? To predict behaviors in the current population, the predictive model must consider previous generations. In order to estimate behaviors of a population at generation t , the predictive model must find trends in the populations of generations $t-1, t-2, \dots, t-h$ (line 6 in Algorithm 2). The minimum number of generations to consider to observe an evolutionary trend, therefore, is $h = 2$ (the two prior generations to the one being evaluated). However, behaviors that have performed well in the past could also be included in a *surprise archive*, similar to the novelty archive of novelty search (Lehman and Stanley, 2011a), and subsequently be used to make predictions of future behaviors. Such a surprise archive would serve as a more persistent history ($h > 2$) but considering only the interesting historical behaviors rather than all past behaviors.

How local (k_{SS}) are the behaviors surprise search needs to consider to make a prediction? Surprise search can consider behavioral trends of the entire population when creating a prediction (global information). In that case, $k_{SS} = 1$ and all behaviors are aggregated into a meaningful average metric for each prior generation. The current generation's expected behaviors 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 behavior based on the behaviors of its ancestors in the genotypic sense (parents, grandparents etc.) or behavioral sense (past individuals with the closest behavior). In this case $k_{SS} = P$ where P is the size of the population, and the number of predictions to deviate

from will similarly be P . Therefore, the parameter k_{SS} determines the level of *prediction locality* which can vary from 1 to P ; intermediate values of k_{SS} 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 modeling approach can be used to predict a future behavior, such as a simple linear regression of a number of points in the behavioral 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_{SS} parameters. For instance, a linear model can only be used if $h = 2$, which is the case for all the experiments presented in this thesis.

Deviation

To put pressure on unexpected behaviors, we need an estimate of the deviation of an observed behavior from the expected behavior (if $k_{SS} = 1$) or behaviors. Following the principles of novelty search (Lehman and Stanley, 2011a), this estimate is derived from the *behavior space* as the average distance to the n -nearest expected behaviors (prediction points). The *surprise score* s for an individual i in the population is calculated as:

$$s(i) = \frac{1}{n_{SS}} \sum_{j=0}^n d_{SS}(i, p_{i,j}) \quad (3.2)$$

where d_{SS} is the domain-dependent measure of behavioral difference between an individual and its expected behavior, $p_{i,j}$ is the j -closest prediction point (expected behavior) to individual i and n_{SS} is the number of prediction points considered; n_{SS} is a problem-dependent parameter determined empirically ($n_{SS} \leq k_{SS}$).

Important notes

Surprise search operates similarly to novelty search with respect to evolutionary dynamics. As surprise search considers a set of prior behaviors (expressed by h and k_{SS}) to make predictions of expected behavior, it maintains a temporal window of where search has been. However, surprise search operates differently to novelty search with respect to the goal: surprise maximizes deviation from the expected behaviors whereas novelty moves the search towards new behaviors. This evidently creates a new form of divergent search that considers prior behaviors *indirectly* to make predictions to deviate from.

As surprise search ignores objectives, a concern could be whether it is merely a version of random walk. Surprise search is not a random walk as it explicitly maximizes unexpectedness: surprise search allows for a temporal archive of behaviors that accumulates a record of earlier positions in the behavioral space (similarly to novelty search). Comparative experiments with various random benchmark algorithms in Chapter 4 show that surprise search traverses the search space in a different and far more effective manner.

3.2 Coupling Novelty and Surprise

As mentioned in the introduction, a working hypothesis of this dissertation is that we can equip computers with better search capacities if we couple algorithms with dissimilar properties and ways of operating in the search space. Earlier work (Lehman and Stanley,

Algorithm 2: Calculation of Surprise Score in a population.

input : population Pop , generation g , neighborhood n_{SS} , locality k_{SS} , history length h , prediction model m
output: population Pop

- 1 Compute and store k_{SS} clustered behaviors \mathbf{b}_g for generation g ;
- 2 **if** $g < h$ **then**
- 3 **foreach** $i \in Pop$ **do**
- 4 | $s(i) \leftarrow random()$
- 5 **else**
- 6 Retrieve clustered behaviors from \mathbf{b}_{g-h} to \mathbf{b}_{g-1} ;
- 7 $\mathbf{p} \leftarrow m(h, k_{SS})$; // (Eq. 3.1)
- 8 **foreach** $i \in Pop$ **do**
- 9 | $s(i) \leftarrow 0$;
- 10 $\mathbf{p}_i \leftarrow$ Sort \mathbf{p} ascending based on distance d_{ss} to i ;
- 11 **for** $j : 0 \rightarrow n_{SS}$ **do**
- 12 | $s(i) \leftarrow s(i) + d_{ss}(i, p_{i,j})$; // (Eq. 3.2)
- 13 | $s(i) \leftarrow s(i)/n_{SS}$;
- 14 **return** population Pop ;

2011a) have revealed that divergent search algorithms such as novelty have shown promise in highly deceptive problems; however, recent findings have demonstrated that some deceptive problems can be challenging even for divergent search (Cuccu and Gomez, 2011). In this section we propose to combine two divergent algorithms, novelty search and surprise search, and we assume that their complementarity in search—searching for the unseen *and* searching for the unexpected—can improve an algorithm’s performance compared to merely searching for novel or surprising solutions. As the concept of novelty is orthogonal to that of surprise and surprise can be viewed as *temporal novelty* (Yannakakis and Liapis, 2016), combining the two seems to be an approach with great potential.

The following subsections introduce two EC algorithms that couple novelty and surprise as the linear aggregation of the two (*novelty-surprise search*) and by means of multi-objective optimization (*novelty search-surprise search*). In Chapter 5 we will elaborate on their domain-specific implementation.

3.2.1 Novelty-Surprise Search Algorithm

In the field of evolutionary computation several approaches exist for simultaneously optimising several objectives. While recent literature has proposed several multi-objective evolutionary algorithms (Deb, 2001), the simplest solution is to linearly combine the objectives. In this implementation we opt for the latter approach, as adding novelty and surprise is trivial and computationally preferred while it represents an intermediate step towards a multi-objective implementation. The novelty-surprise search algorithm executes both novelty and surprise search and at each generation it rewards an individual by adding its novelty and surprise score in the following fashion:

$$ns(i) = \lambda \cdot n(i) + (1 - \lambda) \cdot s(i), \quad (3.3)$$

where $ns(i)$ is the combined novelty and surprise score of individual i and $\lambda \in [0, 1]$ is a parameter that controls the relative importance of novelty versus surprise.

3.2.2 Novelty Search-Surprise Search Algorithm

As described in Chapter 2, it has been shown that a linear aggregation of multiple objectives is generally not able to find all the possible Pareto solutions, in particular when the Pareto front is not convex (Coello et al., 2007). However, it should be noted that when the objectives are dynamic, e.g., their value depends on the state of the current population, the Pareto front might change over the course of the evolutionary process and its resulting behaviour harder to predict. Indeed, this is the case with novelty and surprise rewards, as they are computed relatively to the state of the population (Lehman and Stanley, 2011a). Therefore, we propose to compare the linear aggregation of novelty and surprise (i.e., novelty-surprise search) against its multi-objective counterpart. To accomplish this, we introduce an alternative approach that fuses novelty and surprise independently. This algorithm, named novelty search-surprise search, employs a multi-objective algorithm to optimize novelty and surprise as separate and independent rewards. In this thesis we use NSGA-II (Deb et al., 2002) to search for non-dominated solutions on the two divergent dimensions of novelty and surprise, but other approaches can be employed (Deb, 2001). In the proposed algorithm, the novelty dimension is computed as in Eq. 2.3, while the surprise dimension is computed following the procedure described earlier in Section 3.1.1.

3.3 Surprise for Quality Diversity

Inspired by previous work on quality diversity (Pugh et al., 2016), we introduce three new algorithms that introduce surprise search as an alternative divergent search mechanism with local competition (Lehman and Stanley, 2011b) for quality diversity (surprise-based QD). These three algorithms are named surprise search with local competition, novelty-surprise search with local competition, and the three-objective novelty search-surprise search-local competition.

Fig. 3.2 offers a high-level overview of the three algorithms used for surprise-based quality diversity. The algorithms initially generate a population (Pop) of N individuals and then initialize the novelty archive (A). We then perform an initial update of the surprise model, and we evaluate each individual in the population according to the selected algorithm (i.e., NSS-LC, SS-LC or NS-SS-LC). Right before entering the main loop of the algorithm, we use non-dominated sorting to divide the population into fronts based on the NSGA-II algorithm (right part of Fig. 3.2). While the termination criterion is not met (e.g., high performance is reached) we run a steady-state EA implementation (see left part of Fig. 3.2): two mating parents are selected to generate a new offspring, which is evaluated based on the algorithm chosen (i.e., either NSS-LC, SS-LC or NS-SS-LC); the worst individual of the population is then replaced by the newly generated offspring if the offspring is more fit. At the end of each steady state step of the algorithm, the non-dominated fronts are updated as in Li et al. (2017). Every N offspring generations, we update the surprise model (see right part of Fig. 3.2) by computing the k_{SS} behavioral clusters and then computing the predictions based on the surprise predictive model m . We then re-evaluate the entire population and recompute the non-domination fronts. The algorithm returns to the main steady state loop until either new N offspring are generated or the termination condition is reached. A key part of the framework is its evaluation step (see middle flow chart in Fig. 3.2): it involves

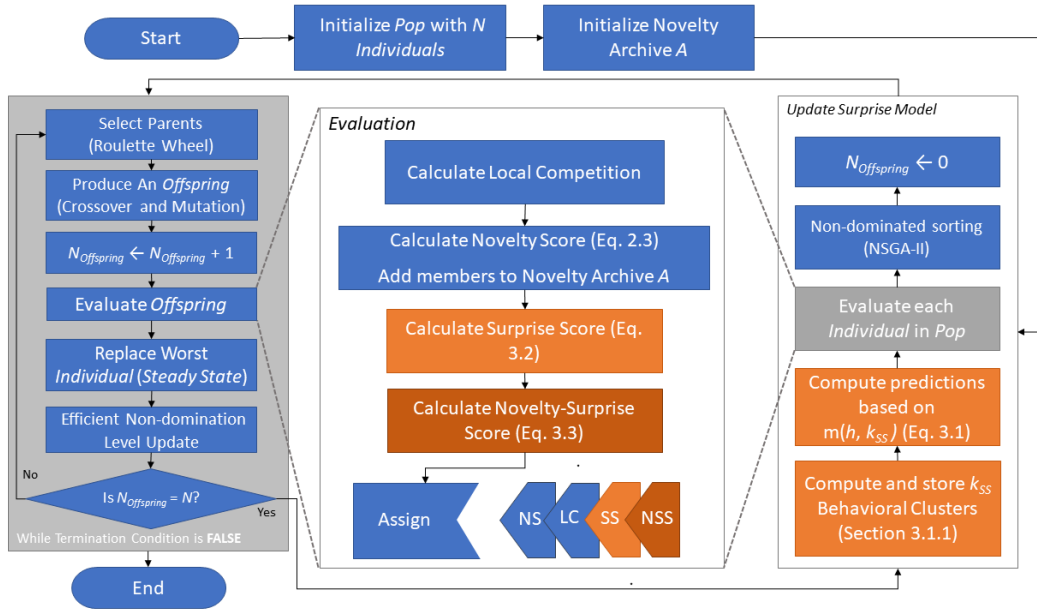


Figure 3.2: **Surprise-based Quality Diversity.** The flow chart illustrates a high level representation of the three introduced QD algorithms: SS-LC, NSS-LC, NS-SS-LC. All algorithms are employed as a steady state EA (left flow chart). The model of surprise is initialized after the generation of the initial population and then updated every N offspring generations (right flow chart). The evaluation of individuals (middle flow chart) goes through the calculation of local competition, novelty, and surprise scores before those are assigned to the corresponding algorithm. Algorithmic loops are depicted as gray boxes. The introduced surprise-based components of the algorithm are depicted in orange: light orange refers to the surprise components and dark orange refers to the novelty-surprise components. For the interested reader, references are made to the equations and sections.

the computation of novelty, surprise and novelty-surprise scores. Based on the algorithm chosen, we assign the selected scores to the individual (i.e., SS-LC, NSS-LC or NS-SS-LC) which will influence the replacement strategy and the non-domination ranking performed by the NSGA-II algorithm. The details of each one of the three algorithms are described in the corresponding subsections below, whereas their domain-specific implementation is further elaborated in Chapter 6.

3.3.1 Surprise Search with Local Competition

As a direct integration of surprise search for quality diversity, the obvious approach is to replace novelty with surprise in the novelty search with local competition (Lehman and Stanley, 2011b) paradigm. In surprise search with local competition, NSGA-II from Deb et al. (2002) searches for non-dominated solutions on the dimensions of local competition and surprise. Local competition is calculated based on the superiority of the individual being evaluated among its closest neighbors. What constitutes a nearby neighbor for local competition is based on behavioral characterization (d_{SS} in this case), rather than a genotypic one. Superiority is established based on a measure of proximity to an ideal solution: the number of neighbors who are worse than the current individual is used as the local competition score to be optimized. Borrowing from local competition as applied to NS-LC, the

closest neighbors are drawn from the current population. Unlike NS-LC, however, SS-LC does not maintain a novelty archive and only considers neighbors in the current population. The surprise dimension uses the surprise score of Eq. (3.2), which assesses how much the behavior of an individual deviates from expected behaviors based on trends in recent generations. As described in Section 3.1.1, the surprise score is calculated based on a two-step process: first, Eq. (3.1) creates predictions based on past generations' dominant behaviors, and then Eq. (3.2) calculates the surprise score based on the distance from predicted behaviors.

3.3.2 Novelty-Surprise Search with Local Competition

Given the orthogonal nature of novelty and surprise, we hypothesize that combining novelty and surprise as different measures of divergence would be valuable for QD. We thus expect that combining both surprise and novelty as measures of divergence with local competition can only improve the performance of the state of the art QD algorithm.

In the novelty-surprise search with local competition algorithm, as we name it, NSGA-II (Deb et al., 2002) searches for non-dominated solutions on the dimensions of local competition and a weighted sum combining novelty and surprise. Local competition measures the number of closest neighbors (in terms of behavior) which underperform compared to the current individual. Unlike SS-LC, local competition considers the novelty archive maintained by the novelty search component and thus the closest neighbors considered for local competition can be from both the current population and the novelty archive. The other dimension targeted by NSGA-II combines the novelty score of Eq. (2.3) and the surprise score of Eq. (3.2) in the weighted sum of Eq. (3.4), where a single parameter (λ) influences both scores.

$$ns(i) = \lambda \cdot n(i) + (1 - \lambda) \cdot s(i), \quad (3.4)$$

where $ns(i)$ is the combined novelty and surprise score of individual i and $\lambda \in [0, 1]$ is a parameter that controls the relative importance of novelty versus surprise, $n(i)$ is the novelty score (Eq. 2.3) and $s(i)$ is the surprise score (Eq. 3.2).

3.3.3 Novelty Search–Surprise Search–Local Competition

As an alternative way to combine novelty search and surprise search in the divergence dimension of a QD algorithm, we can consider them as independent objectives rather than aggregating them as with NSS-LC. The three-objective algorithm novelty search–surprise search–local competition uses NSGA-II (Deb et al., 2002) to search for non-dominated solutions on the three dimensions: the local competition score, the surprise score (as in Eq. 3.2) and the novelty score (as in Eq. 2.3).

3.4 Summary

This chapter has described in detail the algorithm introduced in this thesis, surprise search. We started from a general definition of surprise—deviation from the expected—and we then formalized each component of the algorithm starting from the chosen formalization of surprise. Given the theoretical orthogonality between the concepts of surprise and novelty, this chapter further proposed to expand surprise with novelty search, both for divergent search and quality diversity. For that purpose, we first described formally how to combine the two concepts in two different implementations of divergent search, novelty-surprise

search and novelty search-surprise search. Secondly, we introduced surprise for quality diversity: we presented three new algorithms, namely surprise search with local competition, novelty-surprise search with local competition and novelty search-surprise search with local competition. The next three chapters will be dedicated to the evaluation of the proposed surprise-based algorithms across two domains: maze navigation and soft robot evolution.

Chapter 4

Surprise Search: Experiments

This chapter tests the surprise search algorithm in two different testbeds: maze navigation and soft robot evolution. In particular, in Section 4.1 we test surprise search 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. In Section 4.2, instead, we test surprise search in a more challenging task: soft robot evolution. In particular, we analyze the performance of surprise search with a more complex behavior characterization and we compare the structural diversity of the three evolutionary methods. Finally, Section 4.3 sums up the implications arising from the results obtained by surprise search across the two domains considered.

4.1 Maze Navigation Test Bed

An appropriate domain to test the performance of surprise search is one that yields a deceptive fitness landscape, where search for surprising behaviors can be evaluated against novelty and objective search. Inspired by the comparison of novelty versus objective search in (Lehman and Stanley, 2011a), we consider the two-dimensional maze navigation task as our first test bed in this chapter.

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 (Lehman and Stanley, 2011a), 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. 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. It should be noted that this problem is not *pathfinding*. The main objective of the evolutionary algorithm (i.e., NEAT) is to find an ANN that solves the given maze, and not directly a robot that can find the path from the starting point to the goal. Evolving a controller able to successfully navigate a maze is a challenging problem, as the evolutionary algorithm 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 (Lehman and Stanley, 2011a).

Two properties have made this environment a canonical test for divergent search (Lehman and Stanley, 2008, 2011a; Pugh et al., 2016; Mouret, 2011): the ease of manually designing deceptive mazes and the low computational burden, which enables researchers to run mul-

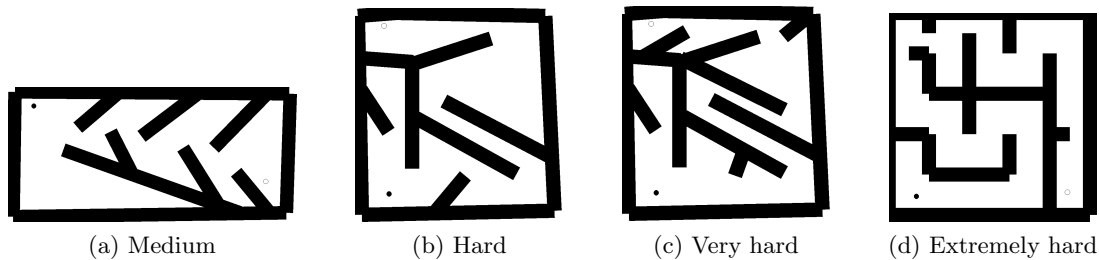


Figure 4.1: **Authored Mazes.** The maze testbeds that appear in (Lehman and Stanley, 2011a) (Fig. 4.1a and 4.1b) and new mazes introduced (Fig. 4.1c and 4.1d respectively). The filled circle is the robot’s starting position and the empty circle is the goal. The maze size is 300×150 units for the medium maze and 200×200 units for the other mazes.

multiple comparative tests among algorithms. Furthermore, the generality of the findings can be tested with automatically generated mazes, as in Section 4.1.4. More information about this domain can be found in Chapter 2.

As with the original implementation of the maze navigation testbed (Lehman and Stanley, 2011a), the ANN of the robot controller is optimized through neuroevolution of augmenting topologies (Stanley and Miikkulainen, 2002), an evolutionary algorithm which has been described more in detail in Chapter 2. Starting from the mazes introduced in (Lehman and Stanley, 2011a), we designed two additional mazes of enhanced complexity and deceptiveness. This section briefly describes the maze navigation problem, the mazes adopted, and the parameters for the experiment.

4.1.1 Domain

We initially test the performance of surprise search on four authored mazes (see Fig. 4.1), two of which (*medium* and *hard*) have been used in (Lehman and Stanley, 2011a). The medium maze (see Figure 4.1a) 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 Figure 4.1b) is more deceptive, due to the dead-end at the leftmost part of the maze; an algorithm must search in less promising (lower-fit) areas of the maze to find the global optimum. For these two mazes we follow the experimental parameters set in (Lehman and Stanley, 2011a) 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 by Lehman and Stanley (2011a), 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. 4.1c) is a modification of the hard maze with more dead ends and winding passages. The extremely hard maze, on the other hand, is a new maze (see Fig. 4.1d) 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.

4.1.2 Algorithms

This section provides details about the general and specific parameters for all the algorithms compared. We primarily test the performance of six 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 (Lehman and Stanley, 2011a), where the maze navigation task and the mazes of Fig. 4.1 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 (Lehman and Stanley, 2011a). However, the number of generations is increased to 1000 for the more deceptive mazes (very hard and extremely hard) to allow us to analyze the algorithms' behavior over a longer evolutionary period. The NEAT algorithm uses speciation and recombination, as described in (Stanley and Miikkulainen, 2002), and the algorithm is steady-state as in (Lehman and Stanley, 2011a). The specific parameters of all compared algorithms are detailed below. A summary of the parameters used can be found in the Appendix A.

Objective search

Objective search uses the agent's proximity to the goal as a measure of its fitness. Following Lehman and Stanley (2011a), 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.

Novelty Search

Novelty search uses the same novelty metric and parameter values as presented in (Lehman and Stanley, 2011a). In particular, the novelty metric is the average distance of the robot from the nearest neighboring 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 (Lehman and Stanley, 2011a).

Sensitivity Analysis: while in (Lehman and Stanley, 2011a) novelty is calculated as the average distance from the 15 nearest neighbors, the introduction of new mazes mandates that the n_{NS} parameter of novelty search is tested empirically. For that purpose we vary n_{NS} from 5 to 30 in increments of 5 across all mazes and select the n_{NS} 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_{NS} 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 4.2 shows the results obtained by this analysis across all mazes.

The best results are indeed obtained with 15 nearest neighbors for the medium and hard maze, as in (Lehman and Stanley, 2011a) (49 and 48 successes, respectively). In the

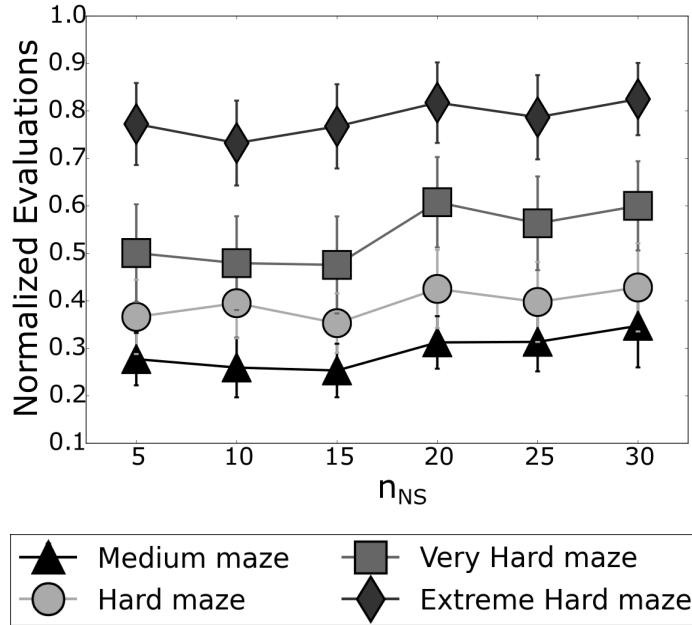


Figure 4.2: **Novelty Search: Sensitivity Analysis.** Selecting the nearest neighbor for novelty search: the figure depicts the average number of evaluations (normalized by the total number of evaluations allocated) obtained out of 50 runs (of 300 generations for the medium and hard maze, of 1000 generations for the very and extremely hard maze) by varying n_{NS} between 5 and 30 for all four authored mazes examined. The error bars represent the 95% confidence interval of the average.

very hard maze there is no difference between 10 and 15 in terms of successes (39) but $n_{NS} = 15$ yields fewer evaluations, while in the extremely hard maze $n_{NS} = 10$ yields fewer evaluations and more successes (24) than any other value tested. In summary, we use $n_{NS} = 15$ for the medium, hard and very hard maze, and $n_{NS} = 10$ for the extremely hard maze.

Surprise search

Surprise search uses the surprise metric of Eq. (3.2) to reward *unexpected behaviors*. As with the other algorithms compared, *behavior* in the maze navigation domain is expressed as the position of the robot at the end of a simulation. The behavioral difference d_{SS} in Eq. (3.2) is the Euclidean distance between the robots' final position and a considered prediction point, p . Following the general formulation of surprise in Chapter 3, the prediction points are a function of a model m that considers k_{SS} local behaviors 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 behavioral space. Thus, only the two previous generations are considered when creating prediction points to deviate from in the current generation (see Fig. 4.3). In the first two generations the algorithm performs mere random search due to a lack of prediction points.

Sensitivity Analysis: To choose appropriate parameters for k_{SS} (information locality) and n_{SS} (number of prediction points) in the prediction and deviation models respectively, a sensitivity analysis is conducted for all mazes. We obtain k_{SS} empirically by varying

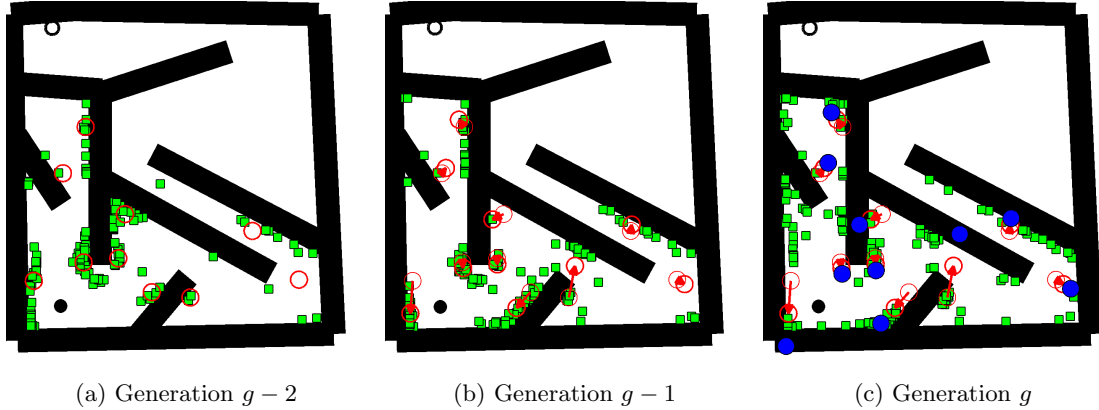


Figure 4.3: **The key phases of the surprise search algorithm as applied to the maze navigation domain.** Surprise search uses a history of two generations ($h = 2$) and 10 behavioral clusters ($k_{SS} = 10$) in this example. Robots’ final positions are depicted as green squares; cluster centroids and prediction points are depicted as empty red and solid blue circles, respectively.

its value between 20 and 240 in increments of 20 for each maze. We also test all k_{SS} for $n_{SS} = 1$ and $n_{SS} = 2$. As in the sensitivity analysis for novelty search we select the k_{SS} and n_{SS} values that yield the highest number of successes in 50 independent runs. If there is more than one k_{SS} , n_{SS} combination that yields the highest number of successes we select the combination that solves the maze in the fewest average evaluations.

Figure 4.4 shows the average number of evaluations for all k_{SS} values tested, for $n_{SS} = 1$ and $n_{SS} = 2$. It is clear that higher k_{SS} values result to fewer evaluations on average. Moreover, it seems that $n_{SS} = 2$ leads to better performance in the two more deceptive mazes. Based on the above selection criteria, we pick $k_{SS} = 200$ and $n_{SS} = 1$ 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 $k_{SS} = 100$ and $n_{SS} = 1$, as it is the most robust (49 success) and fastest (23,214 evaluations on average) among tested values. Following the same procedure $k_{SS} = 200$ and $n_{SS} = 2$ in the very hard maze, and $k_{SS} = 220$ and $n_{SS} = 2$ in the extremely hard maze (see Fig. 4.4).

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 and remains open to future investigations.

Other baseline algorithms

Three more baseline algorithms are included for comparative purposes. *Random search* (RS) is a baseline proposed in (Lehman and Stanley, 2011a) 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_{SS} random prediction points (i.e., $p_{i,j}$ in Eq. 3.2) 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

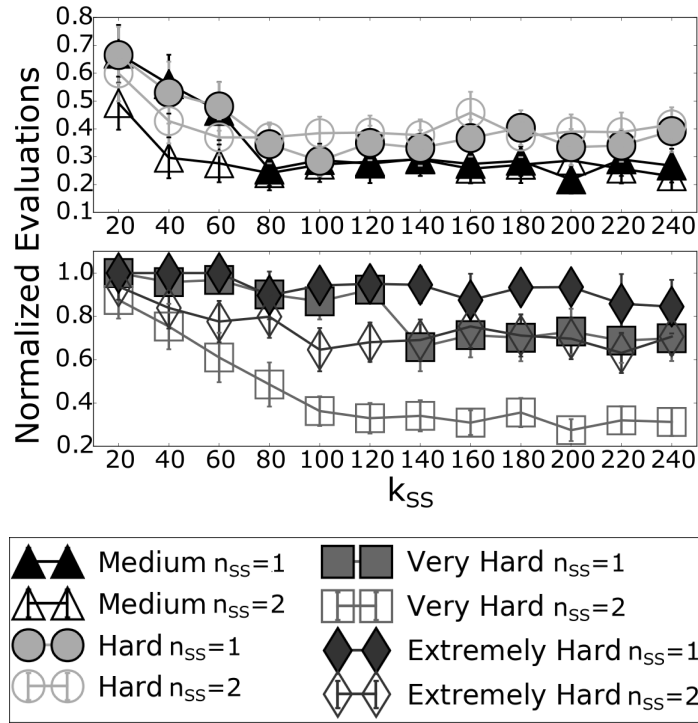


Figure 4.4: **Surprise Search: Sensitivity Analysis.** Selecting k_{SS} for surprise search: the figure depicts the average number of evaluations (normalized by the total number of evaluations allocated) obtained out of 50 runs (of 300 generations for the medium and hard maze, of 1000 generations for the very and extremely hard maze) by varying k_{SS} between 20 and 240 for all four mazes examined. The error bars represent the 95% confidence interval of the average.

clusters as its prediction points (i.e., $p_{i,j}$ in Eq. 3.2), 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_{SS} and n_{SS}) are used for these variants of surprise search.

4.1.3 Experiments and Analysis

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 by Yannakakis et al. (2003) 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 is at a 95% confidence. For multiple pairwise comparisons the Tukey's range test is used to establish significance.

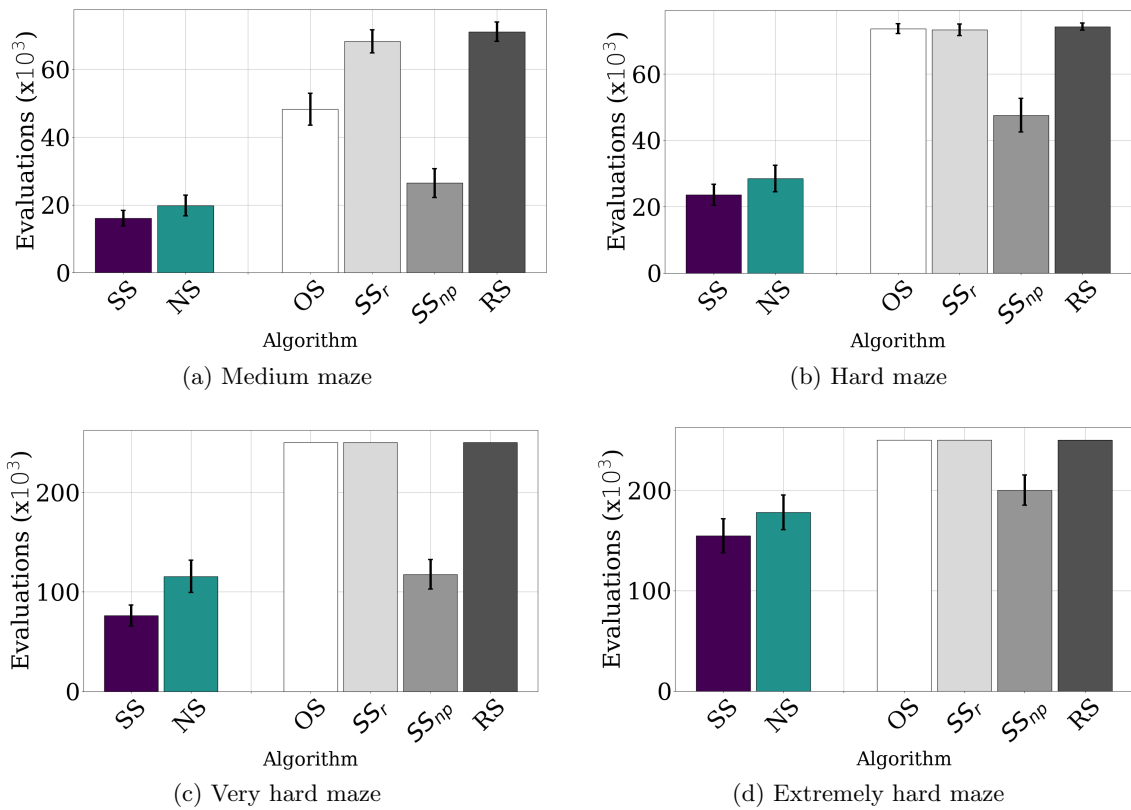


Figure 4.5: **Efficiency:** number of evaluations on average to find a solution for each maze considered. Error bars denote 95% confidence intervals. The maximum number of evaluations is $75 \cdot 10^3$ for the medium and hard maze, $250 \cdot 10^3$ for the very hard and extremely hard maze.

Efficiency

Efficiency is defined as the effort it takes an algorithm to find a solution. In particular, figure 4.5 shows the average number of evaluation to find a solution for each approach for the four mazes.

In the medium maze surprise search manages to find the goal, on average, in $16,0 \cdot 10^3$ ($\pm 2,3 \cdot 10^3$) evaluations which is faster than novelty ($19,8 \cdot 10^3 \pm 3,1 \cdot 10^3$ evaluations) and significantly faster than objective search ($48,2 \cdot 10^3 \pm 4,7 \cdot 10^3$ evaluations) and SS_{np} ($26,5 \cdot 10^3 \pm 4,24 \cdot 10^3$ evaluations). We observe the same comparative advantage in the hard maze as surprise search solves the problem in $23,6 \cdot 10^3$ ($\pm 3,2 \cdot 10^3$) evaluations on average whereas novelty search, SS_{np} , and objective search solve it in $28,5 \cdot 10^3$ ($\pm 4,0 \cdot 10^3$), $47,5 \cdot 10^3$ ($\pm 5,1 \cdot 10^3$) and $73,6 \cdot 10^3$ ($\pm 1,5 \cdot 10^3$) evaluations, respectively. Most importantly surprise search is significantly faster ($p < 0.05$) than novelty search in the very hard maze: on average surprise search finds the solution in $76,3 \cdot 10^3$ ($\pm 10,5 \cdot 10^3$) evaluations, whereas novelty search requires $115,6 \cdot 10^3$ ($\pm 16,2 \cdot 10^3$) evaluations. Also in the extremely hard maze, surprise search is faster than novelty search, as they obtain $154,8 \cdot 10^3$ ($\pm 17,0 \cdot 10^3$) evaluations and $178,1 \cdot 10^3$ ($\pm 17,2 \cdot 10^3$) evaluations, respectively. Furthermore surprise search is significantly faster ($p < 0.01$) than SS_{np} , which requires $117,6 \cdot 10^3$ ($\pm 14,8 \cdot 10^3$) and $200,2 \cdot 10^3$ ($\pm 15,1 \cdot 10^3$) 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 4.1.2. It is also clear that, on average, surprise search finds the solution faster than any other algorithm in all mazes.

Robustness

Robustness is defined as the number of successes obtained by the algorithm across time (i.e., evaluations). In Figure 4.6 we compare the robustness of each approach across the four mazes, collected from 100 runs. In the medium maze (Fig. 4.6a), 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. 4.5a; 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. 4.6b 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. 4.5b): 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}

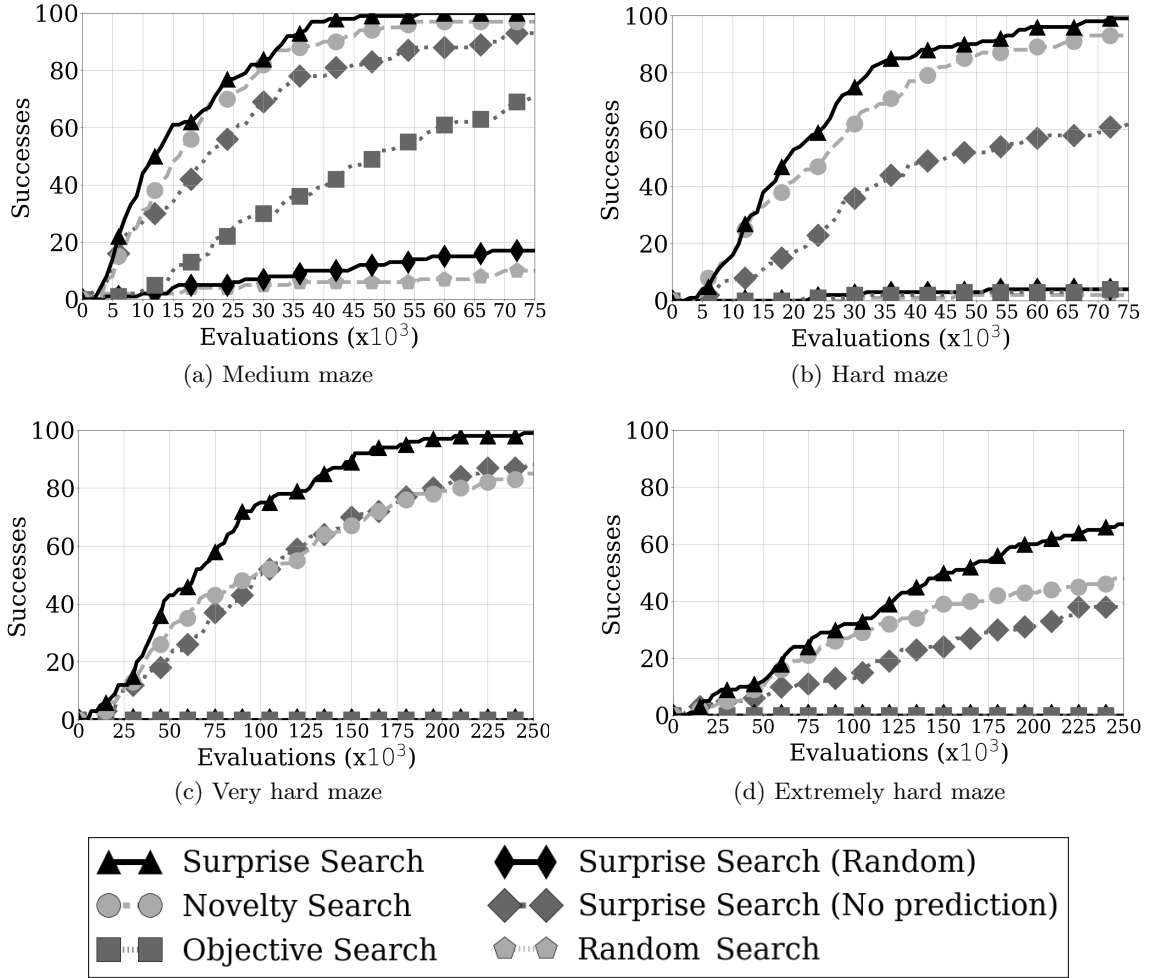


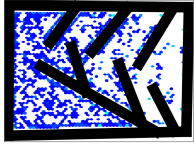
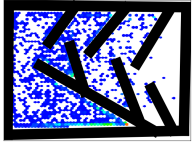


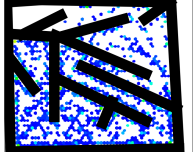



Figure 4.6: **Robustness comparison.** The graphs depict the evolution of algorithm successes in solving the maze problem over the number of evaluations.

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. 4.6c) 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. 4.6d) 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

Table 4.1: **Behavioral 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: $H = -(1/\log C) \sum_i \{(v_i/V) \log(v_i/V)\}$; 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		Hard Maze	
Novelty Search	Surprise Search	Novelty Search	Surprise Search
 $E = 25000$ $H = 0.63$	 $E = 25000$ $H = 0.67$	 $E = 25000$ $H = 0.61$	 $E = 25000$ $H = 0.67$
Very Hard Maze		Extremely Hard Maze	
Novelty Search	Surprise Search	Novelty Search	Surprise Search
 $E = 75000$ $H = 0.63$	 $E = 75000$ $H = 0.69$	 $E = 75000$ $H = 0.64$	 $E = 75000$ $H = 0.68$

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.

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. Furthermore, we perform an analysis of the computational cost to run novelty and surprise search. Objective search and the other baselines are not further analysed in this section to emphasise on comparisons between surprise and novelty search.

Behavioral Space: Table 4.1 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. 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

Table 4.2: **Genotypic Space**. Metrics of genomic complexity of the final evolved ANNs, averaged from successful runs. Values in parentheses are 95% confidence intervals.

Maze	Algorithm	Genomic Complexity	
		Connections	Hidden Nodes
Medium	Surprise	35.97 (3.35)	2.66 (0.30)
	Novelty	28.69 (1.30)	2.28 (0.23)
Hard	Surprise	52.34 (5.73)	3.85 (0.53)
	Novelty	32.56 (2.04)	2.48 (0.27)
Very Hard	Surprise	124.21 (17.01)	8.26 (1.11)
	Novelty	44.42 (3.26)	3.41 (0.45)
Extremely Hard	Surprise	179.94 (30.83)	11.43 (1.90)
	Novelty	48.14 (3.83)	3.88 (0.48)

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.

Genotypic Space: Table 4.2 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*. For genomic complexity we consider the number of connections and the number of hidden nodes of the final ANNs evolved. As noted in (Lehman and Stanley, 2011a), 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). In the more deceptive mazes, differences in genomic complexity become significantly larger. In the very hard maze the average number of connections for surprise search grows to 124.21 (± 17.01) while novelty search evolves ANNs with 44.42 (± 3.26) connections, on average; the number of hidden nodes used by surprise search is significantly larger (8.26 ± 1.11) compared to novelty search. A similar trend can be noticed in the extremely hard maze, where again surprise search evolves denser and larger 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.

4.1.4 Generality

Previously 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 by Lehman and Stanley (2011d) and test the performance of surprise, novelty and objective search as well as the baselines across numerous mazes

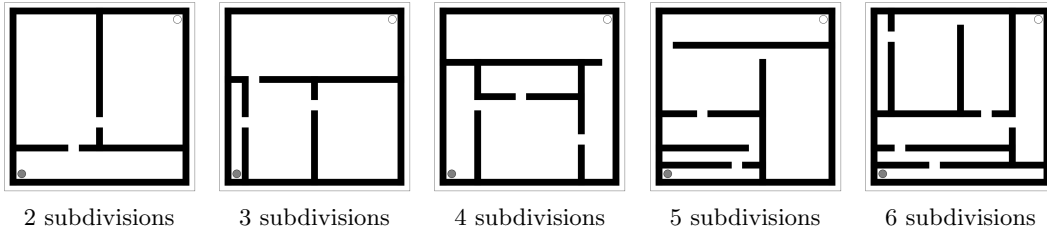


Figure 4.7: **Maze generator:** Sample generated mazes (200x200 units) created via recursive division, showing the starting location (grey filled circle) and the goal location (white circle).

generated through an automated process. Moreover, the parameters of k_{SS} and n_{SS} which were fine-tuned for the problem at hand in each of the four previous authored maze 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.

Experiment Description

To compare the capabilities in navigation policies of surprise, novelty and objective search in increasingly complex maze problems, we test their performance in 60 randomly generated mazes. These mazes are created by a recursive division algorithm (Reynolds, 2010), 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. 4.7, while the complete set of 60 generated mazes used are available in the Appendix B. The parameters of surprise search and novelty search are fixed based on well-performing setups with mazes of Section 4.1.3: surprise search uses $k_{SS} = 200$ and $n_{SS} = 2$ (used in the very hard maze) and novelty uses $n_{NS} = 15$ (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 300 and the number of generations to 600.

Experiments

As a first analysis, we report the efficiency and robustness obtained by aggregating all the runs of the 60 generated mazes for each approach, i.e., a total of 3000 runs. In terms of efficiency, from Fig. 4.8 we can observe that surprise search is faster than novelty search and significantly faster than objective search ($p < 0.05$). Surprise, novelty and objective search require on average $71, 9 \cdot 10^3 (\pm 2, 3 \cdot 10^3)$, $76, 2 \cdot 10^3 (\pm 2, 3 \cdot 10^3)$ and $84, 4 \cdot 10^3 (\pm 2, 3 \cdot 10^3)$ evaluations for each success, respectively. Furthermore, surprise search shows a significant improvement compared to the random baselines. Respectively, SS_{np} , SS_r and random

Table 4.3: **Algorithms tournament:** Percentage of 60 generated mazes for which the algorithm in a row has a strictly greater (≥ 1) number of successes compared to the algorithm in a column. Last row and last column are respectively the average of each column and the average of each row.

	OS	NS	SS	SS_{np}	SS_r	RS	Average
OS	-	15%	5%	6%	71%	78%	35%
NS	40%	-	8%	20%	75%	81%	44.8%
SS	56%	40%	-	45%	78%	85%	60.8%
SS_{np}	51%	35%	11%	-	78%	85%	52%
SS_r	1%	1%	1%	1%	-	36%	8%
RS	0%	0%	0%	0%	20%	-	4%
Average	29.6%	19%	5%	16.5%	62.8%	71.8%	-

search obtain $75,5 \cdot 10^3$ ($\pm 2,3 \cdot 10^3$), $118,7 \cdot 10^3$ ($\pm 1,9 \cdot 10^3$) and $123,6 \cdot 10^3$ ($\pm 1,8 \cdot 10^3$) evaluations; both SS_r and random search are significantly different from the performance of surprise search ($p < 0.05$). In terms of robustness, Fig. 4.9 shows that surprise search finds more successes from 112,500 evaluations onward compared to novelty and objective search. In particular, surprise search finds 1,997 successes within 150,000 evaluations, while novelty search, SS_{np} and objective search find respectively 1,838, 1,907 and 1,676 successes. Unsurprisingly, the two random baselines find significantly less successes than any other approach considered, as they find only 846 successes (SS_r) and 741 successes (random search) after the maximum number of evaluations considered.

As a further analysis on the results obtained with the 60 mazes, we focus on which of the evolutionary approaches finds strictly more successes. Table 4.3 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. As a final analysis, we investigate the genomic complexity of the successful controllers evolved by surprise and novelty search across the 60 generated mazes. Surprise search shows similar trends compared to the results obtained with the four authored mazes, as it consistently evolves larger networks compared to novelty search. In particular, SS generates on average 54.9 (± 2.3) connections, significantly more compared to NS, which evolves more sparse networks (30.1 ± 0.6). We can draw similar conclusions by looking at the average number of hidden nodes: SS evolve significantly bigger networks than NS, as they evolve on average 3.8 (± 0.2) and 2.4 (± 0.1) nodes respectively.

The reported results confirm the conclusion drawn in the four authored mazes (Section 4.1.3), as we can see that the results are consistent across the authored and the generated mazes. Surprise search is, on average, a more efficient and robust approach across different degree deceptiveness and, more importantly, it finds more successes compared to novelty search and the other baselines.

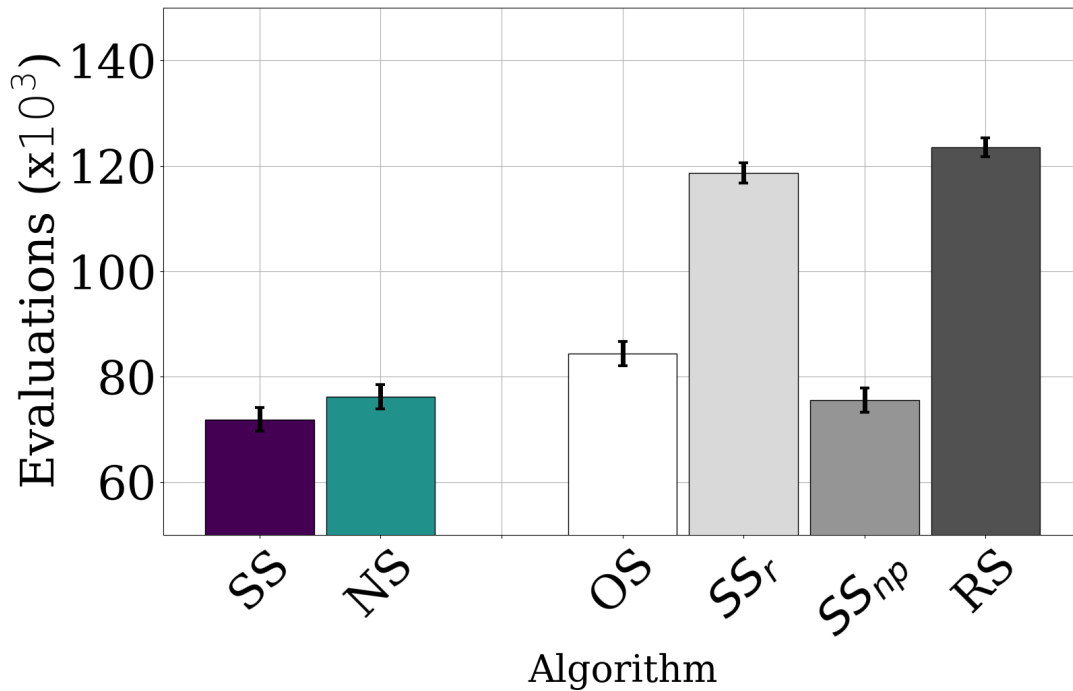


Figure 4.8: **Efficiency:** number of evaluations on average by aggregating all the runs of the 60 generated mazes for each algorithm (i.e., 3000 runs per algorithm). Error bars denote 95% confidence intervals; the maximum number of evaluations is $150 \cdot 10^3$.

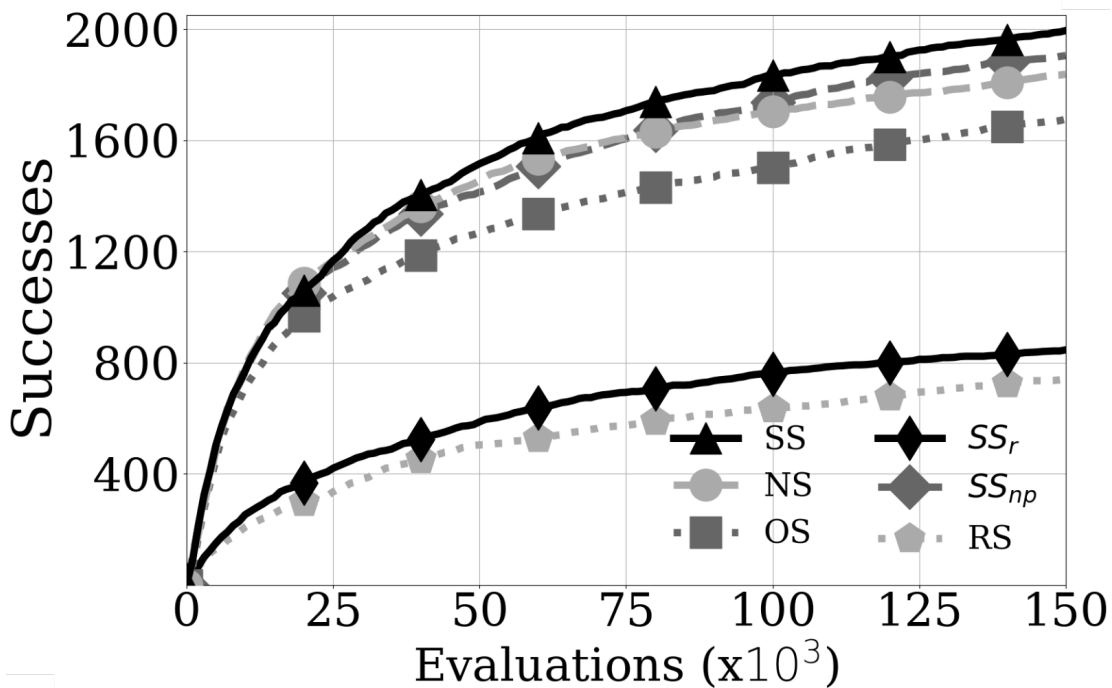


Figure 4.9: **Robustness:** algorithm successes in solving all the generated mazes over the number of evaluations for each considered method.

4.2 Soft Robot Test Bed

The goal of this section is to compare and assess the capabilities of surprise search in a more challenging and complex domain: virtual creature evolution. While the environment and encoding directly affect the design space and the expressivity of evolved robots, a poorly designed reward system can limit the potential for creative discovery in the design space (Lehman and Stanley, 2011b). For example, local optima in the fitness function can strongly bias the search towards less interesting morphologies. By explicitly ignoring the objective, open-ended evolution can instead overcome the limitations of traditional fitness-based search. This section shows how surprise search is able to discover diverse and well-performing solutions in the search space of virtual creatures.

4.2.1 Domain

As described in Chapter 2, we employ the methodology proposed in (Cheney et al., 2013) to evolve soft morphologies, where CPPNs are used as representation (Stanley, 2007) and evolved using the neuroevolution of augmenting topologies algorithm (Stanley and Miikkulainen, 2002). The evaluation is based on data collected through simulations run on VoxCad by Hiller and Lipson (2012), which simulates the statics, dynamics and non-linear deformation of heterogeneous soft materials quantitatively. The simulation framework can reproduce several materials, both active (volumetric actuated materials) and passive (for example soft and hard tissue with different stiffness). Following Cheney et al. (2013), soft robots consist of four materials, two active (red and green) and two passive (cyan and blue). Green voxels expand and contract following a signal at a predefined frequency, while a counter-phase signal actuates the red voxels. Passive materials are not actuated but are deformed by nearby materials: cyan voxels are soft (low stiffness), while blue voxels are harder and stiffer. These voxels are placed on a 3D lattice with a predefined resolution; the evolved morphologies are simulated via VoxCad (Hiller and Lipson, 2012) and the resulting behavior is used to compute the fitness of the evolved robot. As in (Cheney et al., 2013), a CPPN is used to determine the material (if any) of each voxel. Each x, y, z coordinate of the cubic lattice is provided as input to the CPPN: its first output determines whether the voxel is empty, while the highest score of the remaining four outputs decides the material of that voxel (see Figure 4.10).

4.2.2 Algorithms

Three different search methodologies are tested, and the fittest individuals of 90 independent runs are collected and analyzed in terms of efficiency, robustness and diversity. Robots evolved by surprise search are compared with two algorithms: novelty search and objective (i.e., fitness-based) search.

Objective search

In fitness-oriented search, the objective is to evolve robots capable of moving as far away as possible from a fixed starting point. Based on (Cheney et al., 2013; Methenitis et al., 2015), the chosen performance metric is the Euclidean distance of the robot’s center of mass between the initial and the end of simulation time in body lengths (see Figure 4.11a). Objective search attempts to maximize this distance.

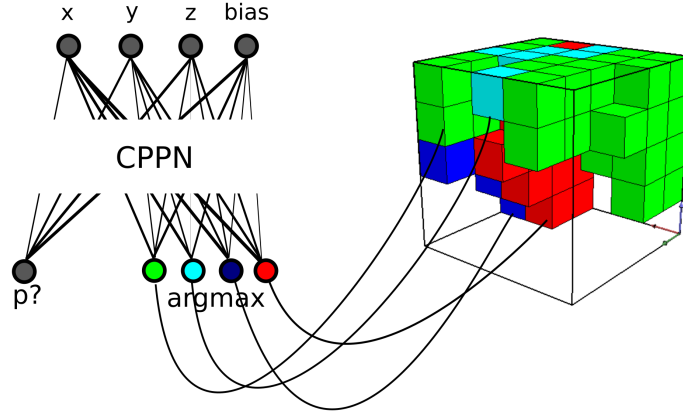


Figure 4.10: **Representation:** a CPPN describes the materials of a $5 \times 5 \times 5$ lattice.

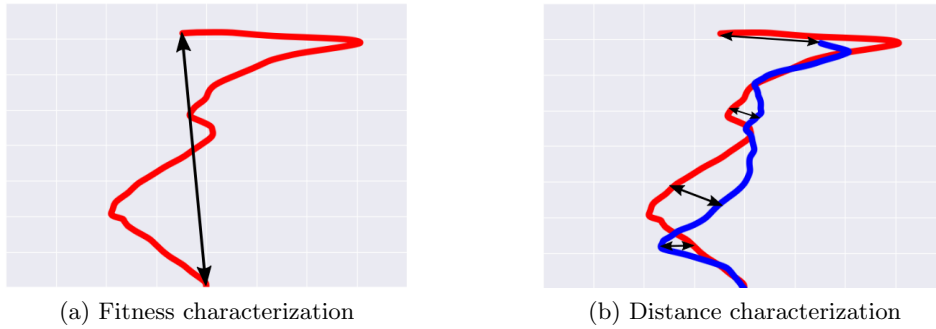


Figure 4.11: **Behavior Characterization:** behaviour characterization used for objective (the euclidean distance between starting point and ending point) and novelty (the average distance of two trajectories sampled at the same rate; here only 5 samples are shown).

Divergent Search

Divergent algorithms such as novelty search and surprise search require a different behaviour characterization in order to compute the distance between individuals ($d_{NS}(i, j)$ in Eq. 2.3 and $d_{SS}(i, j)$ and Eq. 3.2). Several behaviour characterizations have been explored in (Methenitis et al., 2015), such as the number of voxels touching the ground, the kinetic energy or the pressure. A straightforward behaviour characterization is the trajectory of the soft robot during simulation, which is directly correlated with the robot's displacement. The two-dimensional trajectory¹ of the soft robots has proved to be best in achieving good performance with novelty search (Methenitis et al., 2015). For a fair comparison between novelty search and surprise search, the distance characterization for both the algorithms is the average of the Euclidean distance of the sampled points of two trajectories t_i and t_j (see Eq. 4.1 and Fig. 4.11b). All trajectories start at the same point and are sampled at a fixed rate, which guarantees a behaviour with a fixed length. Moreover, all trajectories are transformed to make the computed measures rotation invariant, i.e., all points are rotated so that the average over all path points fall on the x -axis.

¹The three-dimensional trajectory of the robot is simplified by ignoring the height (z) component.

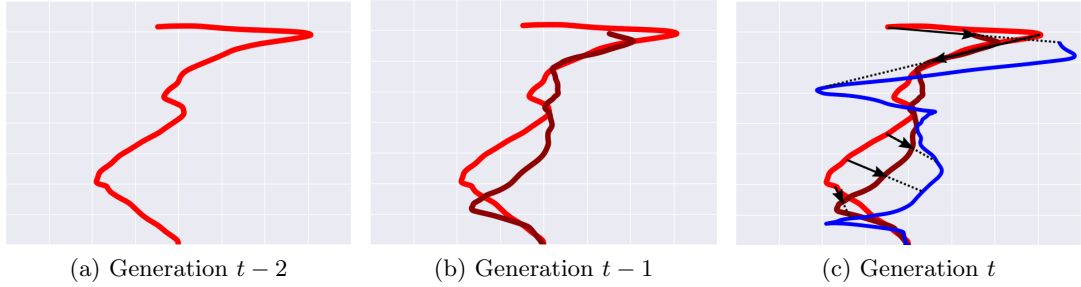


Figure 4.12: **Surprise Behavior Characterization:** The key phases of the surprise search algorithm. Surprise search uses a history of two generations ($h = 2$) and 15 clusters ($k_{SS} = 15$) in this example. One cluster’s centroid in generations $t - 2$ and $t - 1$ as well as their predictions are depicted, respectively, as red, dark red and blue lines.

$$dist(i, j) = \sum_{k=1}^K \|t_{i,k} - t_{j,k}\|, \quad (4.1)$$

where $t_{i,k}$ is the position of the robot i at the simulation step k . K is the total number of samples considered during the simulation.

Novelty Search: Novelty search uses the same parameters as in (Methenitis et al., 2015); the novelty score is computed as the average distance of 10 nearest neighbours ($n_{NS} = 10$ in Eq. 2.3) using Eq. 4.1 for d_{NS} .

Surprise search: As described in Chapter 3, surprise search relies on a prediction model and a distance function. The surprise score is computed as the Euclidean distance between the individual’s trajectory and the four closest predicted trajectories ($n_{SS} = 4$ in Eq. 3.2). The predicted trajectories are computed by using linear regression of the sampled point of the previous two generations ($h = 2$ in Eq. 3.2). The local behaviours are computed via the k -means clustering algorithm (see Fig. 4.13), where $k_{SS} = 15$ (in Eq. 3.1), found empirically based on the best objective score acquired as per Section 4.2.2. Figure 4.12 illustrates the prediction process works for one cluster centroid. When calculating the surprise score for generation t , the robots of generation $t - 2$ are clustered into k_{SS} trajectory centroids based on k -means; in generation $t - 1$ the algorithm computes another set of k_{SS} clusters. Finally, at generation t , k_{SS} prediction are computed via linear interpolation from $t - 2$ to $t - 1$. The surprise score (Eq. 3.2) is then calculated as the average distance from the four closest predicted trajectories, using Eq. 4.1 for d_{SS} .

4.2.3 Experiments and Analysis

Previous work has explored the effectiveness of divergent algorithms such as novelty search in a soft robot’s environment (Methenitis et al., 2015). However, in this section we want to extensively assess how surprise can affect the outcome of soft robot evolution. Therefore we propose to analyze the differences of three algorithms on different dimensions, both in terms of performance and variety of structures.

Reported results are collected from the 90 fittest individuals in 90 independent evolu-

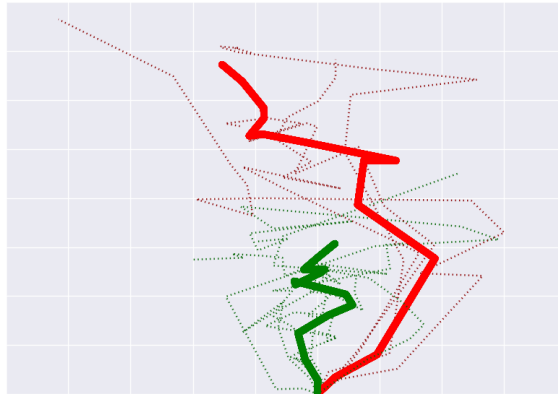


Figure 4.13: **Visualization of the k -means calculation for surprise search:** the thick red and green lines are, respectively, two example centroid trajectories obtained by clustering the dotted red and dotted green robot trajectories.

tionary runs across 8 different resolutions² (based on the characterization of Fig. 4.11a). We also report an in-depth analysis of the behaviours evolved for a specific resolution. Reported significance is at a 95% confidence. For multiple pairwise comparisons the Tukey’s range test is used to establish significance.

Experiment parameters

The simulation in VoxCad uses the same parameters as (Methenitis et al., 2015), in particular a gravity of -27.6 m/s^2 , a simulation time of 0.4 seconds, a rate of 40 Hertz for the signal that actuates active voxels and a sampling rate of 100 Hertz. Robots evolved in this chapter have a lattice resolution between 3^3 and 10^3 included. The evolutionary algorithm has a population of 30 individuals, which evolve for 1000 generations. It should be noted that, unlike the previous domain, the selection mechanism is generational. Other CPPN-NEAT parameters are the same as in (Cheney et al., 2013), and an interested reader may find a list of them and their respective values in the Appendix A.

The main goals of robot locomotion are efficient (via the objective characterization in Fig. 4.11a) behaviours. Focusing exclusively on the 90 fittest individuals collected from 90 independent runs across 8 resolutions, we report the results in terms of efficiency and robustness. All values are normalised to the dimension of the 3D lattice (i.e., every value is in body lengths of the robot).

Efficiency

Efficiency is defined as the average number of body lengths covered by the fittest robot evolved for each method, the same fitness characterization used for objective search, as in Fig. 4.11a.

If we look at the results, it is obvious from Table 4.4 that objective search creates less efficient robots compared to both novelty and surprise search. Despite the fact that objective search explicitly selects robots based on this fitness, surprise search and novelty search explore different behaviours and manages to evolve structures that reach further. The reported results show that the two divergent methods perform in a similar way, with

²The fittest individual of each independent evolutionary run is selected.

Table 4.4: **Efficiency**: average efficiency computed from 90 independent runs (95% confidence interval in parentheses). Bold values are significantly different from all the other approaches.

	OS	NS	SS
3x3x3	7.74 (0.50)	8.88 (0.19)	9.32 (0.26)
4x4x4	8.13 (0.16)	9.89 (0.35)	10.71 (0.34)
5x5x5	7.51 (0.36)	11.13 (0.33)	10.73 (0.37)
6x6x6	8.38 (0.44)	11.15 (0.28)	11.43 (0.48)
7x7x7	8.07 (0.41)	11.03 (0.37)	10.57 (0.32)
8x8x8	9.23 (0.66)	11.47 (0.39)	11.35 (0.38)
9x9x9	8.32 (0.47)	11.48 (0.37)	11.08 (0.32)
10x10x10	9.29 (0.67)	11.32 (0.38)	11.18 (0.43)

an advantage for surprise search for the lower resolutions (significantly at $4 \times 4 \times 4$, $p < 0.05$) while novelty search performs slightly better in the higher resolutions ($5 \times 5 \times 5$, $7 \times 7 \times 7$, $8 \times 8 \times 8$, and $10 \times 10 \times 10$), but not significantly.

Robustness

Robustness for soft robot evolution is defined as the number of fittest robots able to cover a certain threshold distance from a fixed starting point. Unlike maze navigation, where success is implicitly given by the domain formulation, in this test bed what a “successful robot” means is more difficult to define. Assuming that the main interest lies in robots that cover the maximum possible distance, we accept that a robot can be considered successful if it covers at least a certain distance’s threshold. Therefore, the robustness metric is equal to the number of robots that are able to cover a least a distance equal to the selected threshold, where body lengths is the unit used. (based on the fitness characterization used in 4.11a)

Fig. 4.14 shows how robustness changes across 10 different thresholds (from 5 to 15 body lengths), for the 8 selected resolutions. Unsurprisingly, objective search is constantly outperformed by the two divergent approaches. However, it is possible to notice that for certain resolutions (e.g., 8^3 and 10^3) objective search eventually reaches the performances of the other two approaches for the higher thresholds. This implicates that in some fortuitous runs objective search is able to escape the local optimum and find an effective morphology.

On the other hand, as previously noticed, the two divergent approaches have similar performances. If we focus on the lower resolutions, surprise search shows better performances compared to novelty search, in particular for the resolution 4^3 , where SS finds more or equal successes compared to novelty search across all the 10 thresholds. In contrast, in the higher resolutions, novelty search performs better, especially in the two resolutions 5^3 and 9^3 , where NS constantly finds more successful morphologies.

Structural variety

While ultimately the interest lies in the soft robots’ performances, the robot’s structure is what is evolved via the CPPN representation and, evidently, can severely affect its walk cycle. This section examines the structural patterns and variety of the 90 fittest robots from 90 independent runs.

To evaluate the variety of morphologies evolved by the three approaches, we investigate how each algorithm explores the structural space in two main feature dimensions, the num-

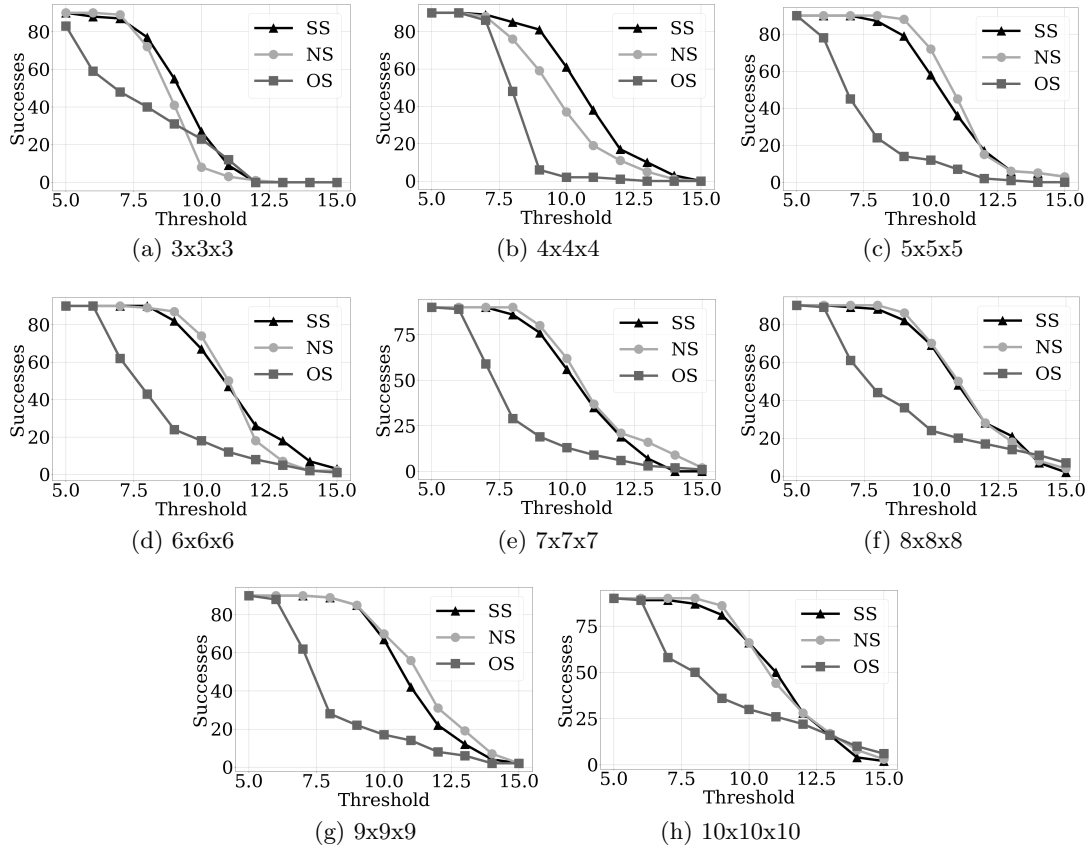


Figure 4.14: **Robustness:** number of successes across different thresholds for the 8 lattice’s resolutions considered. The graphs depict the number of successes for different performance thresholds.

ber of filled materials and the number of bones (i.e., blue voxels). In order to investigate how the different search processes explore the space of robot structures, we take inspiration from the feature mapping employed in the MAP-elites algorithm (Mouret and Clune, 2015) to evolve soft robots. Unlike MAP-elites (Mouret and Clune, 2015), structural diversity is not explicitly targeted in this case; it is interesting to see how this space is explored when the diversity criterion is behavior in terms of movement trails. To assess structural diversity, we compute the feature maps using the individuals evolved in one run sampled every 10 generations, and we add the individual in the map if the selected bin is empty or the fitness is lower compared to the individual tested. In total, therefore, $3 \cdot 10^3$ individuals are tested per run; results are averaged from 90 independent runs per lattice resolution. The two feature dimensions are the same as in (Mouret and Clune, 2015): the percentage of the voxels filled (x -axis), and the percentage of blue stiff voxels, i.e., bones (y -axis). An example of the feature maps and the binning method is shown in Fig. 4.16. As the smallest lattice resolution is 27 voxels, the feature maps have a resolution of 27×27 .

Fig. 4.15 shows the number of explored bins averaged across 90 runs for each of the eight lattice resolutions. Interestingly, novelty search and objective search tend to explore fewer bins in the two feature dimensions considered, while surprise search is able to explore more broadly; the reported metrics of OS, NS, and SS are significantly different for all the

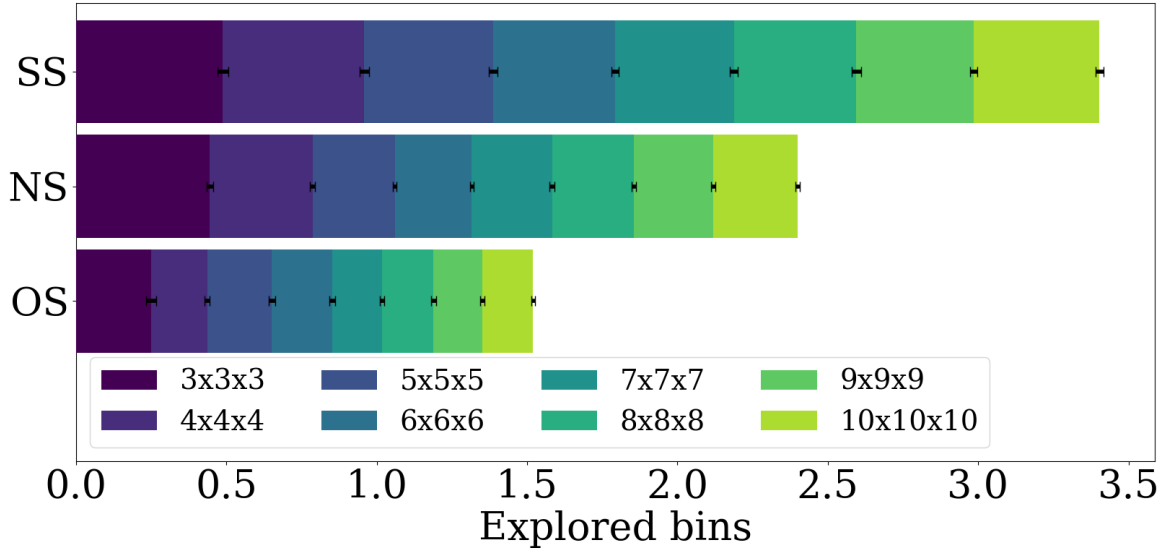


Figure 4.15: **Structural variety**: average number of explored bins for all feature maps. Each bar is normalized by the maximum number of possible bins and error bars display the 95% confidence interval of the average shown.

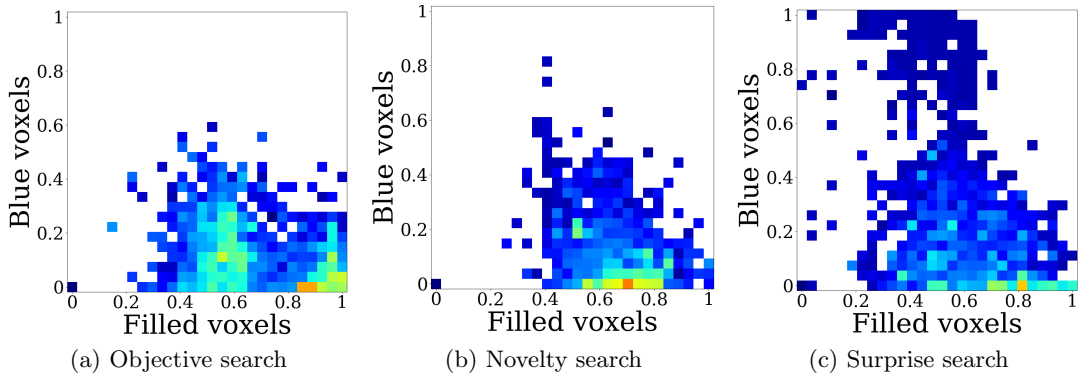


Figure 4.16: **Feature maps**: sample feature maps produced by the three methods, for a single evolutionary run on a resolution of $5 \times 5 \times 5$. White bins do not have any robots, while colored bins denote the fitness of the best individual (blue for low fitness, red for high fitness).

resolutions considered ($p < 0.05$). In terms of the structures favored by the different EC methods, novelty search tends to favor consistently more filled structures composed of more active materials; on the other hand, SS tends to explore less filled structures composed of more non-reactive materials, as can be noticed in the example of Fig. 4.16 for the resolution $5 \times 5 \times 5$; samples taken from all the 8 resolutions are reported in Appendix C.

Typical Behaviour Analysis

In order to have a better understanding of the reasons behind the performance and morphologies evolved by each algorithm, it is worthwhile to investigate what behaviours are favoured by each algorithm. For the sake of visualization, we report here only one resolu-

Table 4.5: **Behavioural analysis:** behavioural performance metrics (resolution $5 \times 5 \times 5$) as the mean values of 90 independent runs (95% confidence interval in parentheses). Bold values are significantly different from all other methods.

	Objective	Novelty	Surprise
Trajectory length	7.64 (0.44)	12.91 (0.33)	12.14 (0.36)
Deviation	0.22 (0.03)	0.59 (0.06)	0.63 (0.07)
Max velocity	34.14 (2.28)	57.24 (1.27)	53.57 (1.81)
Mean velocity	19.11 (1.11)	32.28 (0.84)	30.37 (0.92)

tion, i.e., 5^3 ; an interested reader may find all the 8 resolutions in Appendix C.

Specifically, Table 4.5 shows that in terms of actual trajectory length (i.e., length of the red line in Fig. 4.11a), robots evolved via novelty and surprise unsurprisingly make longer trails than objective-based search. Interestingly for these two approaches the trail is much longer than the Euclidean distance (i.e., their efficiency, see Table 4.4), while for objective these values are fairly similar. Indeed, calculating deviation from the shortest path as the average distance of each point in the trail from the shortest distance (i.e., the distance between the red line and the black line in Fig. 4.11a) it is obvious that the trails of novelty and surprise are curvier (higher deviation) while trails of objective are more straight.

Based on the observations regarding trail lengths, it comes as no surprise that both the mean and the maximum velocity of robots evolved via objective search is significantly lower than that of novelty and surprise search. This indicates that robots evolved for objective, surprise and novelty have different walking rhythms and gaits. Fig. 4.17a shows the velocity on the $x - y$ plane for the resolution $5 \times 5 \times 5$ (i.e., Euclidean distance of consecutive samples in the trajectory over their time interval). It is obvious that while robots evolved for objective have a steady rhythm of speeding up and slowing down, in line with the periodic signal that actuates active voxels, robots evolved for both divergent search approaches maintain a constantly high speed, with novelty showing erratic speed changes not in sync with the signal. From visual inspection, another characteristic of these fast robots evolved by novelty and surprise search is that they move more drastically along the z axis (performing jumps). Fig. 4.17b shows the velocity only in terms of changes in the z axis, where it is obvious that robots from divergent search have more vertical movement overall.

To demonstrate the impact of different robot structures on their behaviour, Fig. 4.18 shows snippets of the walk cycles sampled from the fittest individuals evolved by each evolutionary approach. Fig. 4.18a shows the movement of the pyramid-like structure favoured by objective search, which moves slowly along the ground. It is very stable with the centre of mass at the bottom, and this allows it to keep moving even in longer simulations; however, as one of its pseudopods is made of inactive (blue) voxels, its movement is curved and it ends up walking in circles in longer simulations. On the other hand, both the green cube of Fig. 4.18b and the red-green cube of Fig. 4.18c perform the aforementioned jumps by contracting and expanding their entire body (almost), but as they move they also rotate (90 degrees for Fig. 4.18b, 180 degrees for Fig. 4.18c).

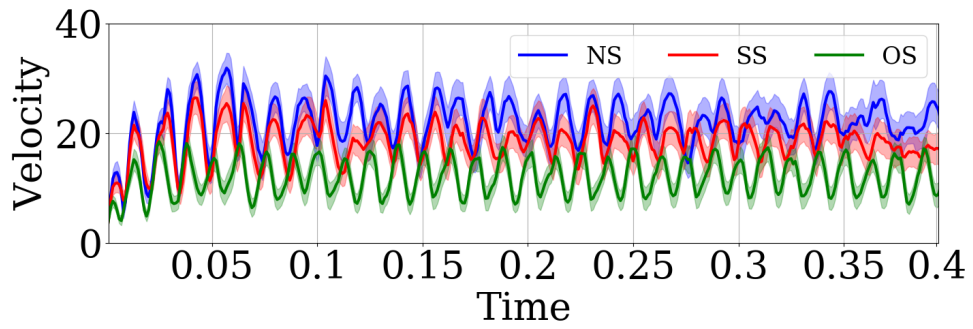
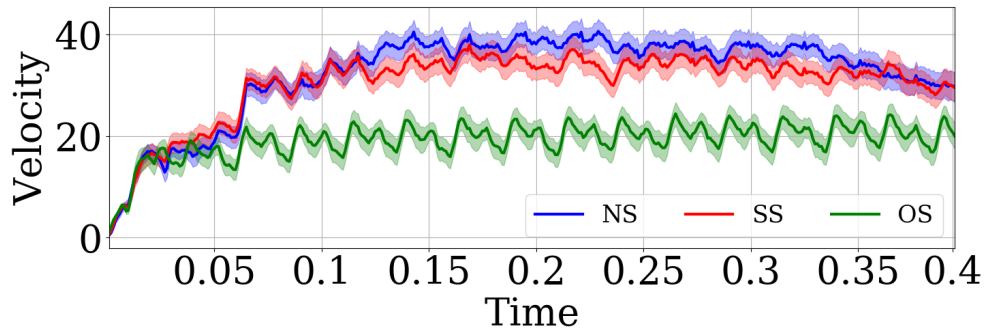


Figure 4.17: **Velocity**: mean of velocities over time for the resolution $5 \times 5 \times 5$ (95% confidence interval as error bar).

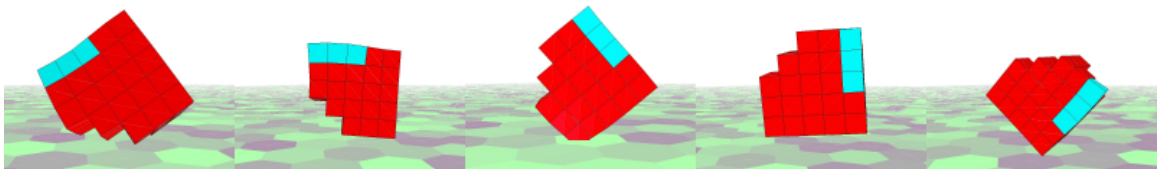
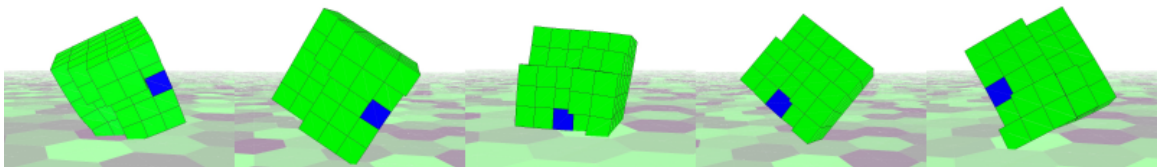
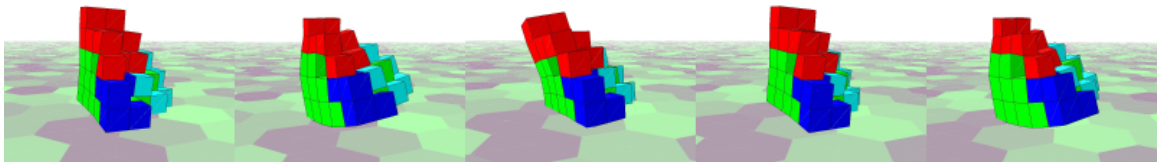


Figure 4.18: **Walk cycles**: walk cycles of different soft robots

4.3 Discussion

This dissertation identified the notion of *surprise*, i.e., deviation from expectations, as an alternative measure of divergence (compared to the notion of novelty) and presented a general framework for incorporating surprise in evolutionary search in Chapter 3. 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 objective search), experiment in two domains have been carried out and comparisons between algorithms were made on several dimensions. The key findings of these experiments suggest that in maze navigation problems 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 the soft robot domain, instead, surprise yields comparable performances to novelty search. This probably can be attributed to the more complex behaviour characterization used. However, in terms of variety, surprise search explores more extensively and consistently the search space across eight resolutions.

Based on the findings in the maze navigation experiment, the difference between the two algorithms is manifested in both the behavioral and the genotypic space. Surprise search is more exploratory than novelty search in the deceptive mazes examined as it leads to higher spatial diversity. Spatial exploration in surprise search increases over time, gradually improving the search capacity of the algorithm. Furthermore, surprise search yields larger and denser ANN controllers while diversifying the population more than novelty search. In summary, the combined characteristics of higher population diversity, ANN connectivity and exploratory capacity indicate that surprise search performs a different and more effective type of search compared to novelty. Furthermore, 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 behavior is beneficial for divergent search.

Looking at the soft robot test bed, surprise search performs similarly to novelty search—with some differences across the eight resolutions considered—and unsurprisingly, it constantly outperforms objective search. However, surprise search tends to explore the morphological space more expansively—especially in terms of the volume of filled materials or passive materials. This can be an advantage for a quality diversity algorithm, as by combining the search capabilities of novelty and surprise might bring better performance overall, as we will see in Chapter 6. Results obtained by objective search clearly show that the problem is deceptive, as it fails to reach similar performances compared to all divergent approaches in any of the 8 lattice resolutions tested. Moreover, the robots evolved by objective search are structurally more similar to each other compared to those evolved via the different divergent search alternatives. Through the in-depth analysis of the evolved robots' behavioural characteristics for the resolution 5^3 , there are several insights on how the search processes differ. Overall, the behaviours outcomes of the two divergent search algorithms were not particularly different: both novelty search and surprise search evolve fast-moving, hopping robots which however are not particularly good at controlling their trajectory. The trajectories of robots evolved for divergent search are curved and erratic (compared to a straight line from start to finish). Contrarily, objective search is shown to evolve very different behaviours than divergent search approaches, with most metrics being significantly different to both of them. Their movement is closer to a straight line, and they

do not tend to hop or move quickly.

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 maze navigation; moreover, in the more challenging domain of soft virtual creatures, surprise search has shown broader search capabilities. The comparative advantages of surprise search over novelty search are inherent to the way the algorithm searches, attempting to deviate from predicted *unseen* behaviors instead of prior *seen* behaviors. Furthermore, through our analysis, we have identified qualitative characteristics of the algorithm that gave us critical insights on the way the algorithm operates. Compared to novelty search, surprise search may also deviate from expected behaviors that exist in areas that have been visited in the past by the algorithm. However, we still lack empirical evidence on the reasons the algorithm manages to perform that well compared to other divergent search algorithms. A possible intuition about the comparative benefits of surprise search is that the algorithm allows search to *revisit* areas in the behavioral space. Such a behavior, in contrast, is penalized 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 behavior space—in a form of *backtracking* or cyclical manner. As a result of this shifting selection pressure in the behavioral space a different strategy is adopted every time a particular area is revisited. As explained in detail in Chapter 2, this is a beneficial emergent search behaviour because already visited solutions might *hide* potentially interesting path, that cannot be found unless the algorithm *backtracks* to previously encountered areas of the search space. 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 characterize *areas* in the behavioral space. Such an algorithmic behavior appears to be beneficial for search and might explain why ANNs get significantly larger in surprise search in the maze navigation domain and why the structural space explored is significantly higher in the soft robot domain.

4.4 Summary

This chapter tested the algorithm proposed in this dissertation, surprise search, in two different testbeds, maze navigation and soft robot evolution. The objectives of these experiments were to test the validity of the proposed approach and to compare its performance against a state-of-the-art divergent algorithm, novelty search. The first section has presented the maze navigation testbed. In particular, initially we proposed four different authored mazes, of increasing complexity and deceptiveness. We tested each algorithm on three main aspects: efficiency, robustness and analysis of the ANN evolved. The results have shown that, in this domain, surprise search is more efficient and robust compared to novelty search and objective search. Moreover, it seems that in the most deceptive mazes, rewarding surprise is even more advantageous compared to novelty search. In order to test the generality of the results obtained with the four authored mazes, we presented a more general and extensive maze navigation task, composed of 60 automatically generated mazes, and we described the experiments conducted regarding efficiency and robustness obtained. Furthermore, we performed an analysis in terms of successes obtained in a tournament-fashion comparison between the approaches tested; this analysis has corroborated the results obtained in the authored mazes.

In the second section we described experiments conducted in a different and more complex task: soft robot evolution. Based on the analysis of the results obtained in terms of efficiency, robustness and diversity of the morphologies evolved, we have shown that searching for surprising solutions can be advantageous on the diversity of the evolved robots, whereas performance-wise novelty and surprise search yield comparable results. These findings support the idea that deviation from expected behaviors can be a powerful alternative to divergent search with key benefits over novelty or objective search. Generally, our analysis indicates that different divergent rewards can produce diverse and effective solutions. Since novelty and surprise were shown to be different, we can imagine that combining the two algorithms can be a promising approach. This idea will be tested in the next chapter.

Chapter 5

Fusing Novelty and Surprise: Experiments

Divergent search techniques applied to evolutionary computation, such as novelty search and surprise search, have demonstrated their efficacy in highly deceptive problems compared to traditional objective-based fitness evolutionary processes, as shown in Chapter 4. While novelty search rewards unseen solutions, surprise search rewards unexpected solutions. As a result, these two algorithms perform a different form of search since an expected solution can be novel while an already seen solution can be surprising. As novelty and surprise search have already shown much promise individually in difficult domains (Chapter 4), the hypothesis is that an evolutionary process that rewards both novel and surprising solutions will be able to handle deception in a better fashion and lead to more successful solutions faster. To test our hypothesis, in this chapter we test two evolutionary algorithms we name novelty-surprise search (NSS) and novelty search-surprise search (NS-SS). The first approach is a first attempt at a combined novelty-surprise search algorithm, and it uses the simplest alternative of aggregating the novelty score and the surprise score into a single reward. The second approach, instead, combines novelty and surprise by means of a multi-objective approach. To validate these two implementations, we employ the same two testbeds used in Chapter 4: maze navigation and soft robot evolution. In Section 5.1 we compare the performances of the two novelty-surprise approaches against their base components in four authored mazes and 60 procedurally generated mazes. In Section 5.2 instead, we compare the performances of two proposed algorithms in a soft robot evolution domain. Finally, we discuss the results obtained by these two new algorithms and we advance an argument on why combining novelty and surprise is beneficial for evolutionary search.

5.1 Maze Navigation Testbed

The maze navigation domain presented in Chapter 4 is a good domain as it is simple to grasp and apprehend. Therefore, we propose to use the maze navigation task to validate the two new proposed divergent approaches that couple novelty and surprise. To be able to compare our findings with the results presented in the previous chapter we test the NSS and NS-SS algorithms in the previously introduced mazes, i.e., four authored mazes and 60 procedurally generated mazes (described in Section 4.1). In this task a simulated mobile robot has to find the goal in a maze in a limited number of simulation steps. The robot (Fig. 2.9b) is controlled by an artificial neural network (Fig. 2.9a), with 10 inputs (6 range-finder

sensors and 4 radar sensors) and 2 outputs (which control robot movement): more details can be found in Chapter 2 and Chapter 4.

5.1.1 Algorithms

This subsection provides a specific formulation for all the algorithms tested. Overall we want to test the performance of four algorithms: novelty search, surprise search, novelty-surprise search and novelty search-surprise search. Below we give the details of the two new algorithms proposed in this chapter, while for novelty search and surprise search we refer to the formulation given in Chapter 4 for the maze navigation domain. The NEAT algorithm uses speciation and recombination, as described in Stanley and Miikkulainen (2002), and it is steady-state as in Lehman and Stanley (2011a).

Novelty-Surprise Search

As noted in Chapter 3, the novelty-surprise search (NSS) algorithm executes both novelty and surprise search and it rewards an individual by adding its novelty and surprise score. Novelty attempts to maximize a novelty score computed in Eq. (2.3), using the Euclidean distance between the two robots' final positions at the end of simulation for calculating distance from the closest individuals. Surprise instead uses the surprise score described in Equation (3.2); as in novelty search, the behaviour of the robot is characterised by its final position in the maze. Fig. 4.3 shows the key steps involved in the computation of the predictions in the maze navigation domain. More details on how novelty and surprise search are implemented can be found in Chapter 3 and Chapter 4.

Novelty Search-Surprise Search

While NSS uses a linear aggregation of novelty and surprise, NS-SS uses a multi-objective approach to simultaneously and independently reward novelty and surprise. As noted in Chapter 3 novelty search-surprise search (NS-SS) uses a NSGA-II multi-objective algorithm (Deb et al., 2002) to search for non-dominated solutions on the dimensions of novelty (Eq. 2.3) and surprise (Eq. 3.2). In particular, we use a steady-state variant of NSGA-II (Li et al., 2017) in this domain. Until the termination condition is not met (i.e., a robot reaches the goal of the maze), we run a steady-state implementation of NEAT. Two mating parents are selected to reproduce an offspring which is evaluated through the novelty score (Eq. 2.3) and surprise score (Eq. 3.2). We then replace the worst individual in the population if the new individual is more fit, and we update the non-dominated fronts as in (Li et al., 2017). Every generation (i.e., every N offspring reproductions) we update the surprise model, as explained in Chapter 3, we evaluate and assign to the entire population the novelty and surprise scores, and we compute the non-domination fronts (Deb et al., 2002). Finally, the algorithm return in the steady-state loop until new N offspring are generated or the maximum number of generations is reached. Distance characterizations for novelty and surprise are computed as in Chapter 4, via the Euclidean distance between the two robots' final positions at the end of the simulation.

Baselines

Novelty Search ignores the objective of the problem at hand and attempts to maximize behavioral diversity. The novelty score, computed through Eq. (2.3), uses the behavioral

distance d_{NS} between two individuals; in this domain d_{NS} is the Euclidean distance between the two robots' final positions at the end of simulation. *Surprise Search* relies on a prediction model and a distance function (d_{SS}). The surprise score is computed as the Euclidean distance between the individual's final positions and the n_{SS} closest predicted behaviours. The predicted trajectories are computed by using linear regression of the sampled points of the previous two generations' final positions. The local behaviours are computed via the k -means clustering algorithm. More details can be found in Chapter 3 and Chapter 4.

5.1.2 Experiments and Analysis

Four mazes, identified as *medium*, *hard*, *very hard* and *extremely hard maze*, are used in this work. The first two mazes were introduced by Lehman and Stanley (2011a), while the more deceptive very hard and extremely hard mazes have been introduced in the previous chapter (Chapter 4).

Parameters

The domain-related parameters used are the same of Chapter 4; but we report them again here for completeness. The evolved robot is considered successful if it manages to find the goal within a radius of five units in 400 simulation steps for the medium and hard maze, 500 steps for the very hard maze and 1000 steps in the extremely hard maze. All four algorithms use NEAT to evolve a robot controller with the same parameters as in (Lehman and Stanley, 2011a). The population size is 250 and evolution is carried out for 300 generations (75,000 evaluations) for the medium and hard maze and for 1000 generations (250,000 evaluations) for the very and extremely hard maze.

Parameters for novelty search are the same as in Chapter 4, i.e., $n_{NS} = 15$ for the medium, hard and very hard maze, and $n_{NS} = 10$ for the extremely hard maze. Regarding surprise search, the surprise score is computed as the Euclidean distance between the robot and the two closest predicted points ($n_{SS} = 2$), whereas the prediction model is based on a one-step linear regression of two past generations ($h = 2$). The parameter that controls the clustering of the behaviours (k_{SS}) for the medium and hard maze is modified from the one used in Chapter 4, in order to make the overall parameter setting more consistent. In particular, we rerun a sensitivity analysis with $n_{SS} = 2$ for the two easiest mazes, and the k_{SS} values that yields the least evaluations are 100 and 50 for the medium and hard maze respectively. On the other hand k_{SS} for the two hardest mazes remains unchanged, i.e., 200 and 220 for the very and extremely hard maze respectively. For a fair comparison, the same parameters for novelty and surprise are used in NSS and NS-SS. An interested reader may find a detailed list of all the parameters used and their respective values in Appendix A.

Sensitivity Analysis

For both algorithms that couple novelty and surprise, the specific parameters of novelty search (n_{NS}) and surprise search (n_{SS} , h , k_{SS}) remain unchanged. However, while NS-SS doesn't require any more parameter tuning, NSS combines the novelty and surprise rewards linearly as in Eq. 3.4. A core component of the proposed NSS algorithm is the λ parameter which determines the impact of novelty versus surprise in Eq. 3.4: higher λ values put more weight on novelty. To select appropriate λ values we run 11 experiments for each maze,

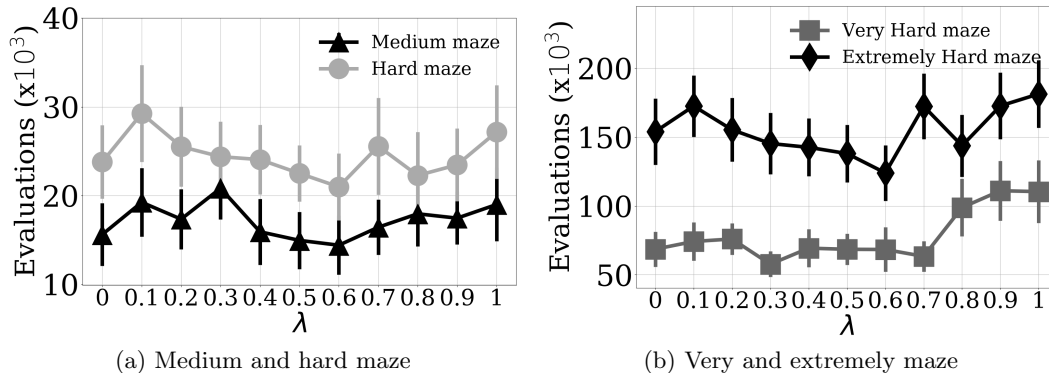


Figure 5.1: **Sensitivity analysis:** selecting λ for NSS. The figure depicts the average number of evaluations obtained out of 50 runs (of $75 \cdot 10^3$ evaluations for the medium and hard maze, of $250 \cdot 10^3$ evaluations for the very hard and extremely maze). Error bars represent the 95% confidence interval.

each time with a different λ value ranging from 0 to 1 in increments of 0.1. Each experiment is composed of 50 runs of the NSS algorithm with the particular λ parameter.

Fig. 5.1 shows the average number of evaluations required to find a solution across λ , for each maze. We pick λ values that solve the corresponding maze in the fewest possible evaluations: in the medium maze and hard maze, this happens for $\lambda = 0.6$ leading to an average of $14.4 \cdot 10^3$ and $21 \cdot 10^3$ evaluations respectively. In the very hard maze the fewest evaluations on average ($57.7 \cdot 10^3$) are found when $\lambda = 0.3$. Finally, in the extremely hard maze the fewest evaluations on average correspond to $\lambda = 0.6$ ($124 \cdot 10^3$). Based on the sensitivity analysis results, we can notice that there is a delicate balance between the two scores in order to obtain the best performance, with the exception of the very hard maze, where the surprise weight is higher ($\lambda = 0.3$), probably because the performance of vanilla surprise search are particularly good in this maze (see Chapter 4). However, in three out of four mazes, λ has bigger values, meaning that novelty is probably the primary driver of the search. A detailed comparison of NSS and NS-SS against novelty and surprise search is presented in the next section.

Results

The evolution of successful navigation policies in four deceptive mazes is used to compare the efficiency and robustness of four divergent algorithms. All results reported are computed from 50 independent runs; reported significance is at a 95% confidence. For multiple pairwise comparisons the Tukey’s range test is used to establish significance.

Efficiency: As in the analysis performed in Chapter 4, efficiency is defined as average number of evaluations to find the solution in the maze. Figure 5.2 shows the efficiency of the four algorithms considered, for each maze, where values are averaged across 50 runs of each algorithm and the values in the error bars represent the 95% confidence interval of the average. In the medium maze, the efficiency of the four algorithms is really similar, even if on average NSS and NS-SS perform better ($13,8 \cdot 10^3 \pm 2,4 \cdot 10^3$ evaluations for the former and $13,7 \cdot 10^3 \pm 2,4 \cdot 10^3$ evaluations for the latter), compared to the two base algorithms which obtain $19,0 \cdot 10^3 \pm 4,1 \cdot 10^3$ evaluations (NS), and $15,6 \cdot 10^3 \pm 3,5 \cdot 10^3$ evaluations (SS).

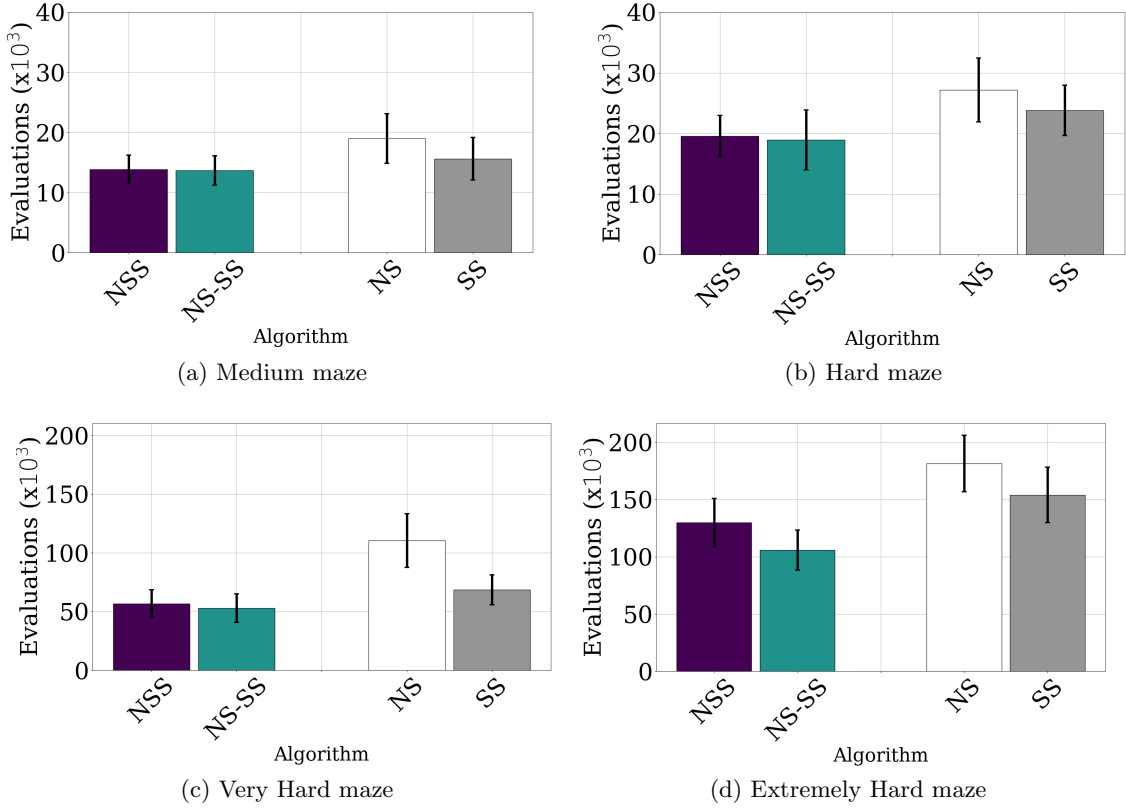


Figure 5.2: **Evaluations.** Number of evaluations on average to solve the three mazes for each algorithm. Error bars denote the 95% confidence interval. The maximum number of evaluations is $75 \cdot 10^3$ for the medium and hard maze, $250 \cdot 10^3$ for the very hard and extremely hard maze.

In the hard maze, NS-SS and NSS outperform both NS and SS (see Fig. 5.2b), but the two new algorithms perform in a similar way. NSS with $19,6 \cdot 10^3 \pm 3,4 \cdot 10^3$ evaluations on average and NS-SS with $19,0 \cdot 10^3 \pm 4,9 \cdot 10^3$ evaluations on average are significantly faster than novelty search and faster than surprise search. In the very hard maze, Fig. 5.2c shows that NSS ($56,8 \cdot 10^3 \pm 1,2 \cdot 10^3$ evaluations) and NS-SS ($53,0 \cdot 10^3 \pm 1,2 \cdot 10^3$ evaluations) outperform significantly NS ($p < 0.05$) and outperform SS. In the hardest maze (Fig. 5.2d) we can notice a bigger gap between the performance of NSS and NS-SS, as they obtain respectively $130,0 \cdot 10^3 \pm 2,1 \cdot 10^3$ and $106,0 \cdot 10^3 \pm 1,7 \cdot 10^3$ evaluations, however the difference is not significant. Most importantly, both approaches outperform significantly the two baselines in the most deceptive maze. From this initial analysis we can notice that both NSS and NS-SS are more efficient than their base components, especially in the two hardest mazes, and NS-SS shows slightly better performance compared to NSS, a difference that grows with the deception of the mazes.

Robustness: In this section we compare the algorithms' *robustness* defined as the number of successes obtained by the algorithm across time (i.e., evaluations). Fig. 4.6 shows the robustness of all four algorithms for each maze, collected from 50 independent runs. In the medium maze (Fig. 5.3a) both NSS and NS-SS are able to find on average more solutions than novelty search. On the other hand, the two approaches outperform SS

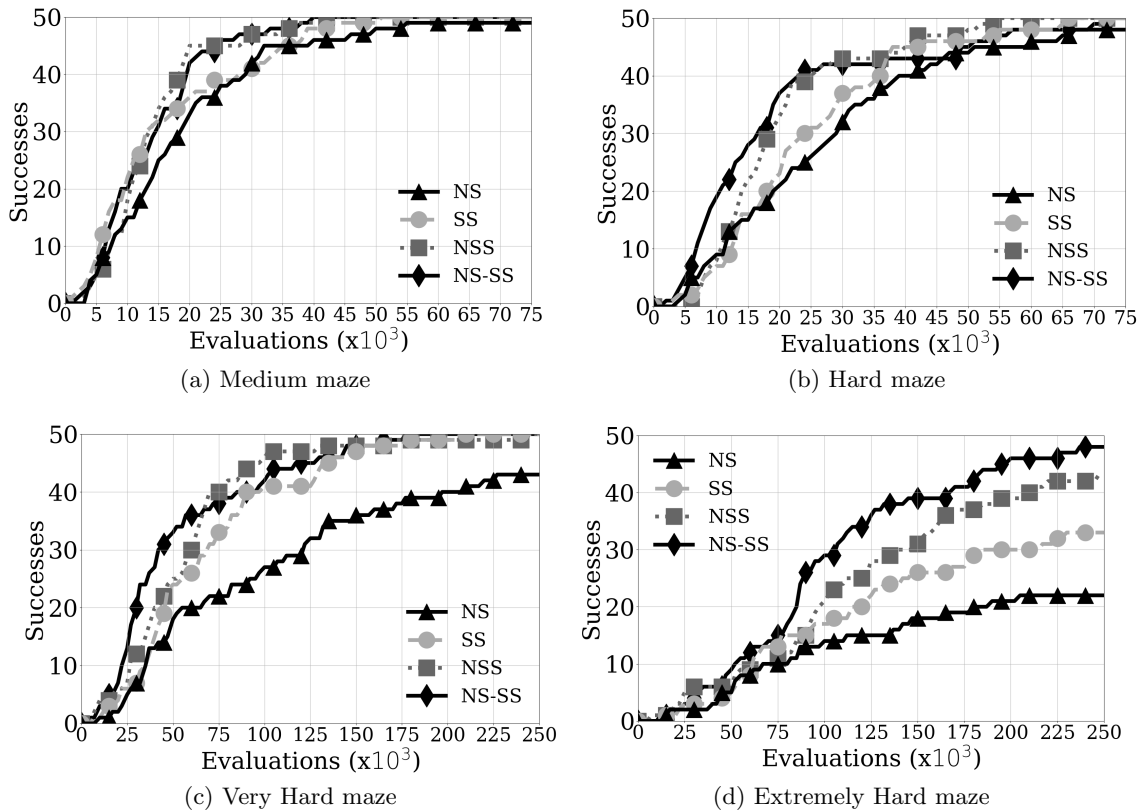


Figure 5.3: **Robustness** comparison: number of successes in solving the maze problems over the number of evaluations.

only in the interval between 15,000 and 50,000 evaluations. In the more deceptive hard maze (Fig. 5.3b) the benefit of combining novelty and surprise is more visible. NSS clearly outperforms novelty search for the entire evolutionary process, and it outperforms surprise search between 7,500 and 40,000 evaluations. NS-SS shows comparable performances to NSS, but it is faster between 5,000 and 15,000 evaluations. In the third testbed, the very hard maze, Fig. 5.3c shows how combining novelty and surprise can improve substantially the performance: from 20,000 evaluations NS becomes slower in comparison to NSS while SS also fall behind between 10,000 and 35,000 evaluations. As in the previous maze, NS-SS performs similarly to NSS, but it is faster in the beginning of the evolutionary process, in particular between 25,000 and 60,000 evaluations. In the most deceptive maze, we notice a bigger performance gap between NSS and NS-SS (Fig. 5.3d). While they clearly outperform NS and SS, we can notice that from 75,000 evaluations onwards NS-SS becomes faster in finding solutions, and it finally reaches 48 out of 50 successes, while NSS only solves the maze in 43 out of 50 runs.

As a general conclusion from the efficiency and robustness comparisons we can clearly observe that coupling novelty and surprise yields better performance both in terms of solutions found and in terms of evaluations for discovering a solution. Furthermore, if we compare NSS and NS-SS, we notice that they perform similarly in first two mazes, but in the hardest mazes, the benefit of using a multi-objective approach is evident.

Further Analysis

Additional insights on the performance of NSS and NS-SS can be gleaned by analysing the output in the behavioural space as well as the genotypic space. For the former, we observe the heatmaps of robot positions in a number of typical runs for each algorithm. For the latter, we present complexity metrics computed from the final ANNs evolved by the four algorithms.

Behavioural Space: as done in the previous chapter, Table 5.1 shows a comparison between typical runs for novelty, surprise, NSS and NS-SS, computed from experiments in the four authored mazes. Each entry describes the number of evaluations (E) taken by the algorithm to find the solution and the heatmap shows the robots' final positions throughout all evaluations. Moreover their corresponding entropy (H) is shown as a measure of their spatial diversity. For each maze, the runs shown are chosen so that the number of evaluations until a solution is discovered are similar among the four algorithms. Not surprisingly, the table shows that the two approaches that couple novelty and surprise are able to explore a larger part of the maze, especially for the very hard and extremely hard mazes: the heatmaps show that NSS and NS-SS result in a more sparse distribution of final robot positions. Furthermore, the entropy values show that, in the very hard and extremely hard maze, the diversity of final positions is always higher for each typical run considered. Investigating other runs with different E values indicated that the difference in entropy values increases when evolution takes longer to find a solution. Therefore we can infer that combining novelty with surprise is beneficial also for exploring the behavioural space: while novelty searches for unexplored points of the maze, surprise search pushes for unexpected points of the maze, which can involve backtracking to already visited places; their combination augments their respective search capacities.









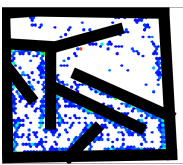




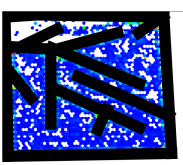



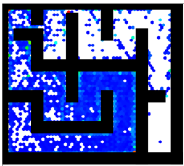
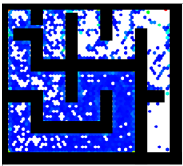

Genotypic Space: following the analysis presented in Chapter 4, Fig. 5.4 and Fig. 5.5 show the metrics collected from the final ANNs evolved by the four algorithms, focusing on their *genomic complexity*. Genomic complexity is defined as the number of hidden nodes and as the number of connections in the final ANNs. In terms of genomic complexity, we observe that NSS evolves ANNs larger than novelty search but at the same time smaller than surprise search; the same behaviour can be observed in terms of number of connections. This means that NSS is able to outperform surprise search without creating as complex networks, which alleviates the computational effort needed to evolve complex and large ANNs. On the other hand, NS-SS evolve more connected and slightly bigger networks compared to NSS for the easier two mazes (medium and hard maze). Instead, if we focus on the two harder mazes, NS-SS evolves smaller networks compared to NSS, which makes this implementation even more advantageous in terms of computational complexity.

5.1.3 Generality

In the previous chapter, we introduced a second maze navigation testbed, consisting of 60 randomly generated mazes. Therefore, it is interesting testing the capacity in solving these 60 mazes also for the newly introduced algorithms. The generated mazes and the maze navigation parameters are the same employed in the previous chapter (Section 4.1.4), i.e., 300 simulations timesteps and 600 generations (150,000 evaluations) for each approach tested.

In this subsection, we test the performances in terms of efficiency and robustness of

Table 5.1: **Behavioural Space.** Typical successful runs solved after a number of evaluations (E) on the three 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: $H = -(1/\log C) \sum_i \{(v_i/V) \log(v_i/V)\}$; 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			
NS	SS	NSS	NS-SS
			
$E = 25000$	$E = 25000$	$E = 25000$	$E = 25000$
$H = 0.63$	$H = 0.60$	$H = 0.62$	$H = 0.62$
			
Hard Maze			
NS	SS	NSS	NS-SS
			
$E = 25000$	$E = 25000$	$E = 25000$	$E = 25000$
$H = 0.61$	$H = 0.62$	$H = 0.65$	$H = 0.65$
			
Very Hard Maze			
NS	SS	NSS	NS-SS
			
$E = 75000$	$E = 75000$	$E = 75000$	$E = 75000$
$H = 0.63$	$H = 0.69$	$H = 0.71$	$H = 0.72$
			
Extremely Hard Maze			
NS	SS	NSS	NS-SS
			
$E = 75000$	$E = 75000$	$E = 75000$	$E = 75000$
$H = 0.64$	$H = 0.68$	$H = 0.69$	$H = 0.69$
			

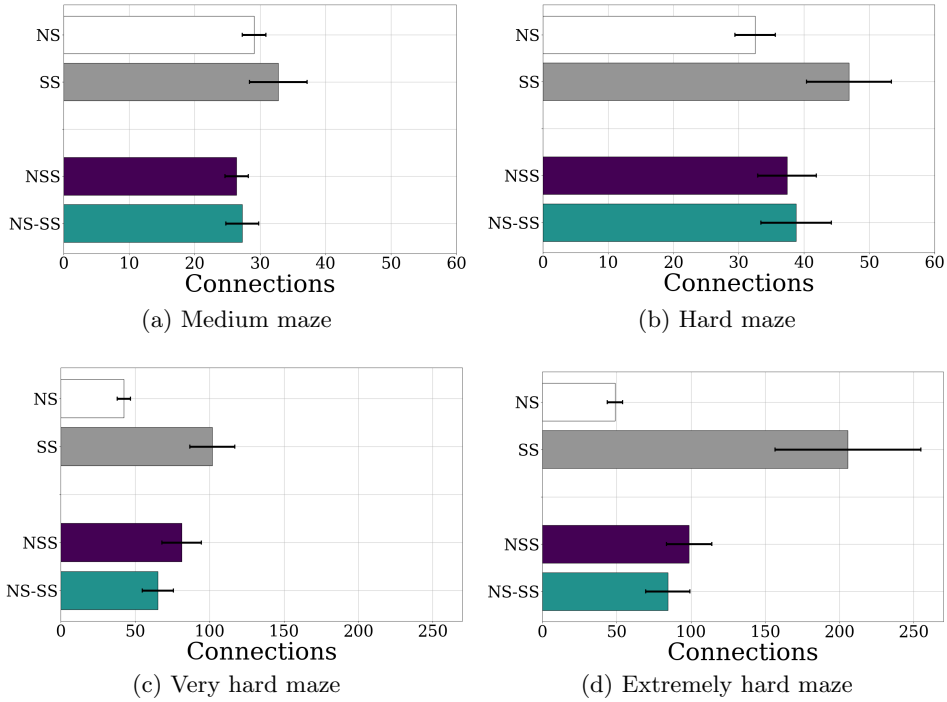


Figure 5.4: **Genotypic Space: Connections.** Metrics of genomic complexity of the final evolved ANNs, averaged from successful runs. Values in parentheses are 95% confidence intervals.

NSS, NS-SS, NS and SS against the 60 generated mazes. In particular, the parameters of four algorithms tested are based on well-performing setups used in the four authored mazes: $\lambda = 0.6$ for NSS, $n_{NS} = 15$ for NS, NSS and NS-SS, and $k_{SS} = 200$ and $n_{SS} = 2$ for SS, NSS and NS-SS.

Results

Fig. 5.6 shows the efficiency of the four algorithms by aggregating all the 3000 runs computed by using the 60 generated mazes. We can observe that both approaches that couple novelty and surprise show significantly better performance compared to their base components ($p < 0.05$). On the other hand, as in the four human-designed mazes, the performance of NSS and NS-SS are comparable ($67, 0 \cdot 10^3 \pm 2, 2 \cdot 10^3$ and $66, 2 \cdot 10^3 \pm 2, 2 \cdot 10^3$ evaluations on average respectively), even if we can see a small advantage for NS-SS. If we look at the robustness of the aggregated runs, we can draw similar conclusions. Fig. 5.7 shows that from 10,000 evaluations both NSS and NS-SS become faster compared to novelty and surprise search. From this analysis, we can see that NSS finds more solutions (2,123 solutions), while NS-SS is faster in the first 40,000 evaluations, but eventually, it reaches a slightly smaller number of solutions (2,116). On the other hand, the two baselines perform worse, as surprise search obtains 1,997 solutions and novelty search reaches 1,838 successful robots.

A further analysis involves looking at which approach obtains strictly more successes for each maze in a tournament-like comparison. From Table 5.2, interestingly surprise search seems to be more successful compared to the other three approaches, as SS “wins”

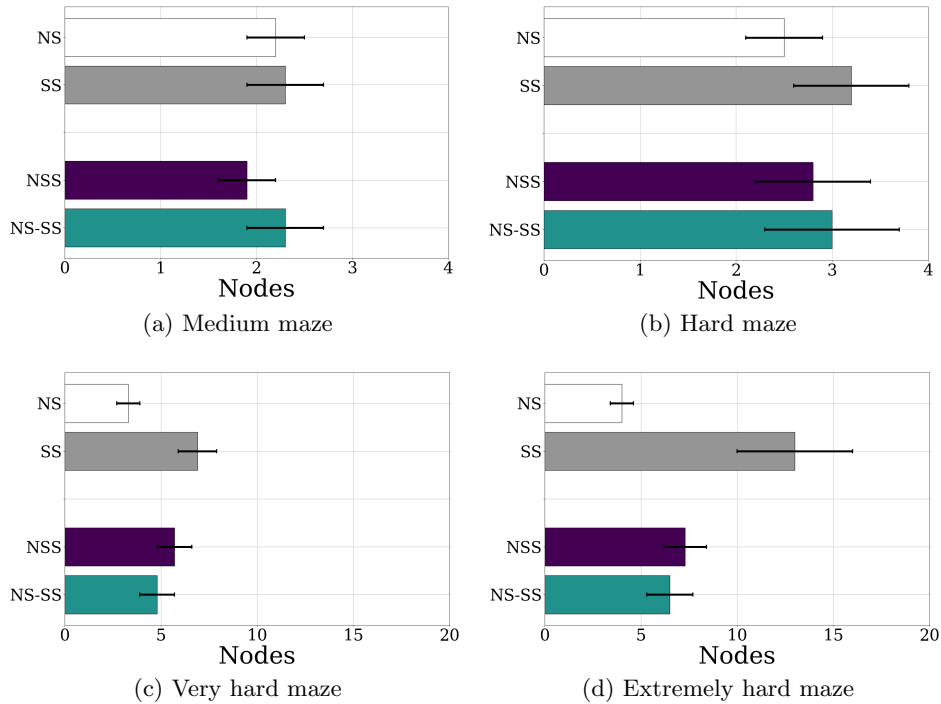


Figure 5.5: **Genotypic Space: Hidden nodes.** Metrics of genomic complexity of the final evolved ANNs, averaged from successful runs. Values in parentheses are 95% confidence intervals.

on average 40.5% comparisons, while NS-SS and NSS obtain on average respectively 33.8% and 36.6%. However, this finding is not inconsistent with the previous two analyses. In fact, in this performance comparison, we are comparing the four algorithms on a maze-by-maze basis, i.e., we compare the successes obtained for each of the 60 mazes. The results show that most of the mazes are easy to solve, and therefore surprise search is still competitive as shown in the medium and hard maze. On the contrary, when the mazes become particularly deceptive, surprise search is outperformed significantly by NS-SS and NSS, as the analysis of the aggregated runs shows.

Finally, Fig. 5.8 show the average complexity of the successful artificial networks evolved for each method, i.e., all the ANNs able to find the goal within 300 simulations steps. We see that similar patterns emerge compared to the analysis performed with the four easiest mazes. On the one hand, novelty search evolved the simplest networks, and surprise search pushed for the most complex genotypes. On the other hand, the two combinations of novelty and surprise evolve network of similar sizes, in between the two extremes represented by NS and SS. Overall the sizes of these ANNs are similar to the ones evolved for the two easiest authored mazes: this is probably a further confirmation that these generated mazes can be in general solved easily by all the approaches, and the differences emerge only in the hardest ones.

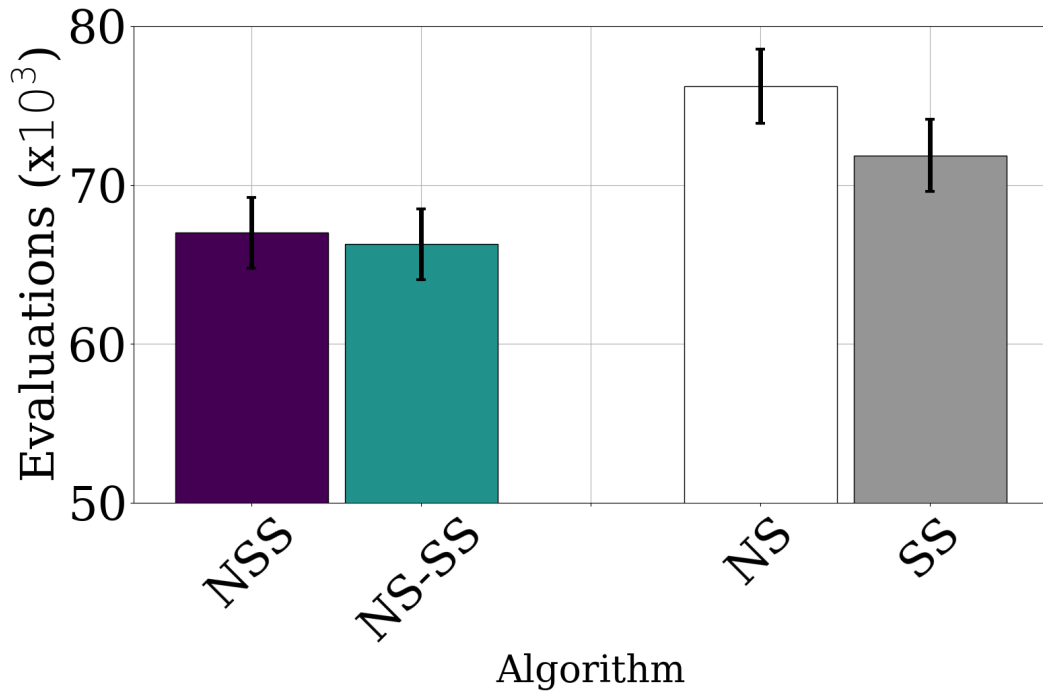


Figure 5.6: **Efficiency:** algorithm successes in solving all the generated mazes over the number of evaluations for each considered method. Error bars denote 95% confidence intervals; the maximum number of evaluations is $150 \cdot 10^3$.

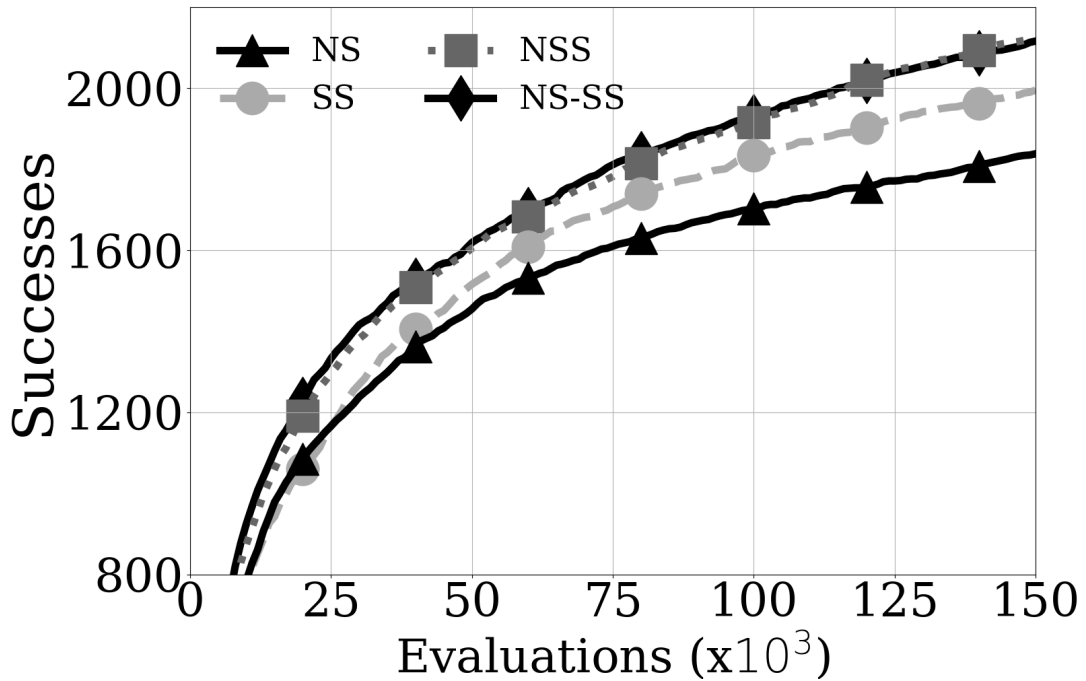


Figure 5.7: **Robustness:** algorithm successes in solving all the generated mazes over the number of evaluations for each considered method.

Table 5.2: **Algorithms tournament:** Percentage of 60 generated mazes for which the algorithm in a row has a strictly greater (≥ 1) number of successes compared to the algorithm in a column. Last row and last column are respectively the average of each column and the average of each row.

	NS	SS	NSS	NS-SS	Average
NS	–	8.3%	35.0%	36.6%	26.6%
SS	43.3%	–	38.3%	40.0%	40.5%
NSS	41.6%	38.3%	–%	21.6%	33.8%
NS-SS	41.6%	38.3%	30.0%	–	36.6%
Average	42.2%	28.3%	34.4%	32.7%	–

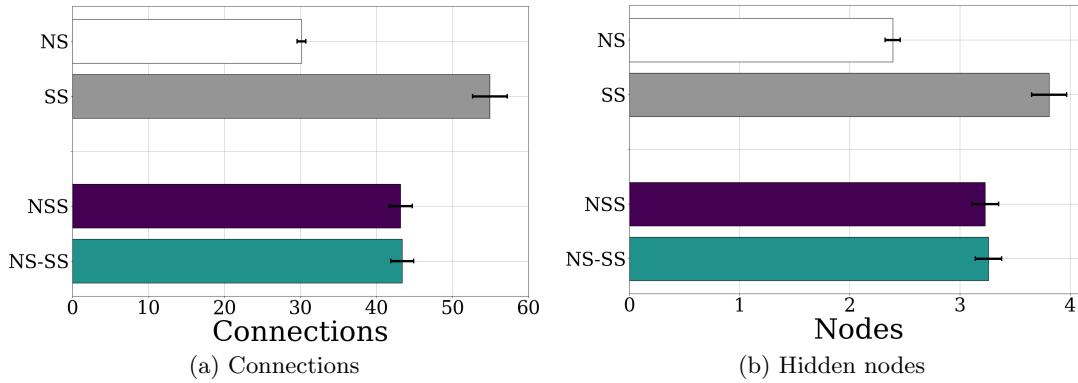


Figure 5.8: **Complexity:** number of connections and hidden nodes on average for evolved ANNs which solve the mazes per approach. Error bars denote 95% confidence intervals.

5.2 Soft Robot Testbed

In Chapter 4 we have shown that in the virtual creature evolution domain novelty search and surprise search obtain comparable performance across different test cases, but we obtained several insights on how their search processes differ: SS explores more extensively the space of possible soft morphologies, while NS performs slightly better on the higher resolutions. Based on the promising findings of the previous section and the results reported in Chapter 4, our hypothesis is that coupling novelty and surprise is a necessary condition for discovering even more highly-performing and unconventional solutions in the search space of virtual creatures. In order to test our hypothesis, this section evaluates how novelty-surprise search and novelty search-surprise search perform compared to their base components (i.e., novelty search and surprise search) in the domain of soft robot evolution in terms of efficiency and robustness. The performance of these divergent search approaches are compared based on the distance travelled by the evolved robots, as shown in Fig. 4.11a. More details can be found in Chapter 2 and Chapter 4. All algorithms in this section are tested across eight different soft robot setups, with varying lattice resolution, allowing for a comprehensive assessment of their efficiency and robustness.

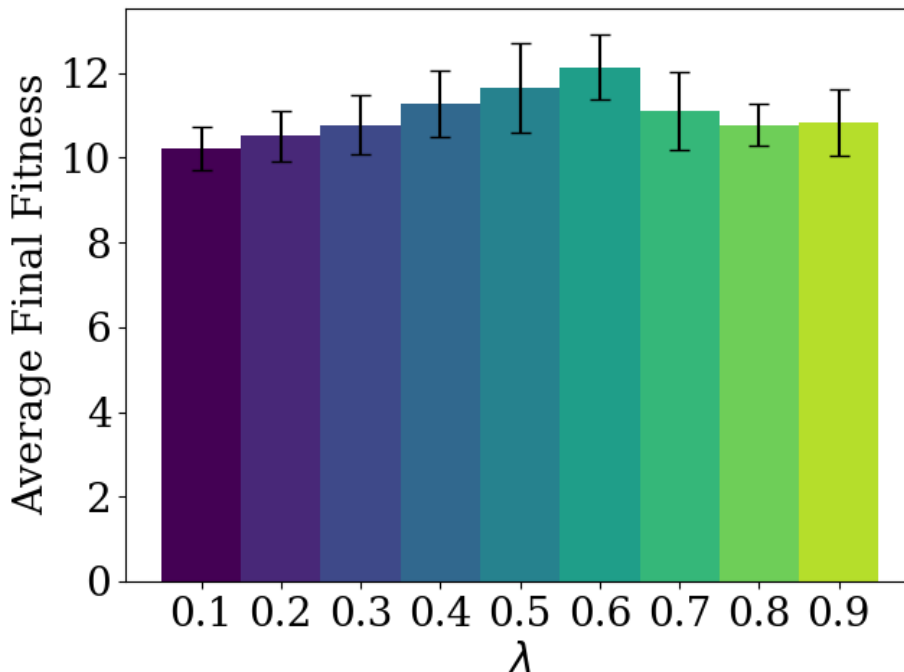


Figure 5.9: **Sensitivity analysis of λ** . The figure depicts the final average fitness of the fittest individuals obtained from 90 runs across nine λ values in the resolution $5 \times 5 \times 5$. Error bars display the 95% confidence interval of the average shown.

5.2.1 Algorithms

Similarly to Chapter 4, in this section we evaluate the outcomes of soft robot evolution in terms of efficiency, robustness and structural diversity. The goal is to assess how coupling two divergent approaches can affect the quality of the outcome and investigate the emerging differences between the robot structures favored by each EC approach. To compare the algorithms in terms of efficiency and robustness, the fittest individuals (based on the characterization of Fig. 4.11a) in each of 90 independent runs are collected. Further, the structural diversity of the obtained robots is analyzed based on the 90 populations evolved by each algorithm, to test their ability to explore different morphologies throughout evolution.

Novelty-Surprise Search

Novelty-surprise search (NSS) linearly combines the novelty and surprise scores as in Eq. (3.4). While the specific parameters of novelty search and surprise search remain unchanged (as reported above), the linear combination of novelty and surprise hinges on the λ parameter that controls the relative importance of the two rewards. In order to select the appropriate λ parameter, we run 9 experiments with λ ranging from 0.1 to 0.9. Each experiment is composed of 20 runs of the NSS algorithm with a particular λ parameter. Fig. 5.9 shows the average number of body length covered by the fittest individual across λ , for a representative resolution of $5 \times 5 \times 5$, chosen as it has already featured in previous work (Cheney et al., 2013; Gravina et al., 2017b). We pick a λ that yields the highest average performance after 1000 generations: this happens for $\lambda = 0.6$, which leads to an average of 12.13 body lengths.

Novelty Search-Surprise Search

Novelty search-surprise search (NS-SS) combines novelty search and surprise search in a multi-objective fashion. For the soft robot domain, we use the NSGA-II multi-objective algorithm by Deb et al. (2002) to simultaneously reward the novelty (Eq. 2.3) and unexpectedness of the behaviors shown by the evolved morphologies (Eq. 3.2). Unlike NSS, this implementation doesn't need any more parameters, while the parameters related to novelty and surprise remain the same described previously.

Baselines

Novelty search uses the parameters of (Methenitis et al., 2015); the novelty score is computed as the average distance of 10 nearest neighbors, i.e. $n_{NS} = 10$ in Eq. (2.3), using Eq. (4.1) for d_{NS} . Novelty search makes use of a novelty archive, where the most novel individuals in each generation are stored. *Surprise search* relies on a prediction model and a distance function (d_{SS}). The surprise score is computed as the Euclidean distance between the individual's trajectory and the four closest predicted trajectories ($n_{SS} = 4$). The predicted trajectories are computed by using linear regression of the sampled points of the previous two generations' trajectories. The local behaviours are computed via the k-means clustering algorithm, where $k_{SS} = 15$. More details can be found in Section 4.2.2.

5.2.2 Experiments and Analysis

While in the previous chapter we explored the effectiveness of divergent search against objective (fitness-based) search, in this section we want to assess the performance of the combination of novelty and surprise in the soft robot environment. Therefore, we focus only on four algorithms—novelty-surprise search, novelty search-surprise search, novelty search and surprise search—across eight different lattice sizes.

All reported results are obtained from 90 independent evolutionary runs; reported significance is at a 95% confidence. For multiple pairwise comparisons the Tukey's range test is used to establish significance. In the experiments conducted we use the same parameters used in Methenitis et al. (2015) and in Chapter 4: a gravity of -27.6 m/s^2 , a simulation time of 0.4 seconds, a rate of 40 Hz for the signal that actuates active voxels and a sampling rate of 100 Hz. Eight different lattice resolutions, from 3^3 to 10^3 , are employed. For all the tested approaches, the population has 30 individuals, and the number of generations is equal to 1000. The selection mechanism used in this domain is generational; other CPPN-NEAT parameters are described in Appendix A.

Efficiency and Robustness

The main goal of robot locomotion is to evolve efficient behaviors, i.e., robots that reach the most distant point at the end of the simulation (based on the characterization of Fig. 4.11a). This section focuses exclusively on the 90 fittest individuals¹ collected from 90 independent runs across 8 resolutions. All values are normalized to the dimension of the 3D lattice (i.e., in body lengths of the robot).

Results from Table 5.3 show that, in terms of efficiency, NS-SS and NSS outperforms any other approach for every resolution selected. In particular, NSS significantly outperforms novelty search in 5 of the 8 resolutions tested (3^3 , 4^3 , 6^3 , 8^3 , and 10^3) and surprise search

¹The fittest individual of each independent evolutionary run is selected.

Table 5.3: **Efficiency**: distance covered on average by the fittest individuals collected from 90 independent runs for each method (95% confidence interval in parentheses). Bold values are significantly different from all the other approaches.

	NS	SS	NSS	NS-SS
3x3x3	8.88 (0.19)	9.32 (0.26)	9.67 (0.23)	9.89 (0.23)
4x4x4	9.89 (0.35)	10.71 (0.34)	11.18 (0.36)	11.71 (0.44)
5x5x5	11.13 (0.33)	10.73 (0.37)	11.28 (0.30)	11.73 (0.35)
6x6x6	11.15 (0.28)	11.43 (0.48)	11.72 (0.30)	11.68 (0.35)
7x7x7	11.03 (0.37)	10.57 (0.32)	11.35 (0.32)	11.99 (0.34)
8x8x8	11.47 (0.39)	11.35 (0.38)	12.36 (0.41)	12.56 (0.45)
9x9x9	11.48 (0.37)	11.08 (0.32)	11.53 (0.38)	12.61 (0.45)
10x10x10	11.32 (0.38)	11.18 (0.43)	12.05 (0.35)	12.60 (0.63)

in 4 of the resolutions tested (5^3 , 7^3 , 8^3 and 10^3). On the other hand, NS-SS outperforms significantly novelty and surprise search for 7 of the 8 resolutions considered (3^3 , 4^3 , 5^3 , 7^3 , 8^3 , 9^3 and 10^3). If we compare NSS and NS-SS, we can notice that overall the multi-objective approach is more efficient in 7 out of 8 resolutions, but the difference is significant only for two resolutions, i.e., 7^3 and 9^3 . Fig. 5.10 shows the results of efficiency across all the resolutions by means of linear regression. We can notice that every approach has a linear relationship between their final average efficiency and the robots' resolution, as their final average efficiency is highly correlated with lattice size ($r > 0.7$, $p < 0.05$ for each method except SS, where $r = 0.7$ and $p = 0.051$). As noted in Table 5.3 and observing the trends of the regression lines, we can notice that NSS and NS-SS constantly achieves better results compared to the baselines. The intercept values of NSS and NS-SS are significantly different from the two baselines based on an ANCOVA test ($p < 0.05$). If we focus on the two new approaches, we can notice that NS-SS is performing constantly better compared to NSS, but not significantly.

As in Chapter 4, we investigate the *robustness* of each algorithm defined as the number of robots able to cover a distance greater than the selected threshold (in body lengths). Fig. 5.11 shows the distribution of successes across different thresholds, cumulated across all lattice resolutions (i.e., 90 fittest robots for each of 8 resolutions). The distribution shows that generally NSS obtains more successes than the baselines in thresholds between 8 and 14. Furthermore, NS-SS achieves more successes for the higher thresholds, in particular between 10 and 15. The robustness of novelty search and surprise search, on the other hand, is comparable across all shown thresholds, as noticed in Chapter 4.

Structural Variety

In order to evaluate the variety of morphologies explored by the four approaches, we perform an analysis of the evolved structures. In particular, we collect the individuals evolved in one run sampled every 10 generations and we add the individuals to the bin if and only if it is empty or the fitness is lower. We compute these feature maps across the 8 resolutions considered, and the results are averaged from 90 independent runs per lattice resolution. The two feature dimensions are the same as in (Mouret and Clune, 2015): the percentage of the voxels filled (x -axis), and the percentage of blue stiff voxels, i.e., bones (y -axis). To perform a fair comparison with the results of the previous chapter, the feature maps have a resolution of 27×27 .

Fig. 5.12 shows the number of explored bins averaged across 90 runs for each of the

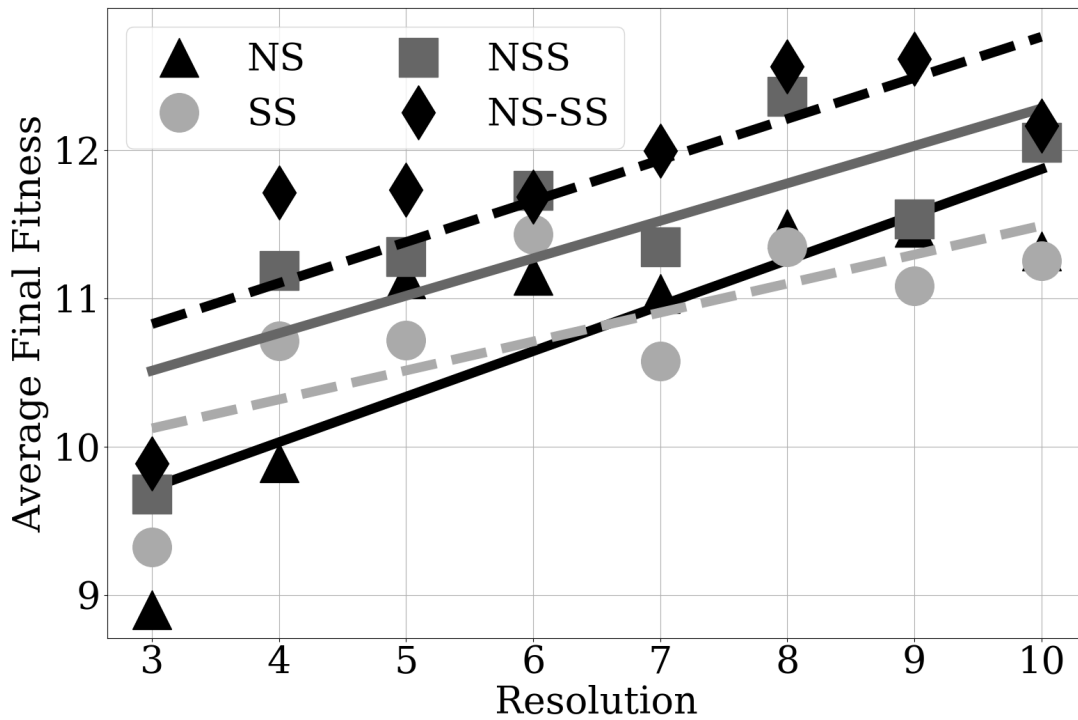


Figure 5.10: Relation between the resolution of the robots and the maximum fitness of each approach (averaged from 90 runs).

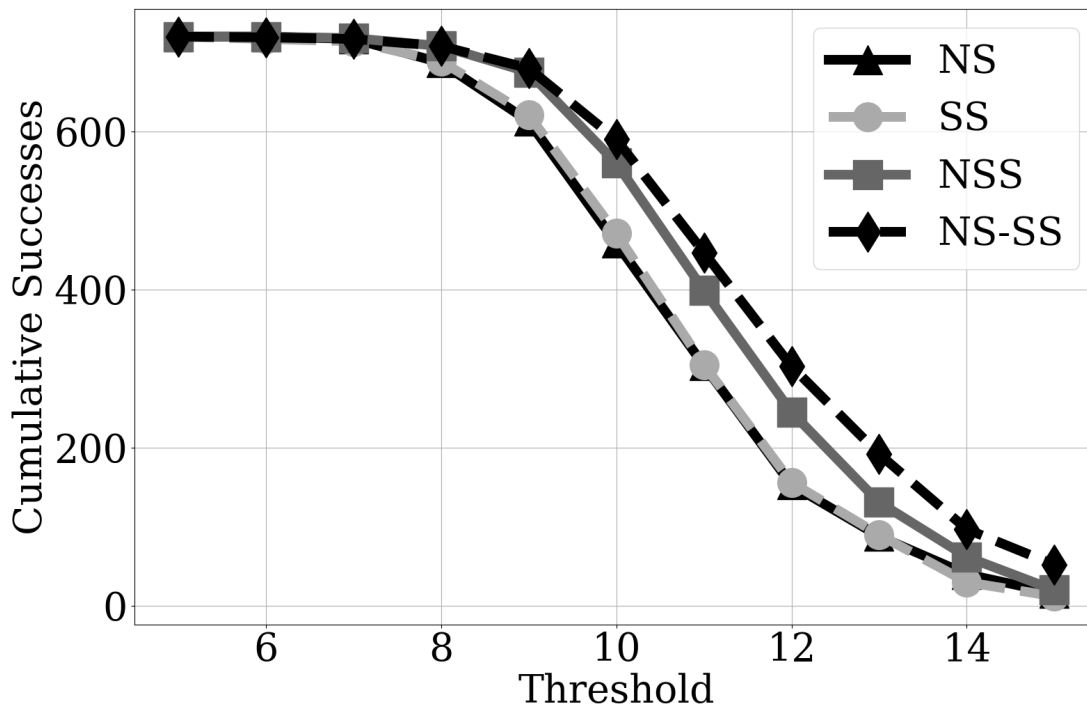


Figure 5.11: **Robustness**: number of successes, cumulated on all resolutions, for different performance thresholds.

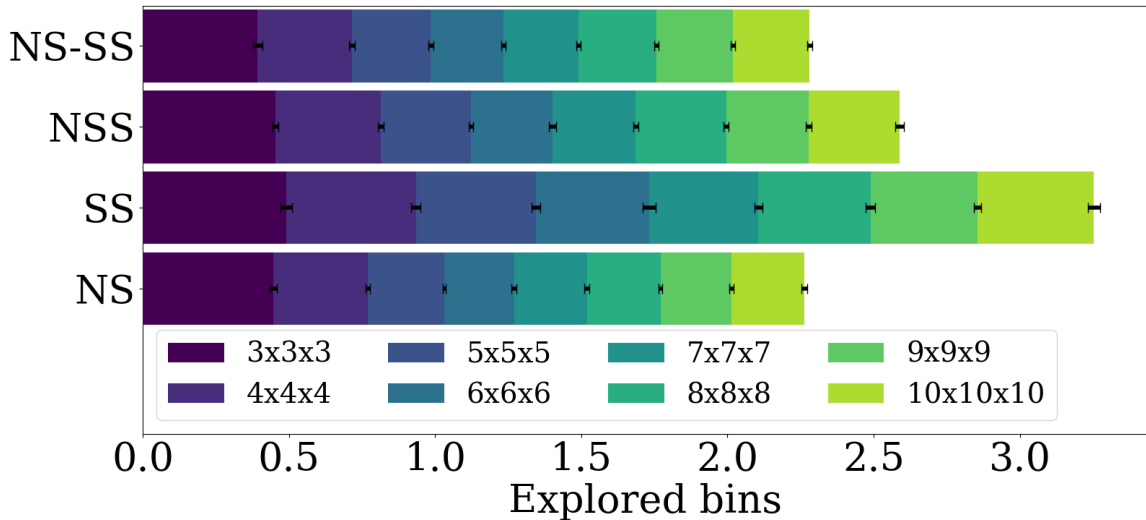


Figure 5.12: **Structural variety**: average cumulative number of explored bins for all feature maps. Each bar is normalized by the maximum number of possible bins and error bars display the 95% confidence interval of the average shown.

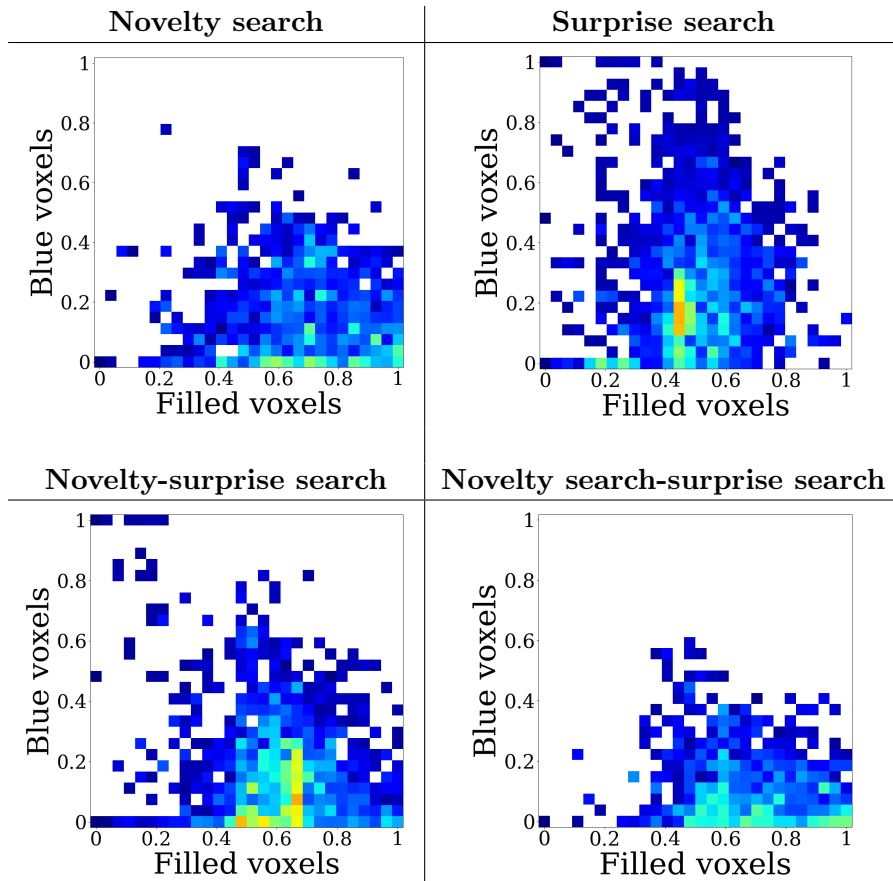
eight lattice resolutions. Surprise search shows better exploratory capacity compared to all the other divergent approaches. Interestingly, novelty search tends to explore fewer bins in the two feature dimensions considered, while NSS, NS-SS, and especially surprise search are able to explore more broadly. Regarding the two approaches that combine novelty and surprise, NSS and NS-SS structures lie between these two “extremes”. This algorithmic property seems to be beneficial for divergent search in terms of performance (in terms of efficiency and robustness). Indeed, NSS and NS-SS find more bins in the feature space chosen compared to novelty search (significantly in 7 out of 8 resolutions and 1 out of 8 resolutions for NSS and NS-SS respectively). However, surprise search finds significantly more bins in the feature space chosen compared to NSS and NS-SS in all lattice resolutions ($p < 0.05$). We show a sampled run for each approach in Table 5.4; samples taken from all the 8 resolutions are reported in Appendix C.

The example robots shown in Fig. 5.13 attest to the variety of forms which can be well-performing while structurally different. Further, the figure shows four frames of simulation per robot that illustrate the variance in the way the different evolved morphologies move away from their starting point.

5.3 Discussion

In this chapter, we have shown that by coupling novelty and surprise search we can obtain two algorithms that are faster and more robust than novelty search or surprise search alone in the maze navigation and soft robot domain. In the experiments of Section 5.1.1 we have compared two new divergent algorithms in various maze setups. The reported results show that coupling surprise and novelty improves the algorithm performance more than its constituent parts, and it proves advantageous with regards to efficiency and robustness. The first algorithm, NSS, has shown improved performance in the four authored mazes, especially in the two hardest ones. Moreover, the analysis in the procedurally generated mazes shows that NSS is more effective on average across different degrees of deceptiveness. While these

Table 5.4: **Feature maps**: feature maps produced by the four methods, by aggregating all the individuals evolved on a resolution of $4 \times 4 \times 4$. White bins do not have any robots, while colored bins denote the fitness of the best individual (blue for low fitness, red for high fitness).



results already demonstrate the advantages of NSS as a combined form of divergent search, even better results have been obtained by employing a multi-objective approach. As NSS couples novelty and surprise linearly, it cannot exploit various combinations of novelty and surprise in a dynamic fashion. This static behavior of NSS limits the capabilities of the algorithm. This probably explains why NS-SS shows better results when navigating the hardest authored mazes (i.e., very hard maze and extremely hard maze). However, in the second experiment with 60 generated mazes, the performance of the two implementations is comparable: this is probably due to the relatively low deceptiveness of the majority of the randomly generated mazes, which does not highlight the differences between these two implementations.

In the second domain employed, the objective was to test the performance of the two introduced algorithms, NSS and NS-SS, in the soft robot evolution domain. In this section, we methodically compare the performance of soft robot evolution across different lattice resolutions. Admittedly, the main motivation for this analysis is to test how sensitive each of the divergent search approaches is to the granularity allowed per morphology. Overall, both the proposed implementations have shown improvements in performance both in terms of efficiency and robustness. Results show that both NSS and NS-SS are consistently

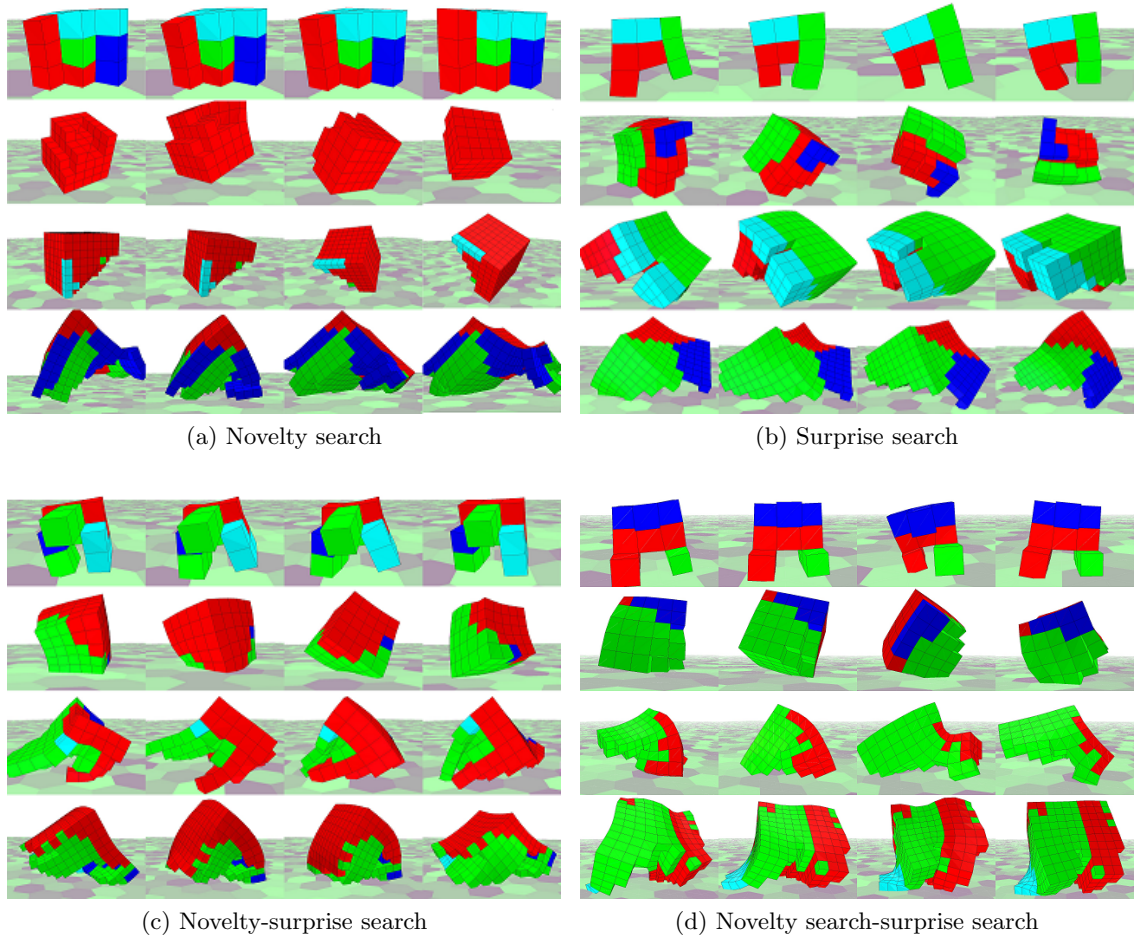


Figure 5.13: Fittest robots evolved in the first run of each approach for the resolutions 3x3x3, 5x5x5, 8x8x8 and 10x10x10 (from top to bottom). Four simulation frames are depicted for each robot.

more efficient than the other algorithms when considering all eight resolutions tested as a whole (via ANCOVA tests), showing that the algorithm can scale to more or less complex problems. Notably, larger lattices generally lead to more efficient behaviors, with robots of 27 voxels reaching 20% shorter distances (normalized to robots' body lengths) than robots with 1000 voxels. This is perhaps not surprising, as a higher resolution allows for more expressive morphologies (more voxels to choose from) and more robust behaviors. While the higher resolutions would challenge a direct representation, the indirect encoding (via CPPNs) can scale and perform better at higher resolutions for all EC methods tested. Like maze navigation, NS-SS shows better search capabilities compared to the linear aggregation solution. In particular it seems that using a multi-objective approach leads to better efficiency and robustness in 7 out of 8 resolutions; however, the difference is not significant enough to draw definitive conclusions and more analysis is required.

Based on the evidence collected, we can argue that combining two different yet powerful ways of divergent search ends up being beneficial with respect to search. Both NSS and NS-SS algorithms search for novel solutions, which means exploring not yet seen points in the search space, and at the same time also rewards surprising solutions, which means devi-

ating from predicted behavioral trends. The working hypothesis is that novelty search and surprise search give orthogonal rewards; their combination should benefit divergent search, which seems to be confirmed by the evidence shown in this chapter. It seems that the combination of novelty and unexpectedness results in a deeper exploration of the search space, as a greedy search only for novel behavior might “hide” less novel but efficient behaviors. Combining novelty with surprise alleviates that, as surprise may backtrack to previously seen behaviors. On the one hand, when novelty search finds a highly novel solution, this will likely be still considered novel for several generations until it gains sufficient neighbors. On the other hand, surprise search will consider a solution surprising only in the first generation it appears, as the prediction model will change in the following generations; this has the drawback that a really good solution in terms of global fitness can be eliminated by surprise search because it is not surprising anymore. By aggregating the novelty and surprise scores, we eliminate both of these drawbacks: novelty is able to “maintain” surprising solutions from evolutionary rejection (if they also have a high novelty score) and, at the same time, very novel solutions have less impact on the selection mechanism as their surprise score degrades throughout evolution. As seen in the behaviour analysis of the testbed navigation task, the coupling is beneficial as it pushes robot controllers to spread out and explore the behavioural space more extensively. We can draw similar conclusions in the soft robot domain. Through the in-depth analysis of the evolved robots’ structural characteristics, we have shown that surprise search helps the search algorithm to explore more extensively the structural space, while novelty search explores less but find more efficient morphologies. Combining these two processes allow to explore more efficiently the search space and improve the overall performance of the algorithm.

5.4 Summary

This chapter has introduced two new divergent algorithms which couple novelty and surprise search and it tested their performance in the maze navigation and soft-robots domain. In particular, we started our investigation from the well-studied maze navigation domain and we performed two separated experiments. In the first experiment, we tested the performance of NSS and NS-SS in four human-designed mazes introduced in the previous chapter. Subsequently, we conducted a second experiment where we compared the efficiency and the robustness of NSS and NS-SS against 60 randomly generated mazes. In the third set of experiments, finally, we explored how combining novelty and surprise affects soft robot evolution, both in terms of performance and variety of evolved structures. Extensive experiments which vary the impact between novelty and surprise (λ) for the linear aggregations, a multi-objective implementation and vary the number of voxels available for the robot showed that the combined search for novel and surprising solutions is advantageous. Evidently, combining novelty and surprise search allows search to eliminate, in part, limitations inherent in novelty search and surprise search, as it outperforms the performance of the two algorithms when tested on their own. The orthogonality of novelty and surprise is evidenced by the fact that combining the two in one algorithm evidently results in improved robustness and efficiency. Therefore we can imagine that coupling surprise and novelty with other objectives might be even more advantageous. Towards that end, in the next chapter, we test three novel extensions of surprise search in the context of quality diversity optimization.

Chapter 6

Surprise for Quality Diversity: Experiments

Motivated by the promising results obtained by surprise search on its own (Chapter 4) and coupled with other approaches (Chapter 5), in this chapter we investigate how surprise can be combined with other objectives besides novelty and test these new approaches as quality diversity algorithms. There is limited exploration on how different paradigms of divergent search may impact the solutions found by quality diversity. To cover this gap, this chapter investigates the impact of surprise to quality diversity performance. In Section 6.1 we describe the domain employed and the implementation details of the tested algorithms. We then introduce a new challenging set of 60 deceptive mazes, specifically selected to be deceptive and hard to solve. To complete the performance overview of surprise-based QD, in Section 6.2 we perform a second set of experiments in other 60 automatically generated mazes and four authored mazes, both introduced in Chapter 4. This chapter finally concludes with an analysis of the obtained results (Section 6.3).

6.1 Maze Navigation Testbed: First Set

As noted previously, the maze navigation problem is interesting and effective domain to test divergent and quality diversity approaches. The problem of maze navigation is the identical to the one presented in Chapter 2: a wheeled robot starting at a specific position in the maze must reach the goal position in a maze using incomplete information given by sensors. More details can be found in Chapter 2 and Chapter 4. In the following subsection we describe the implementation details of the surprise-based QD approaches in this domain.

6.1.1 Algorithms

In this chapter, genetic operators, mutation chances and speciation parameters are identical to those reported in Lehman and Stanley (2011a) and used in Chapter 4 and Chapter 5; parameters of novelty search are described in Section 6.1.2. This subsection describes the implementation details of the algorithms which are compared in subsection 6.1.3 to test our hypothesis that surprise search enriches the capacity of QD.

Baselines

Objective Search: As an indication of how deceptive landscapes can hinder traditional EC approaches, objective search is used as a baseline for selecting among generated mazes in Section 6.1.2. As per Lehman and Stanley (2011a), the objective f is to minimize the Euclidean distance between the goal in the maze and the robot’s final position at the end of simulation. Objective search never finds a solution in any of the mazes used in the experiments of Section 6.1.2, so its results are omitted.

Novelty Search: As discussed in Chapter 2 and Chapter 4, novelty search ignores the objective of the problem at hand and attempts to maximize behavioral diversity. The novelty score, computed through Eq. (2.3), uses the behavioral distance d_{NS} between two individuals; in this domain d_{NS} is the Euclidean distance between the two robots’ final positions at the end of simulation. The same distance is used to identify closest neighbors (μ_j). Finally, in all novelty search experiments in this section we follow the literature and consider the 15 closest individuals as our n_{NS} parameter as in (Lehman and Stanley, 2011b), Chapter 4 and Chapter 5.

Novelty Search with Local Competition

As noted in Chapter 2, novelty search with local competition combines the divergence of novelty search with the localized convergence obtained through a local competition (Lehman and Stanley, 2011b). In this chapter, NS-LC uses a steady-state NSGA-II multi-objective algorithm (Li et al., 2017) to find non-dominated solutions on two dimensions: novelty and local competition. Novelty attempts to maximize a novelty score computed in Eq. (2.3), using the Euclidean distance between the two robots’ final positions at the end of simulation for calculating distance from the closest individuals. As with novelty search, we are based on the successful experiments performed in the literature (Lehman and Stanley, 2011b) and consider the $n_{NS} = 15$ closest individuals in all NS-LC experiments. Local competition is calculated based on the closest individuals (in terms of Euclidean distance) in the current population and the novelty archive. In the maze navigation task, local competition counts the number of neighboring robots with final positions that are farther from the goal in terms of Euclidean distance. Note that unlike earlier work (Lehman and Stanley, 2011b), this chapter decouples the number of individuals considered for the novelty score (n_{NS}) with the number of individuals used to calculate local competition (n_{LC}). In this section we perform a sensitivity analysis for the best locality parameter (n_{LC}) of local competition in NS-LC and the other QD algorithms.

Surprise Search with Local Competition

As noted in Chapter 3, surprise search with local competition uses a steady-state NSGA-II multi-objective algorithm (Li et al., 2017) to find non-dominated solutions on two dimensions: surprise and local competition. Surprise uses the prediction model of Eq. (3.1) to make a number of predictions (\mathbf{p}) for the expected behaviors in the current population. The individuals in the current population are then evaluated based on their distance from the n closest predictions as per Eq. (3.2). In this testbed, predictions are made on the robots’ final position, and distance refers to the Euclidean distance between two robots’ final positions at the end of simulation. For deriving k_{SS} behaviors to predict, k-means clustering is applied

on the robots’ final position in one generation. Based on earlier results on surprise search and novelty-surprise search on difficult maze navigation tasks (see previous Chapter 4 and Chapter 5), $k_{SS} = 200$ and $n_{SS} = 2$. The two last generations are used ($h = 2$) to cluster behaviors, and predictions are based on a linear interpolation between cluster centroids in subsequent generations (Fig. 4.3 illustrates how behaviors are clustered and how predictions are computed). More details on the way clustering is performed and predictions are made for surprise search in maze navigation can be found in Chapter 3.

Novelty-Surprise Search with Local Competition

As noted in Chapter 3, novelty-surprise search with local competition uses a steady-state NSGA-II multi-objective algorithm (Li et al., 2017) to search for non-dominated solutions on the dimensions of local competition (computed in the same way as NS-LC) and a weighted sum of the novelty score and surprise score as in Eq. (3.3). Parameters for novelty search are the same as in (Lehman and Stanley, 2011b) (i.e., $n_{NS} = 15$), while parameters for surprise search are the same as above (i.e., $h = 2$, $n_{SS} = 2$, $k_{SS} = 200$). Distance characterizations for NSS-LC are computed as in the other QD algorithms, via the Euclidean distance between the two robots’ final positions at the end of simulation. A sensitivity analysis is performed in this section to identify which values for λ and n_{LC} (the neighbors considered for local competition) are most appropriate for this testbed.

Novelty Search–Surprise Search–Local Competition

As noted in Chapter 3, NS-SS-LC uses a steady-state NSGA-II multi-objective algorithm (Li et al., 2017) to search for non-dominated solutions on three dimensions: local competition (computed in the same way as NS-LC), novelty search, and surprise search. Parameters for novelty search are the same as in (Lehman and Stanley, 2011b) (i.e., $n_{NS} = 15$), while parameters for surprise search are the same as above (i.e., $h = 2$, $n_{SS} = 2$, $k_{SS} = 200$). Distance characterizations for NS-SS-LC are computed as in the other QD algorithms, via the Euclidean distance between the two robots’ final positions at the end of simulation. A sensitivity analysis is performed in this section to identify the best n_{LC} values for NS-SS-LC in this testbed.

6.1.2 First Set of Generated Mazes

While a substantial portion of research on divergent search and quality diversity have focused on hand-crafted deceptive mazes (Lehman and Stanley, 2011a, 2010; Pugh et al., 2016), this section uses a broader set of mazes to test the QD algorithms introduced in Chapter 3. These mazes are not crafted by a human designer but generated procedurally; a similar set of generated mazes has been used in (Lehman and Stanley, 2011d), Chapter 4 and Chapter 5 to evaluate the performance of divergent search algorithms. Generating rather than hand-crafting mazes allows for a broader range of spatial arrangements to be tested, without human curation towards possibly favorable patterns. Additionally, this chapter uses maze generation to find appropriate deceptive mazes which satisfy two criteria related to the algorithms tested rather than inherent structural patterns. These criteria are:

1. Objective search must not find any solutions in 50 evolutionary runs.
2. NS-LC (with parameters as in (Lehman and Stanley, 2011b)) must find a solution in at least one of 50 evolutionary runs.

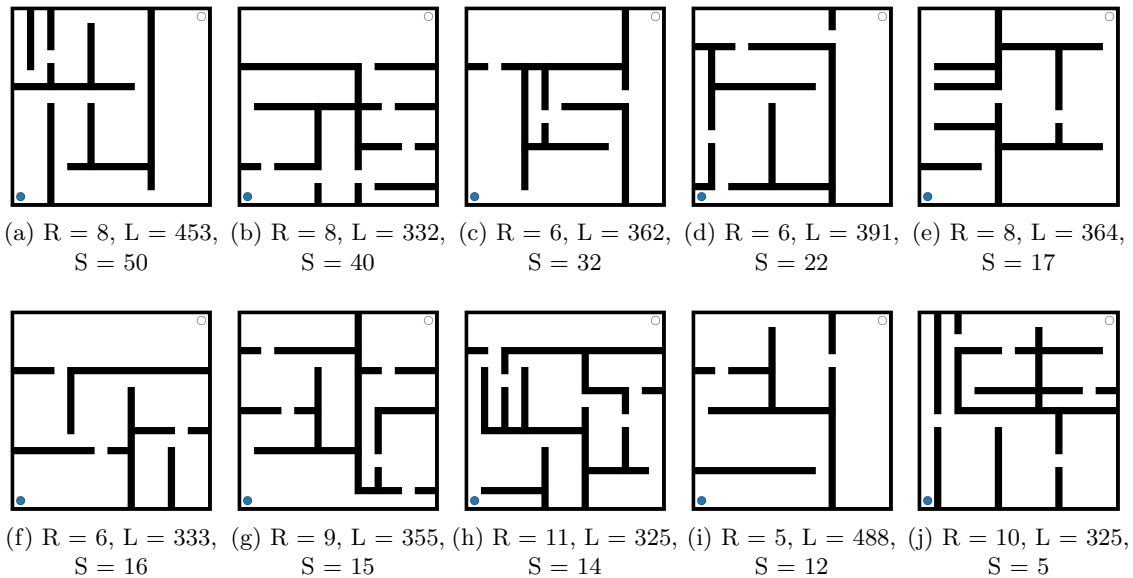


Figure 6.1: **Maze generation:** 10 easiest generated mazes created via recursive division, sorted by NS-LC successes. The starting position (blue filled circle) is at the bottom left corner; the goal position (black empty circle) is at the top right corner. In the caption, R is the number of subdivisions, L is the A^* length and S is the number of successes of NS-LC ($n_{LC} = 15$).

Table 6.1: **Distribution of the 60 selected mazes** per number of subdivisions and corresponding average length of the shortest path computed with the A^* pathfinding algorithm.

Subdivisions	5	6	7	8	9	10	11	12
# Mazes	2	9	5	16	13	10	4	1
A^* length	403	350	358	353	330	318	323	291

The first criterion establishes that each maze is deceptive, as attempting to optimize proximity to the goal does not result in any solutions. However, the first criterion may be satisfied by mazes which have extremely long paths from start to finish which may not be reachable within the allotted simulation time for each robot. To ensure that the maze is not too difficult (or practically impossible), the state-of-the-art QD algorithm NS-LC is used as a second criterion: if “vanilla” NS-LC (Lehman and Stanley, 2011b) cannot find a solution even after numerous retries, then the maze is characterized as too difficult and is ignored.

As for the randomly generated mazes in Chapter 4 and Chapter 5, mazes are generated via a recursive division algorithm (Reynolds, 2010), which subdivides the maze (by adding walls at the border) recursively until no more walls can be added. The algorithm stops after a specific number of subdivisions, or when adding a new wall would make the path non-traversable. The start position is always set on the bottom-left corner and the goal position on the top-right corner. Width of gaps and the minimum width of corridors is defined in a way that allows the robot controller to comfortably pass through. All mazes tested have between 5 and 12 subdivisions (chosen randomly), and evolution for both objective search and NS-LC is performed on a population of 250 individuals for a maximum of 600 generations and a simulation time of 300 frames.

Through the above process, more than 800 mazes were generated and tested but only 60 mazes were found to satisfy the two criteria. The 60 mazes used in this chapter are available in Appendix B. Details of the 60 mazes’ properties are described in Table 6.1, including the actual distance between the start and goal position based on A* pathfinding. It is immediately obvious that most mazes chosen have between 8 and 10 subdivisions, and that the A* distance decreases as the subdivisions increase. Indeed, there is a significant ($p < 0.01$) negative correlation between the number of subdivisions and the A* distance (-0.39). This is likely due to the fact that with more subdivisions the likelihood that a random maze would not be solvable even by NS-LC increases; for instance due to narrower corridors and more complex structures. Through an informal assessment of the many mazes generated, mazes with fewer subdivisions tend to fail the first criterion while mazes with more subdivisions tend to fail the second criterion.

While the criterion of at least one success in 50 evolutionary runs for NS-LC is satisfied in all 60 mazes in our test set, it is worthwhile to investigate how many successes are actually scored by NS-LC per maze. The number of evolutionary runs (out of 50) in which NS-LC finds a solution is indicative of the hardness of the problem, and will be used in subsection 6.1.3 as an important performance metric for comparing QD algorithms. Fig. 6.1 shows the 10 easiest mazes, where NS-LC found the most solutions in 50 runs. Among those 10 mazes, the average number of successes was 22.3 (95% CI = 10.03) while in all 60 mazes this was 5.93 (95% CI = 2.45). More broadly, the number of successes per maze ranged from 50 out of 50 (in 1 maze) to 1 out of 50 (in 20 mazes). There seems to be a significant ($p < 0.05$) correlation between NS-LC successes and A* path length (0.32). There is a weak negative correlation between NS-LC successes and number of subdivisions (-0.11) which is however not significant ($p > 0.05$); indeed, among the easiest mazes of Fig. 6.1 there is one with 11 subdivisions (Fig. 6.1h) as well as one with 5 subdivisions (Fig. 6.1i).

6.1.3 Experiments and Analysis

As discussed previously, the maze navigation problem is used to test four quality diversity algorithms: NS-LC, SS-LC, NSS-LC and NS-SS-LC. Sixty generated mazes are used to test each algorithm in 50 evolutionary runs per method per maze with the same parameters: a population size of 250 individuals, a maximum of 600 generations, and a simulation time of 300 time steps. The core performance measure we consider is the aggregated number of evaluations per successful run across all mazes per method. A successful run discovers one robot that can reach the goal position from the start position within the 600 generations allotted; when a solution is found, evolution immediately ends. It is important to mention that for those methods where multi-objective optimization is applied (e.g., NS-LC), the final front of solutions is not considered as a performance indicator. Significance reported for all experiments in this section is at a 95% confidence. For multiple pairwise comparisons the Tukey’s range test is used to establish significance.

In order to find appropriate values for the many parameters in each evolutionary method, a sensitivity analysis is performed on a subset of mazes and reported. Then, the best parameters are selected for each algorithm which in turn are compared on their best setup. Note that evolutionary runs reported in the sensitivity analysis are independent from those reported in the main comparison of QD algorithms.

Sensitivity Analysis

Due to a plethora of parameters that may impact the performance of QD algorithms examined in this section, we rely largely on successful parameter values reported in the literature for the baseline algorithms. However, we perform a sensitivity analysis along two parameters for which we could not find suggested values: the locality of local competition (n_{LC}) in all QD algorithms and the weight of novelty versus surprise (λ) in NSS-LC. Since most mazes in the test set are particularly difficult to solve, the impact of a parameter tweak is not expected to have a strong impact on the algorithm’s performance¹. Therefore, the 10 easiest mazes (based on “vanilla” NS-LC) shown in Fig. 6.1 are used for the sensitivity analysis. The core performance metric for sensitivity analysis is the average number of evaluations needed by each algorithm to discover a controller able to solve the maze. In a run where no solution was found within the allocated budget, the maximum allocated evaluations ($150 \cdot 10^3$) is used. This metric is averaged across 50 evolutionary runs.

Beyond testing the sensitivity of n_{LC} and λ , we assess how the various sub-components of the introduced algorithms (novelty, surprise and the linear combination of novelty and surprise) affect algorithmic performance with an additional set of baseline algorithms described. In the sensitivity analysis we report results based on 50 evolutionary runs per maze; the experiments used in this analysis are independent from those in the subsequent main comparison of the QD approaches. Furthermore, we run separate and independent runs in the two sensitivity analysis, i.e., 50 runs per maze in the first analysis of the n_{LC} and λ parameters, and other 50 runs per maze in the sub-components analysis.

Sensitivity to Parameters: the parameter setups range from a n_{LC} value between 5 and 20 (increments of 5) and λ for NSS-LC between 0.4 and 0.8 (increments of 0.1). The results of this analysis in terms of average number of evaluations across all runs in all 10 easiest mazes are shown in Fig. 6.2.

The results of the sensitivity analysis show that both parameters have an impact on the performance of both SS-LC and NSS-LC (where applicable). Notably, however, n_{LC} seems to have very limited impact on the performance of NS-LC, with marginally better performance obtained with $n_{LC} = 5$. The same parameter impacts SS-LC in a much more pronounced manner, with $n_{LC} = 5$ and $n_{LC} = 15$ giving a far worse performance for SS-LC than other values tested. The best performance for SS-LC is for $n_{LC} = 10$, which however requires more evaluations than NS-LC. NS-SS-LC tends to fluctuate in the same way as NS-LC, although it seems to fall behind for $n_{LC} = 15$. Finally, NSS-LC hinges on both λ and n_{LC} and it is clear from Fig. 6.2 that good performance can be achieved for several combinations of these two parameters. However, the least evaluations are achieved with a high λ value (see Table 6.2), in which case the novelty score contributes more than the surprise score. Based on this analysis, the parameters for the remaining experiments (sub-components analysis and the final comparison of the best setup for the QD algorithms at test) are shown in Table 6.2.

Sensitivity to Algorithmic Components: apart from the many different parameters in the proposed QD algorithms, there are several design decisions in the implementation of each. Based on the core components of algorithms considered (novelty, surprise, local competition), we will compare the four QD algorithms (NS-LC, SS-LC, NSS-LC and NS-

¹For example, in a maze where NS-LC has only one success in 50 runs, the impact of any parameter is expected to be a product of chance.

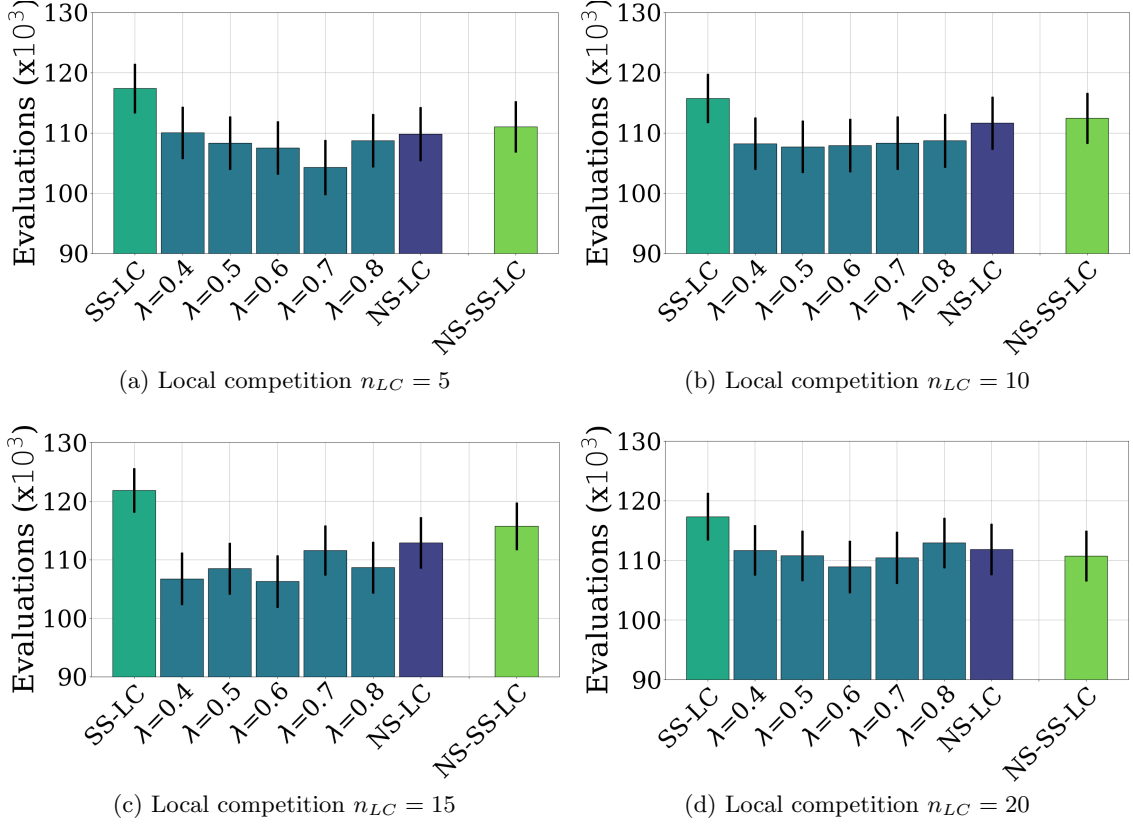


Figure 6.2: **Sensitivity analysis:** Average number of evaluations across the 10 easiest mazes with four local competition sizes (5, 10, 15, 20), with five values of λ (0.4, 0.5, 0.6, 0.7, 0.8).

Table 6.2: **Final parameters for QD algorithms**, based on a sensitivity analysis. Shown is the average number of evaluations (and 95% confidence intervals) across the 10 easiest mazes of Fig. 6.1.

Algorithm	Parameters	Evaluations ($\cdot 10^3$)
NS-LC	$n_{LC} = 5, n_{NS} = 15$	109.8 ± 4.4
SS-LC	$n_{LC} = 10, n_{SS} = 2,$ $h = 2, k_{SS} = 200$	115.7 ± 4.0
NSS-LC	$\lambda = 0.7, n_{LC} = 5,$ $n_{NS} = 15,$ $n_{SS} = 2, h = 2, k_{SS} = 200$	104.2 ± 4.5
NS-SS-LC	$n_{LC} = 5, n_{NS} = 15,$ $n_{SS} = 2,$ $h = 2, k_{SS} = 200$	111.0 ± 4.2

SS-LC) against novelty search (described in subsection 6.1.1) and four more baselines:

- Surprise Search, described in Chapter 3. Surprise search uses the same predictive model as SS-LC, as described in subsection 6.1.1 and Fig. 4.3. The parameters of SS

(n_{SS}, h, k_{SS}) are the same as SS-LC in Table 6.2.

- Novelty-Surprise Search, which is a single objective implementation that linearly combines novelty and surprise (see Eq. 3.3) as described in Chapter 3. Performing the same analysis across λ values as done for NSS-LC, we use $\lambda = 0.4$ for NSS while n_{SS} , h , and k_{SS} are the same as NSS-LC in Table 6.2.
- Novelty Search–Surprise Search uses a steady-state NSGA-II multi-objective algorithm (Li et al. (2017)) to search for non-dominated solutions on the dimensions of novelty (Eq. 2.3) and surprise (Eq. 3.2). All parameters are the same as NS-SS-LC in Table 6.2.
- Surprise Search Archive with Local Competition (SSA-LC), which is a variant of SS-LC where the divergence objective of NSGA-II uses the surprise score but also maintains a novelty archive identical to how it is maintained in NS-LC Lehman and Stanley (2011b) (using the same fluctuating threshold and assessing individuals for insertion to the novelty archive based on the novelty score of Eq. 2.3). To calculate the surprise score, SSA-LC considers the nearest neighbors both in the prediction space and in the novelty archive. To calculate the local competition score, SSA-LC considers the nearest neighbors in the current population and in the novelty archive (similar to NS-LC). Results for the best performing n_{LC} parameter are reported ($n_{LC} = 5$), while other parameters of surprise search (n_{SS}, h, k_{SS}) are the same as SS-LC in Table 6.2.

The first three baseline algorithms test only the diversity dimensions of some of the proposed QD algorithms (NSS is the diversity dimension used in NSS-LC, while NS-SS is the two-objective diversity component of NS-SS-LC). SSA-LC tests another way of combining deviation from expected (through the prediction space) and seen (through the novelty archive) behaviors.

Fig. 6.3 shows the performance of all algorithms discussed in this section, separating QD approaches (on the left of the figure) from pure divergent search approaches (on the right). It is clear that using a local competition as a second objective significantly improves performance over a divergence-only variant (e.g., SS-LC versus SS, NSS-LC versus NSS). Notably, the NS-SS multi-objective divergent search approach performs surprisingly well, being the most efficient divergent search approach and requiring significantly fewer evaluations than both SS and NS. NS-SS actually performs at a similar level as some QD approaches (such as SS-LC).

Fig. 6.3 also highlights that while novelty search alone underperforms compared to surprise search, NSS and SS-NS, when combined into a QD algorithm the performance rankings are much different. NSS-LC and NS-LC perform much better compared to SS-LC (significantly for NSS-LC), although the fact that a version of NS-LC was used to find appropriate mazes that it can solve may have been a reason for the good performance of NS-LC. Moreover, NS-SS-LC does not perform better than NSS-LC and NS-LC in the same way as NS-SS does over NSS and NS. While the performance of NS-SS-LC is further discussed in the following main comparison of QD algorithms and in section 6.3, a likely reason is that the simultaneous optimization of three objectives makes the problem more difficult for a multi-objective approach to solve (Purshouse and Fleming (2007)).

Finally, it is obvious that the introduction of a novelty archive to surprise search leads to a much worse performance. The working hypothesis for this behavior is the divergence

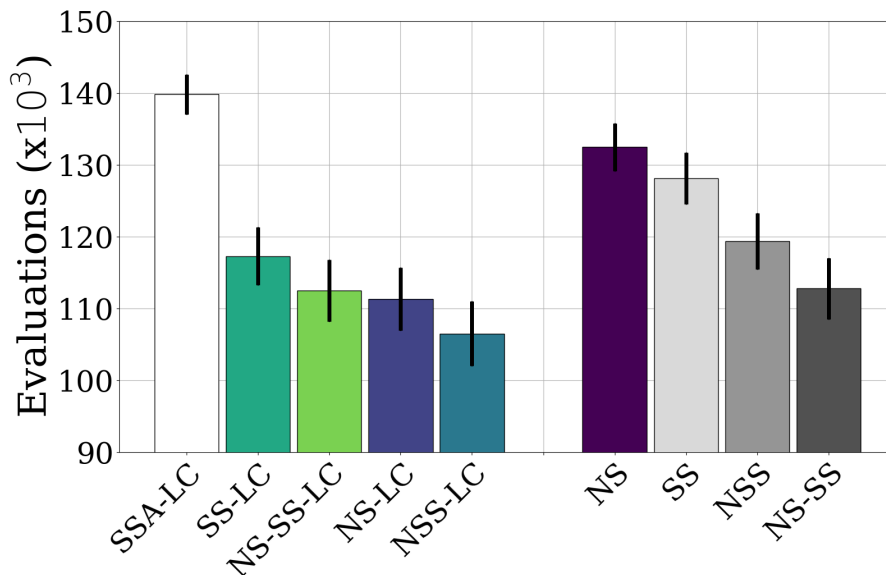


Figure 6.3: **Sensitivity to algorithmic components:** Average number of evaluations across the 10 easiest mazes for five QD approaches (NS-LC, NSS-LC, SS-LC, SSA-LC, NS-SS-LC) and four divergent algorithms (NS, SS, NSS, NS-SS).

dimension of this multi-objective approach. Specifically, “vanilla” surprise search attempts to deviate from predicted points in the search space, which can force it to back-track to previously seen areas of the search space. Introducing a novelty archive, where such previously visited areas of the search space are kept, actively deters back-tracking and impairs the surprise search component of SSA-LC.

Comparison of QD Algorithms

Having ensured an optimal set of parameters for all QD algorithms tested, each of the four quality diversity methods are applied to each of the 60 generated mazes. Comparisons among QD methods—and against novelty search as an indicative divergent search algorithm—aim to answer two core questions: whether an algorithm is more likely to find a solution in an evolutionary run than others (addressed in the successes analysis), and whether an algorithm finds such a solution in fewer evaluations (robustness analysis). While not a measure of performance, the evolved controllers are also compared in terms of the complexity of the ANN evolved (genomic complexity analysis), as an indication of genotypes favored by each approach.

Successes: Table 6.3 shows how each method compares in terms of number of successes in each of the 60 generated mazes. In this table, the number of mazes where one approach outperforms the other (in terms of successful runs out of 50) is shown on a per column and per row basis. As expected, novelty search as a pure divergent approach has the lowest rank compared to the other algorithms with an average of 2.1% instances outperforming the four QD algorithms and 78.7% instances being outperformed. On the other hand, NSS-LC outperforms all approaches more often (58.7%) than it is outperformed (20.4%). Notably, SS-LC does not perform equally well, as it is often outperformed by NS-LC (51.7%); however, NSS-LC outperforms NS-LC (48.3%) more often than it is outperformed by NS-LC

Table 6.3: **Algorithms tournament:** Percentage of 60 generated mazes for which the algorithm in a row has a strictly greater (≥ 1) number of successes compared to the algorithm in a column. Last row and last column are respectively the average of each column and the average of each row.

	NS	NS-LC	NSS-LC	SS-LC	NS-SS-LC	Average
NS	–	1.7%	0.0%	5.0%	1.7%	2.1%
NS-LC	83.3%	–	30.0%	51.7%	40.0%	51.2%
NSS-LC	81.7%	48.3%	–	60.0%	45.0%	58.7%
SS-LC	70.0%	28.3%	25.0%	–	30.0%	38.3%
NS-SS-LC	80.0%	41.7%	26.7%	46.7%	–	48.7%
Average	78.7%	30.0%	20.4%	40.8%	29.2%	–

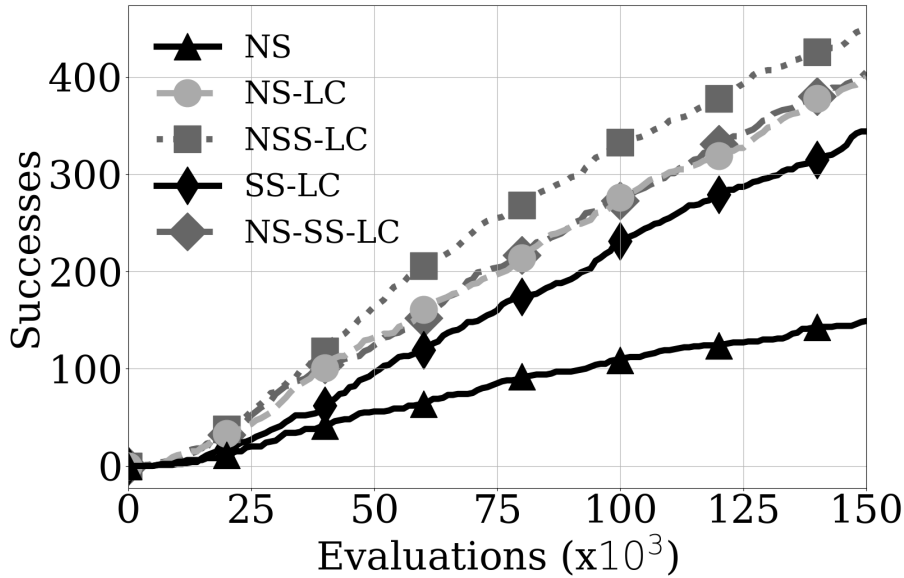


Figure 6.4: **Robustness:** number of successes over evaluations by aggregating all the runs of the 60 generated mazes for each approach.

(30%). It is interesting that NS-SS-LC outperforms NS-LC (41.7%) marginally more often than the opposite (40%), but when compared to all the approaches it has fewer instances of superior behavior (48.7%) than NS-LC.

It is important to note that NSS-LC is also superior to NS-LC in the 10 easiest mazes, where both approaches are more likely to have a successful run than in the hardest mazes of the test set; in harder mazes, a successful run is largely a matter of chance. In the 10 easiest mazes of Fig. 6.1, NSS-LC outperforms NS-LC in 6 of 10 mazes and is outperformed by NS-LC in 4 mazes. Note that results in Table 4.3 are from 50 runs independent of the runs performed for the sensitivity analysis; however, admittedly the parameters of both NS-LC and NSS-LC were optimized explicitly for these 10 mazes of Fig. 6.1.

Robustness: While the number of evolutionary runs resulting in a success is a good indication of algorithms’ performance, it is worthwhile to investigate how quickly those solutions are found. We use robustness as a secondary performance metric: robustness is

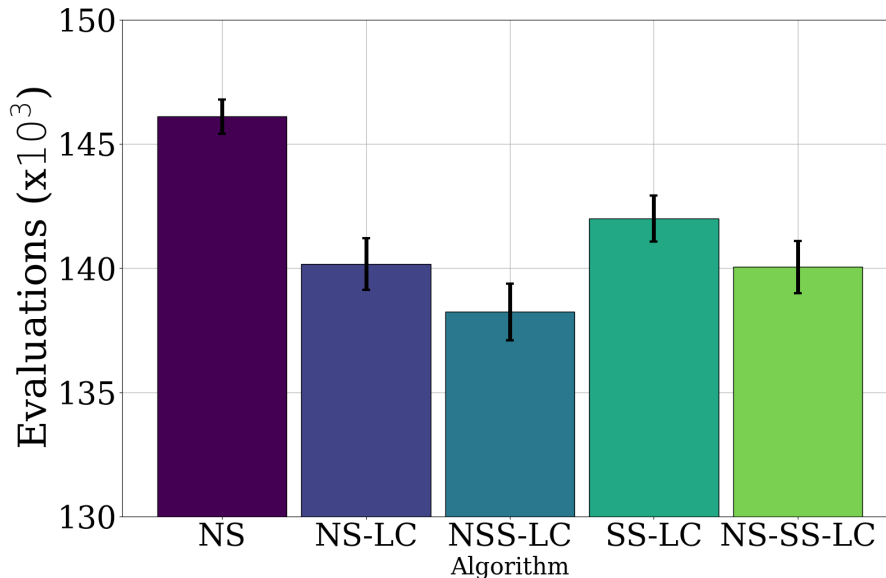


Figure 6.5: **Number of evaluations** on average by aggregating all the runs of the 60 generated mazes for each algorithm (i.e., 3000 runs per algorithm). Error bars denote 95% confidence intervals.

defined as the number of overall successes obtained by each algorithm after a number of evaluations. This indicates how consistent the results are, should evolution stop earlier than the current generation limit.

Fig. 6.4 and 6.5 report the robustness and the average number of evaluations respectively per method. Results are aggregated among all the runs of the 60 generated mazes per method, i.e., a total of 3000 runs. Aggregating successes across mazes allows for a more realistic view of algorithmic performance: averaging successes across the 60 mazes leads to very large deviations as some mazes are much harder than others (based on NS-LC performance described in subsection 6.1.2). Unsurprisingly, novelty search is outperformed by every other algorithm from $20 \cdot 10^3$ evaluations onwards, achieving a total of 149 successful runs in 60 mazes. NSS-LC outperforms NS-LC and NS-SS-LC after $20 \cdot 10^3$ evaluations and consistently reaches more successes than any other method from that point onward. When the maximum evaluations are reached, NSS-LC obtains 451 successes versus the 400 successes of NS-LC, 405 successes of NS-SS-LC and the 344 successes of SS-LC. The superiority of NSS-LC over NS-LC is also evident in Fig. 6.5, as on average NSS-LC requires significantly fewer evaluations than every other algorithm. This finding is not surprising per se, since NSS-LC finds more solutions while other approaches often spend their entire budget (evaluations) searching but not finding a solution. However, Fig. 6.5 establishes that this difference in successes results in a statistically significant acceleration of the algorithm. NS-SS-LC seems to perform very similarly to NS-LC, both in terms of number of successes over the progress of evolution, and in terms of average evaluations of Fig. 6.5. This observation is corroborated by tests in the sensitivity analysis. Finally, results of SS-LC are underwhelming also in terms of robustness, as it quickly falls behind both NSS-LC and NS-LC, and on average it needs significantly more evaluations than other QD alternatives.

Genomic Complexity: As an indication of the type of solutions favored by each evolutionary method, we compare the complexity of the evolved ANNs which were successful

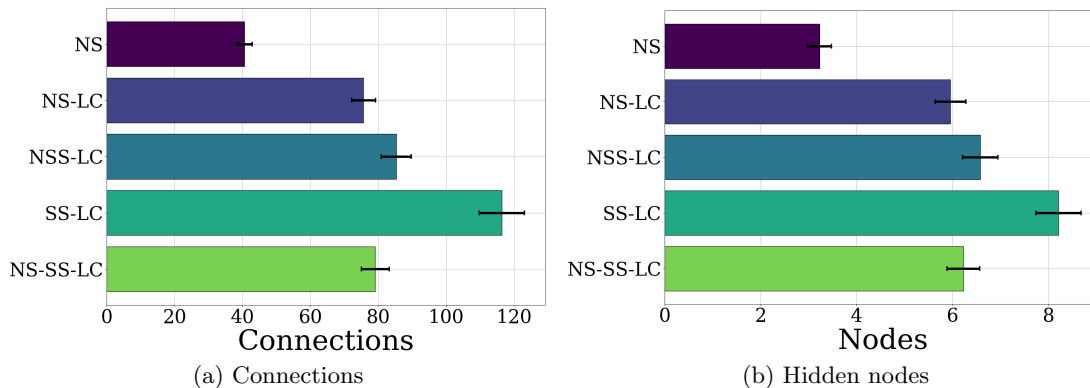


Figure 6.6: **Complexity**: number of connections and hidden nodes on average for evolved ANNs which solve the mazes per approach. Error bars denote 95% confidence intervals.

in finding the goal in each method (i.e., 400 ANNs for NS-LC, 149 ANNs for novelty search etc.). Genotypic complexity refers here to the average number of hidden nodes and connections of a successful ANN. In Chapter 4 and 5, the genotypic complexity across four divergent search methods (NS, SS, NSS and NS-SS) showed that each method favored different structures, with surprise search favoring far larger networks than novelty search. Figure 6.6 shows the average number of hidden nodes and connections per approach. As in previous findings (Chapter 4 and Chapter 5), novelty search favors very small networks (few connections, few hidden nodes), while NS-LC has denser and larger networks than novelty search but less than both NSS-LC and SS-LC (significantly so for SS-LC). Finally, it is interesting to note that NSS-LC has significantly smaller and sparser networks than SS-LC, striking a happy (i.e., most effective) medium between NS-LC and SS-LC. Structural metrics of ANNs evolved by NS-SS-LC have no significant differences with either NSS-LC or NS-LC, but seem to fall between the two; the similarity with networks of NS-LC is further evidence that NS-SS-LC performs a very similar search process as NS-LC, which explains their similar performance in Fig. 6.4.

6.2 Maze Navigation Testbed: Second Set

In the previous two chapters we tested the capabilities of surprise search (Chapter 4), novelty-surprise search and novelty search-surprise search (Chapter 5) in two maze navigation testbeds, one made of human-designed mazes and one made of procedurally generated mazes. It is interesting to conclude our analysis of the newly introduced QD algorithms (SS-LC, NSS-LC and NS-SS-LC) with the previously introduced mazes, in particular the four authored mazes (i.e., medium, hard, very hard and extremely hard mazes, see Fig. 4.1) and the set of procedurally generated mazes presented in Chapter 4. The old set of generated mazes will be called “second set” of generated mazes from now on, in order to avoid confusion with the first set of generated mazes introduced in the previous section.

This section starts with an analysis of the performance in terms of efficiency and robustness of NS-LC, SS-LC, NSS-LC and NS-SS-LC in the four authored mazes (Section 6.2.2). Then, in Section 6.2.3 we perform an analysis in terms of efficiency, robustness and successes with the second set of generated mazes. For all the experiments reported in this section, significance is at a 95% confidence. For multiple pairwise comparisons the Tukey’s

range test is used to establish significance.

6.2.1 Algorithms

The maze navigation testbed is the same used in the two previous chapters and the previous section. In order to assess the performance of the surprise-based QD algorithms, we follow the same methodology as in Chapter 4 and validate the three QD algorithms across the previously introduced set of mazes.

In particular, we test three surprise-based QD algorithms and two baselines:

- Novelty Search, as described in Chapter 2. The parameters of NS (n_{NS}) are the same as NS-LC in Table 6.2.
- Novelty Search with Local Competition. NS-LC uses a steady-state NSGA-II multi-objective algorithm (Li et al., 2017) to search for non-dominated solutions on the dimensions of novelty (Eq. 2.3) and local competition. All parameters are the same of Table 6.2.
- Surprise Search with Local Competition. SS-LC uses a steady-state NSGA-II multi-objective algorithm (Li et al., 2017) to search for non-dominated solutions on two objectives: surprise (Eq. 3.2) and local competition. All parameters are the same of Table 6.2.
- Novelty-Surprise Search with Local Competition. NSS-LC uses a steady-state NSGA-II multi-objective algorithm (Li et al., 2017) to search for non-dominated solutions on two objectives: a linear combination of novelty and surprise, and local competition. All parameters are the same of Table 6.2.
- Novelty Search-Surprise Search-Local Competition. NS-SS-LC uses a steady-state NSGA-II multi-objective algorithm (Li et al., 2017) to search for non-dominated solutions on three objectives: novelty, surprise and local competition. All parameters are the same of Table 6.2.

To perform a fair comparison with the results obtained in the previous section, we use the same parameter setup used for the QD algorithms in Section 6.1.1. Also for NS and NS-LC we use the same setup chosen previously, i.e., $n_{NS} = 15$ and $n_{LC} = 5$. A summary of the parameters can be found in Appendix A.

6.2.2 Experiments and Analysis: Authored Mazes

Firstly, we test the performance of surprise-based QD algorithms against the four authored mazes introduced in the Chapter 4 (see Fig. 4.1). For the medium (Fig. 4.1a) and hard maze (Fig. 4.1b) we employ the parameter setup used in Lehman and Stanley (2011a) and we count as successful a navigation policy that manages to reach the goal within a radius of five units at the end of an evaluation of 400 simulation steps. For the harder mazes, very hard (Fig. 4.1c) and extremely hard (Fig. 4.1d) we use the parameters used in Chapter 4, i.e., the number of simulation timesteps grows to 500 and 1000 for the very hard and extremely hard maze respectively. Following the same logic, the number of generations is fixed to 300 for the two easiest mazes (medium and hard maze) and 1000 for the hardest ones (very hard and extremely hard maze).

Performance

Firstly we investigate the performances of the QD algorithms in terms of *efficiency* and *robustness*. *Efficiency* is defined as the average number of evaluations taken by each approach to find the solution in the maze. If we look at the number of evaluations across the four authored mazes we can see that the performances of the four QD algorithms are comparable, and the differences are minimal, especially between NSS-LC and NS-LC. In the easiest maze, Fig. 6.7a we can see that all the QD approaches outperform novelty search. On average, NS-LC and NSS-LC perform better compared to other approaches, with a really small advantage for the latter ($9,8 \cdot 10^3 \pm 3,0 \cdot 10^3$ and $10,7 \cdot 10^3 \pm 2,3 \cdot 10^3$ evaluations for NS-LC and NSS-LC respectively). Similar evidence is shown in the hard maze, where Fig 6.7b shows that all the QD approaches outperform novelty search. As in the previous maze, NS-LC and NSS-LC take fewer evaluations on average to find successful robots, respectively $18,6 \cdot 10^3 \pm 4,3 \cdot 10^3$ and $18,9 \cdot 10^3 \pm 4,2 \cdot 10^3$ evaluations. In the more deceptive very hard maze (Fig. 6.7c) we can notice that all the tested approaches perform significantly better than novelty search ($p < 0.05$), but the differences between the four QD approaches are again minimal; on average NSS-LC is faster than the other algorithms with $52,0 \cdot 10^3 \pm 1,3 \cdot 10^3$ evaluations. Finally in the extremely hard maze, Fig. 6.7d, NS-LC and NSS-LC outperform the other two QD algorithms with respectively $104,5 \cdot 10^3 \pm 2,0 \cdot 10^3$ and $104,2 \cdot 10^3 \pm 2,0 \cdot 10^3$ evaluations on average. Furthermore, novelty search is outperformed significantly by any other approach ($p < 0.05$).

If we look at the *robustness* across the four mazes, similar conclusion can be drawn. In the medium maze, Fig. 6.8a shows that the four QD approaches perform similarly, and from 5,000 evaluations onwards they all start to be faster compared to NS. In the hard maze (Fig. 6.8b), we can notice that SS-LC struggles to perform as well as the other approaches, while NSS-LC, NS-LC and NS-SS-LC perform similarly across the entire evolutionary history. Fig. 6.8c shows the robustness comparison for the very hard maze; we can notice that in this maze NS-SS-LC is the slowest of the four QD approaches, while the behaviors of the other three approaches are quite similar. In the last maze (extremely hard maze, Fig. 6.8d), we can see that even the quality-diversity algorithms struggle in finding 50 out of 50 successes, but no clear differences are identified between the QD algorithms tested. In particular, NSS-LC and NS-LC find 46 successes, while NS-SS-LC and SS-LC obtain 44 and 39 successes respectively. SS-LC seems to be again the slowest of the QD approaches, while the other three algorithms have comparable performances.

Furthermore we can compare the performance of surprise-based QD with surprise search, NSS and NS-SS (see Chapter 4 and Chapter 5). In terms of efficiency, the introduced QD approaches perform similarly to NSS and NS-SS, while NS-LC, NSS-LC and NS-SS-LC outperform them only in the hardest maze, but no significance can be established. Unsurprisingly the surprise-based QD algorithms outperform surprise search in the two hardest mazes.

Overall we can see that, unlike the mazes tested in Section 6.1, these mazes are not deceptive enough to show clear differences in terms of performance. In terms of efficiency, on average NSS-LC outperform NS-LC in three out of four mazes, but the performance gap between the two approaches is not significant.

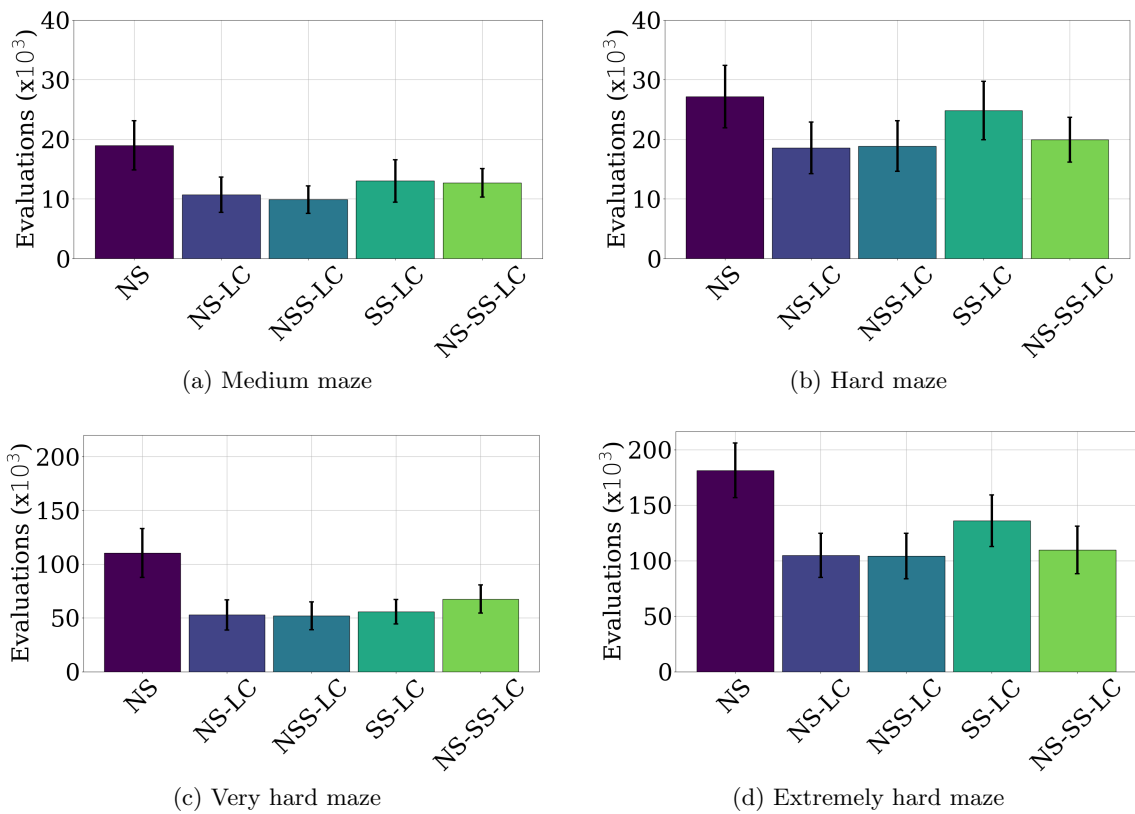


Figure 6.7: **Efficiency** comparison for the four mazes in Fig. 2.8. The graphs depict the evolution of algorithm successes in solving the maze problem over the number of evaluations. The maximum number of evaluations is $75 \cdot 10^3$ for the medium and hard maze, $250 \cdot 10^3$ for the very hard and extremely hard maze.

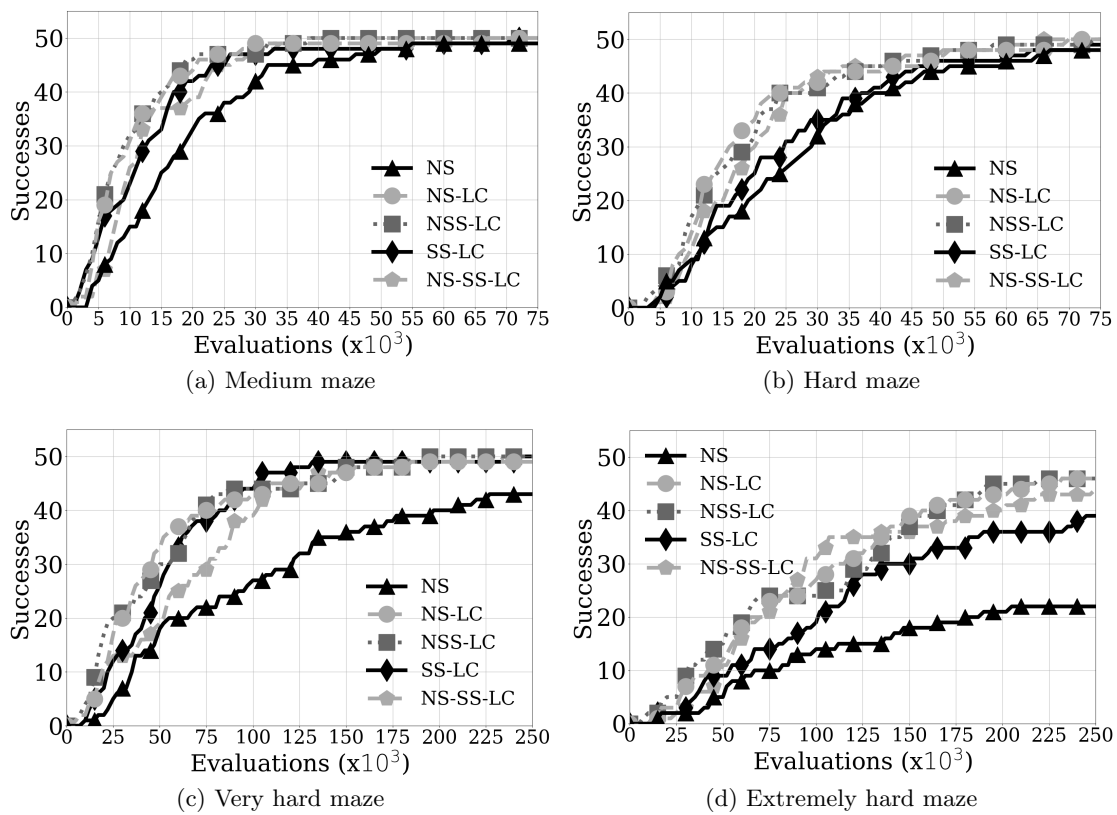


Figure 6.8: **Robustness** comparison for the four mazes in Fig. 2.8. The graphs depict the evolution of algorithm successes in solving the maze problem over the number of evaluations.

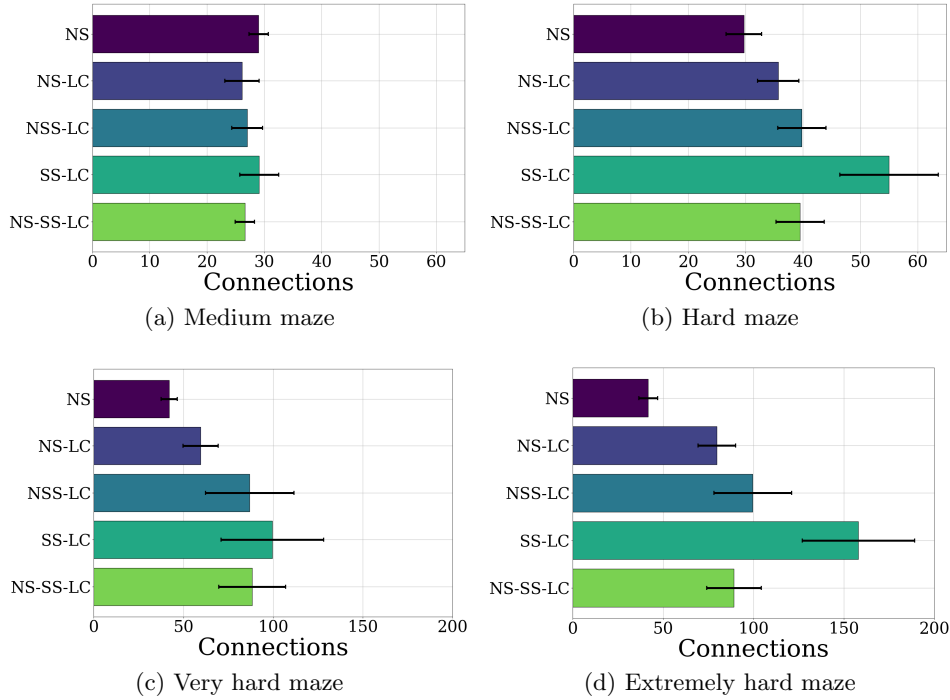


Figure 6.9: **Genotypic Space: Connections.** Metrics of genomic complexity of the final evolved ANNs, averaged from successful runs. Values in parentheses are 95% confidence intervals.

Genomic Complexity

As done in Chapter 4 and 5, we offer an overview of the metrics collected from the successful ANN evolved by each approach across the four authored mazes. In particular, Fig. 6.9 and Fig. 6.10 present the genomic complexity of the successful robots collected from the experiments run in the four mazes. We can observe that, as expected, SS-LC tends to create the bigger networks, both in terms of connections and number of hidden nodes, as can be noticed in the hard, very hard and extremely hard mazes. On the other hand, NS-LC tends to create the simplest networks compared to the other QD algorithms, and only NS creates smaller and less dense ANNs. Finally, NSS-LC and NS-SS-LC create “medium” size ANNs, i.e., the complexity of the ANNs evolved by these two approaches is between the two extremes represented by NS-LC and SS-LC.

6.2.3 Experiments and Analysis: Second Set of Generated Mazes

In the previous section, we tested the capabilities in search policies for the surprise-based QD algorithms across a number of generated mazes. However, these mazes were biased by the selection policy explained in Section 6.1.2. In Chapter 4 we introduced another set of generated mazes, used to test the capabilities of surprise search across several mazes. Therefore, in order to offer a thorough view of the performances of surprise-based QD algorithms, we test them against the same set of generated mazes. The parametrization setup is analogous to the one used in Chapter 4 and Chapter 5: we run each algorithm for 50 times per maze, the number of simulation steps is set to 300 and the number of

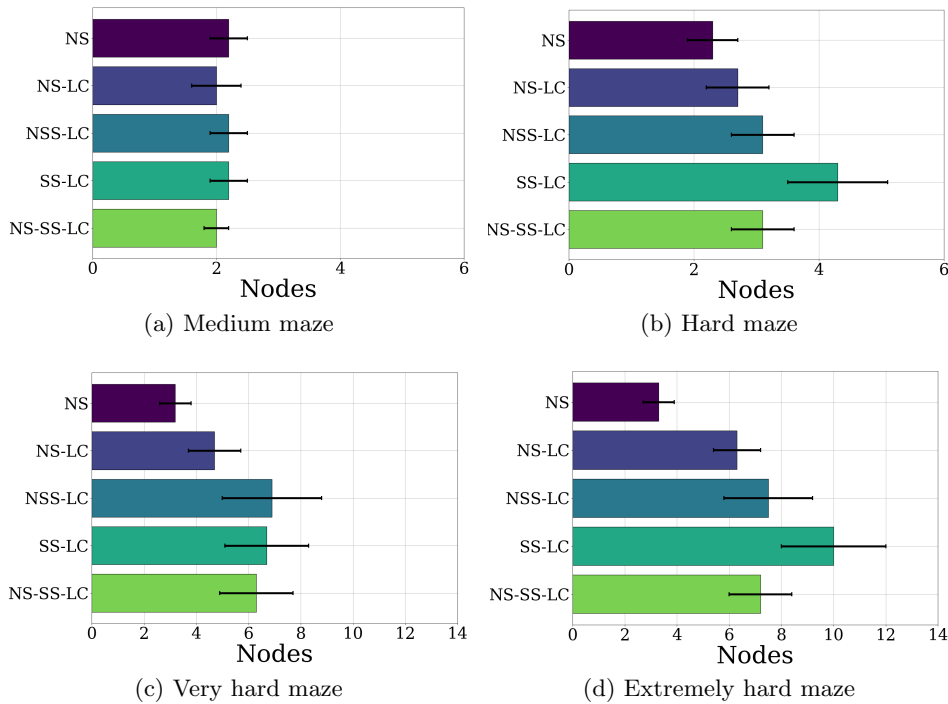


Figure 6.10: **Genotypic Space: Hidden Nodes.** Metrics of genomic complexity of the final evolved ANNs, averaged from successful runs. Values in parentheses are 95% confidence intervals.

generations is 600.

Performance

Inspired by the analysis done in the previous two chapters, we perform an analysis by aggregating all the runs of the 60 generated mazes, and we compare the efficiency and robustness for the algorithms at test. If we look at the *efficiency*, Fig. 6.11 shows that like with the four authored mazes, the performances of the four QD approaches are comparable. We can see that NSS-LC performs on average better than the other three QD algorithms, obtaining $62,0 \cdot 10^3 \pm 2,2 \cdot 10^3$ evaluations, while NS-LC, SS-LC and NS-SS-LC require respectively $64,0 \cdot 10^3 \pm 2,2 \cdot 10^3$, $64,9 \cdot 10^3 \pm 2,2 \cdot 10^3$ and $64,0 \cdot 10^3 \pm 2,2 \cdot 10^3$ evaluations on average. Unsurprisingly, they all outperform significantly novelty search ($p < 0.05$).

In terms of *robustness*, Fig. 6.12 reveals that NSS-LC begins to be faster from 40,000 evaluations onwards, and it constantly finds more successes from that point on. At the end of the 150,000 evaluations, NSS-LC reaches 2,195 successes, while NS-LC obtains 2,152 successes, NS-SS-LC 2,157 successes and SS-LC 2,146 successes. The novelty search algorithm finds fewer successes than any other approach considered, in particular from 15,000 evaluations onwards. An interesting insight arising from this analysis is that all the QD algorithms require half of the evaluations compared to the mazes in Section 4.1.4 and they obtain 5 times more successes. As noticed in Chapter 5, this highlights the relatively low difficulty of the majority of the generated mazes in this set, which evidently are not deceptive enough to show clear differences in terms of performance between the four QD algorithms. If we compare these results with the previous two chapters, we notice

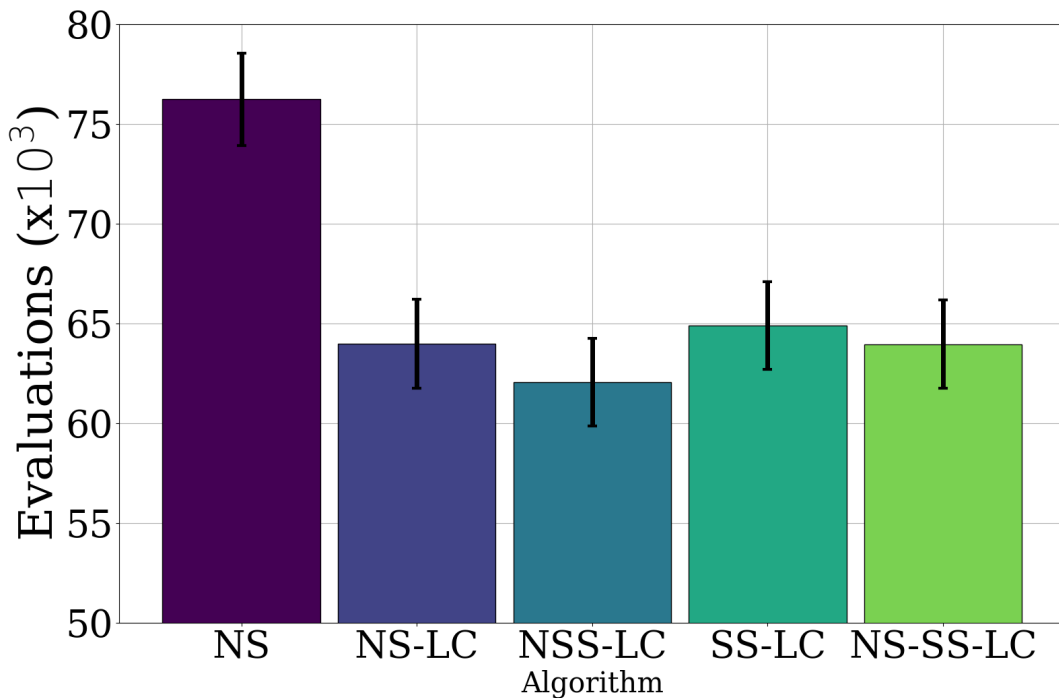


Figure 6.11: **Efficiency:** algorithm successes in solving all the generated mazes over the number of evaluations for each considered method.

that—in terms of efficiency—all the surprise-based QD approaches outperform the two novelty-surprise approaches, in particular, NSS-LC manage to outperform both significantly ($p < 0.05$). We can draw similar conclusions if we compare the results with surprise search: all the surprise-based QD algorithms significantly outperform SS and reach more successes overall.

Finally, we focus on the successes obtained by the QD algorithms in the 60 generated mazes. Table 6.4 shows the number of mazes where a selected method achieves at least one more success (out of 50) compared to the successes obtained by another approach. The reported results show that NSS-LC performs better compared to the other QD approaches, both in terms of number of comparisons won and the number of comparisons lost. Surprisingly, novelty search overall outperforms the other approaches on average more often compared to the other algorithms. As noted in Chapter 5, it seems that this set is relatively easy for divergent and quality diversity algorithms, and the majority of the mazes are equally solvable for all the approaches. Interestingly, the average number of mazes where NS is outperformed is mostly constant: it seems that for a selected number of mazes, quality diversity is a necessary condition to reach more successes more often.

Genomic Complexity

In order to investigate the networks evolved by each evolutionary approach, we compare the complexity of the successful ANNs across the 60 generated mazes for each algorithm testing. Genomic complexity is defined as the average number of hidden nodes and connections of the ANNs that were able to find the goal for the selected method. Fig. 6.13 shows the number of hidden nodes and the number of connections on average for each method. As expected, novelty search evolves the simplest networks overall, i.e., ANNs with the fewest number

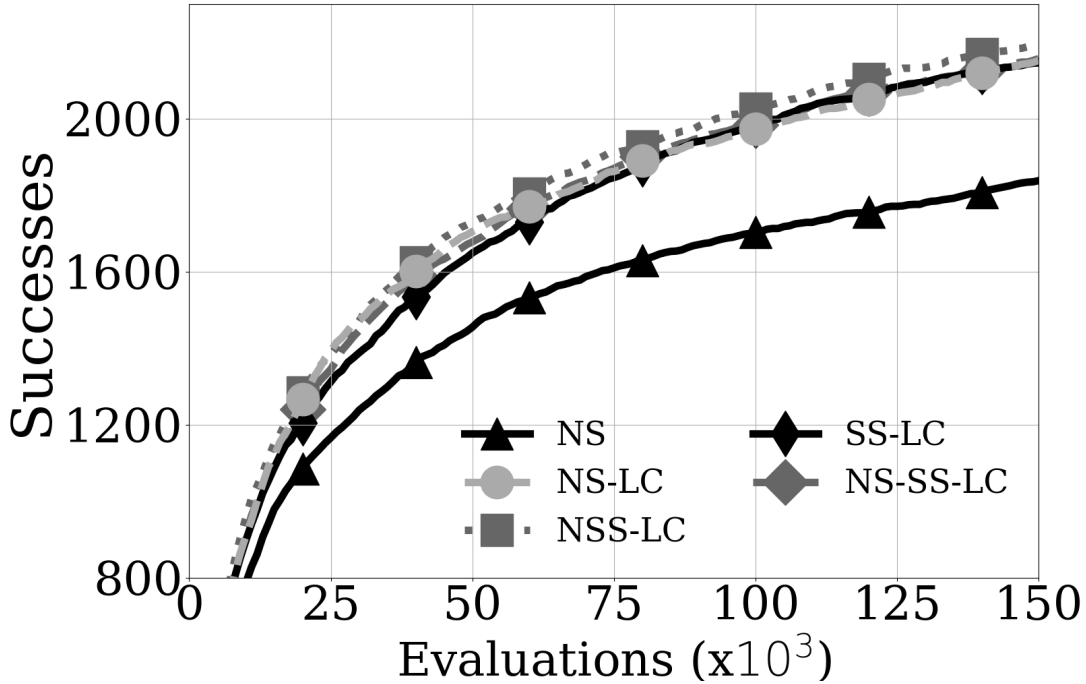


Figure 6.12: **Robustness:** algorithm successes in solving all the generated mazes over the number of evaluations for each considered method.

Table 6.4: **Algorithms tournament:** Percentage of 60 generated mazes for which the algorithm in a row has a strictly greater (≥ 1) number of successes compared to the algorithm in a column. Last row and last column are respectively the average of each column and the average of each row.

	NS	NS-LC	SS-LC	NSS-LC	NS-SS-LC	Average
NS	–	36.6%	36.6%	35.0%	36.6%	36.3%
NS-LC	43.3%	–	28.3%	16.6%	23.3%	27.9%
SS-LC	41.6%	21.6%	–	18.3%	20.0%	25.4%
NSS-LC	41.6%	33.3%	30.0%	–	28.3%	33.3%
NS-SS-LC	41.6%	23.3%	26.6%	16.6%	–	27.1%
Average	42.1%	30.4%	21.6%	21.3%	27.1%	–

of nodes and connections. This pull for the simplicity is probably what makes the ANNs smaller for NS-LC compared to the surprise-based QD approaches. On the other hand, SS-LC shows the higher complexity, consistently with the results obtained with the harder set of mazes evaluated in Section 6.1.2 and with the four authored mazes (Section 6.2.2). Finally, the two QD approaches that couple novelty and surprise with local competition, NSS-LC and NS-SS-LC, generate medium-complex ANNs (i.e., their metrics fall between the two extremes represented by NS-LC and SS-LC), with slightly more complexity for the former. If we compare the complexity of the five methods with the results obtained in the four authored mazes (6.2.2), we can notice that similar conclusions can be drawn. Interestingly, unlike the previous results in 6.1.2, NS-LC and NS-SS-LC show a significantly different number of connections in this set of easier mazes.

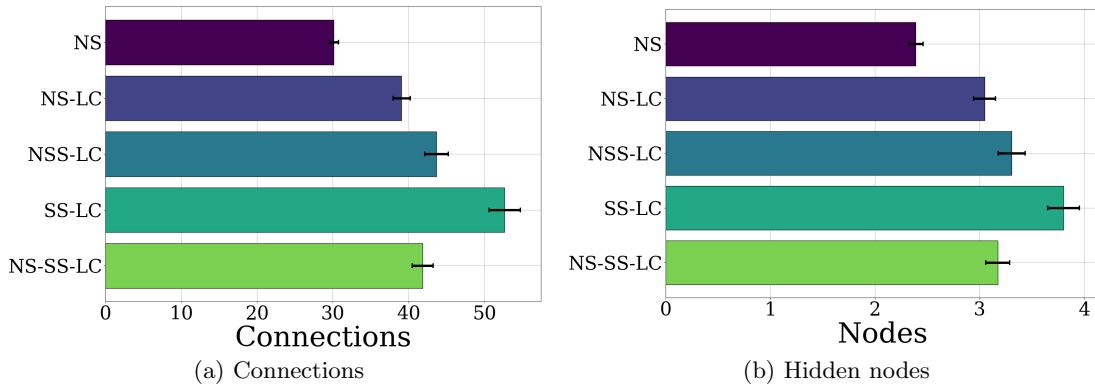


Figure 6.13: **Complexity:** number of connections and hidden nodes on average for evolved ANNs which solve the mazes per approach. Error bars denote 95% confidence intervals.

If we compare overall the complexity of the methods across the two set of mazes, we can notice that a higher number of hidden nodes and connections are generated for the harder (more deceptive) mazes. If we look at the number of connections, in the first set of generated mazes (Section 6.1.2), we can notice that if we average the number of connections across the five algorithms analyzed we obtain 79.4, while in the easier set of mazes this metric goes down to approximately 41.4 connections. We can notice a similar pattern for the hidden nodes. In particular, for the harder mazes, the number of nodes is doubled compared to the less deceptive mazes. This analysis confirms our hypothesis that the second set of generated mazes is relatively easy, as the required ANNs complexity to solve this set of mazes is comparable to the two easiest authored mazes (i.e., medium and hard maze), where the five approaches evolve on average ANNs of approximately 34 connections and 3 nodes.

6.3 Discussion

In the experiments of Section 6.1 and 6.2, we compared three new quality diversity algorithms with alternate dimensions of diversity combined with local competition in terms of robustness and efficiency in discovering solutions in diverse maze setups. The key findings are that NSS-LC outperforms the other QD approaches considered in this work, namely NS-LC, NS-SS-LC and SS-LC; NS-SS-LC has a very similar performance as NS-LC, while SS-LC underperforms by comparison. Furthermore, the experiments conducted with the second set of generated mazes and the authored mazes (Section 6.2) partially confirmed the results obtained with the first set of mazes. The relatively low difficulty of the mazes tested in the second section doesn't highlight the performance advantage of NSS-LC, which still outperform all the other approaches, but not significantly as with the first set of mazes.

The findings on NSS-LC validate our hypothesis that surprise can be a beneficial form of search for quality diversity. Coupled with our findings from sensitivity analysis in subsection 6.1.3 when components of the QD algorithms were added or removed and the results obtained with the second set of mazes in Section 6.2, we speculate and discuss on two likely reasons why adding surprise to novelty helps to achieve better overall performance. As discussed extensively in Chapter 5 and Chapter 4, a core hypothesis for the differences in algorithmic performance is that surprise search allows the algorithm to back-track, re-

visiting areas of the search space that have been explored in previous generations. Since surprise search operates on diverging from predictions, it may favor past behaviors if they are sufficiently different from predicted future trends. We have already provided such an example from nature in Chapter 1 where “terror birds” were surprising as they went against behavioral trends (birds were becoming smaller and better fliers) while not being particularly novel historically. By comparison, novelty search actively discourages re-visiting areas of the search space due to the novelty archive. This has been a hypothesis for the efficiency of surprise search compared to novelty search in purely deceptive tasks (Chapter 4), although admittedly it is difficult to prove quantitatively. However, an indication of the back-tracking nature of surprise is gleaned by the more complex networks evolved to solve the mazes in Fig. 6.6, Fig. 6.13, Fig. 6.9 and Fig. 6.10; the increased size compared to novelty search, which is consistent in other findings (as in Chapter 4, Chapter 5), is likely because same areas of the behavioral space are re-visited in later stages of evolution with larger networks. A clearer indication is that when surprise search with local competition also considers an archive of past novel individuals (SSA-LC), its performance drops significantly. We expect that the influence of an archive which actively deters back-tracking goes against the main drive of surprise search. It is evident that the back-tracking of surprise search does not seem to perform well in QD algorithms, as combining it with local competition seems to lead to local optima. This is perhaps why SS-LC underperforms in these tests, although it should be noted that it still scores more successes than novelty search alone in 70% of the first set of generated mazes (Table 6.3), in 41.6% of the second set of generated mazes (Table 6.4) and in 75% of the authored mazes. On the other hand, allowing novelty search to back-track through a surprise score component is likely why NSS-LC performs better. There is a fine balance between too much back-tracking (e.g., in SS-LC) and too little (e.g., in NS-LC), but the fact that NSS-LC performs better at higher λ values (e.g., $\lambda = 0.6$ or $\lambda = 0.7$ in Fig. 6.2) should be an indication that the novelty score primarily drives the search and back-tracking is favored in specific circumstances.

The second likely cause for a high-performing NSS-LC is based on recent research on novelty in highly successful patents, where the authors focused on the “sweet spot” of novelty (He and Luo, 2017). That study suggests that a sweet spot of novelty might exist on the border of what we can call a “conventional solution” and a completely novel solution. A similar hypothesis suggests that optimal solutions exist on the border between feasible and infeasible spaces in constrained optimization (Schoenauer and Michalewicz, 1996). In the maze navigation domain, for instance, behaviors that are “too novel” might dominate solutions that are less novel but more promising. Surprise may help to search in a more fine-grained fashion the space between a completely new solution and solutions already in the population or in the archive, in order to find the desired sweet spot of novelty. This is not only possible through back-tracking, but especially in NSS it is due to the balance of the two scores (novelty versus surprise). When the population in one generation becomes too novel compared to that of the previous generation, predictions will be even more distant points; in this case, the surprise score will have a larger impact as the distance from predicted points will be larger than that of past or current points. In general this balancing factor softens the greedy search for all-new solutions and, coupled with the secondary objective of local competition, results in a more efficient search process. The difference between NS-SS-LC and NSS-LC is perhaps the most surprising, as we as we would expect the decoupling of novelty and surprise scores to perform better compared than an aggregated approach. The very good performance of NS-SS as a two-objective divergent search algorithm (outperforming NSS, as reported in Chapter 5 and Section 6.1)

also indicated that a decoupling of the two measures of divergence would be beneficial and pointed to the orthogonal nature of surprise and novelty. However, as a QD algorithm the three-objective NS-SS-LC approach seemed to perform on par with NS-LC, to the degree of evolving similar sized networks.

If we look at the results of the previous two chapters, we notice that overall surprise-based QD algorithms perform better in the hardest of the four authored mazes and the second set of generated mazes. These results, together with the findings reported in Fig. 6.3, confirm that a quality-diversity approach helps the maze navigation algorithm to find faster the goal of the mazes, especially for the hardest mazes, and that combining novelty and surprise in a quality diversity formulation helps to improve even further its performance.

6.4 Summary

This chapter explored the impact of different diversity strategies on quality diversity evolutionary approaches and tested the hypothesis that surprise search may augment the exploration capacity of QD algorithms. Building on the concept of novelty search as one dimension coupled with local competition as a second dimension, alternatives to novelty search which used surprise or a combination of novelty and surprise were devised. These new QD algorithms were tested extensively on the maze navigation domain; we used a broad set of 60 mazes with varying degrees of deceptive fitness landscapes and each algorithm was evaluated across a total of 3000 evolutionary runs. Additionally, we tested the capabilities of these newly introduced algorithm against two supplementary set of mazes (four authored mazes and a second set of 60 generated mazes), to obtain a complete picture of the performances against the previously introduced maze navigation tasks. This makes the total number of runs performed for each approach equal to 6200 runs. Experiments concluded that an aggregated novelty-surprise search with local competition outperforms other QD algorithms, both in terms of number of runs which find a solution to the problem and the number of evaluations in which such solutions are found. The chapter finally ended with an in-depth discussion of the several experiments conducted and how the obtained results opened new directions in exploring different notions of divergence and quality diversity. The following chapter will offer a general overview of the results obtained in this dissertations and dissect the main limitations of the methodologies presented in this thesis.

Chapter 7

Discussion and Conclusions

This dissertation aims to investigate how the notion of surprise can be exploited for evolutionary search. In order to do so, we provided a general definition of the algorithm that follows the principles of searching for surprise and we applied the idea across two paradigms, divergent search and quality diversity. In particular, we addressed three research questions: *1) how the notion of surprise can be generally defined as a quantitative measure for computational search, 2) when and under which circumstances surprise search can be an effective optimization algorithm in deceptive domains, 3) how surprise can be interwoven with other divergent and quality diversity algorithms.*

In order to answer the first question, we formalized surprise for computational search and then, starting from this quantitative formulation, we defined an algorithm that uses surprise as a *proxy* measure to make the required steps towards the global optimum. Inspired by the divergent search paradigm, we introduced a new algorithm called surprise search. In Chapter 3 we described in detail each sub-components of the algorithm and a number of extensions, which involve both novelty search and quality diversity approaches. In Chapter 4 we addressed the second question, by validating the proposed surprise search in two independent domains. The experiments suggest that surprise search yields either better (in maze navigation) or comparable (in soft robot evolution) efficiency compared to novelty search and it significantly outperforms objective search. Given the promising results obtained with surprise search alone, Chapter 5 and Chapter 6 tackled the last remaining research question, by investigating how we can couple surprise with other evolutionary approaches. In particular, Chapter 5 tested the idea of coupling novelty and surprise for the purposes of divergent search. The results in the maze navigation task and soft robot evolution show that coupling novelty and surprise (i.e., NSS, NS-SS) can be advantageous in terms of efficiency and robustness compared to their base components. In Chapter 6, finally, we investigated how surprise search impacts the performance of QD algorithms. We devised a number of surprise-based QD approaches and we tested them thoroughly against 120 generated mazes, with varying degree of difficulty. Surprise has shown to be an effective reward for quality diversity and to improve the performances of state-of-the-art approaches in a significative way.

In conclusion, we argue that this thesis has investigated extensively to what degree surprise can be employed and used in the context of computational search. The obtained results show that (a) surprise search has clear advantages over other forms of evolutionary divergent search, and outperforms traditional fitness search in deceptive problems, (b) surprise shows greater exploration capacity compared to other divergent search approaches,

(c) surprise efficaciously complements other forms of evolutionary search, i.e., novelty and local competition.

However, during this exploration, we found a number of limitations in the subcomponents of the introduced algorithms. Furthermore, although a great effort has been made to cover the two chosen domains thoroughly, there is an upper bound to the generality of the reported results. In Section 7.1 we describe these limitations in detail, both regarding the algorithms and the experiments reported in this dissertation. Finally, we broaden the scope of this work in Section 7.2: building on the existing work, we describe how the introduced surprise-based family of algorithms can be transferred to other domains.

7.1 Limitations

In this dissertation, a great effort has been made to cover most of the possible drawbacks, but, unsurprisingly, some limitations remain to be addressed. This section offers a detailed description of each limitation found and tries to address the main drawbacks with proposals for future work.

7.1.1 Limitations of Surprise Search

The experiments described in Chapter 4 have shown that surprise search is an efficient and robust approach for solving deceptive problems. However, we took several design decisions in terms of subcomponents employed, representation used and parameter settings, which affect the generality of the obtained results.

Model of Expected Behaviour

As described in Chapter 3, 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 that information is used to make a prediction*. In this thesis, the prediction of behaviour is based on the simplest form of 1-step predictions via linear regression. This simple predictive model shows the capacity of surprise-based algorithms given their performance advantages over state-of-the-art algorithms, but more work on exploring how much time and how the information is used to make the prediction is required. Further, the prediction space is missing a proper in-depth analysis, because admittedly it is difficult to understand how the predictions are affecting the performance of the algorithm and the exploration of the search space. In Chapter 4, it is argued that deviating from the predicted behaviour may create a *backtracking* behaviour that can result in revisiting areas of the search space and this may help surprise search to explore more broadly. The behavioural space analysis in the maze navigation testbed (Section 4.1.3) partially assessed the explorations capabilities of surprise search. However we lack empirical evidence on this intuition and a deeper investigation of the backtracking dynamic will be considered for future work. Another possible issue is related to the computation of prediction in the general case, especially in more complex tasks. While a simple linear regression model seems to be sufficient for delivering state of the art results in both simple and complex behavioral spaces (Chapter 4) other surrogate models (Jin, 2005; Jin et al., 2018) could be investigated for more complex tasks and domains (Gaier et al., 2018). A first possible step towards this investigation could be to use different prediction models (m in Eq. 3.1) for the computation of surprise. For instance, we can imagine learning in an unsupervised manner a neural

network model of the environment, e.g., a *world model* (Ha and Schmidhuber, 2018), and use the learned model to predict non-linear behaviors of heterogeneous tasks.

Moreover, 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. For instance, in the maze navigation domain, it is possible to compute a 3-dimensional line over the three dimensions considered (two-dimensional Euclidean space and time) 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, an artificial neural network model or a *kriging* model (Jin et al., 2018). Investigating how the correctness of the prediction model affects the performance of surprise search is another possible direction of future work. For instance, we can consider using multiple prediction models at the same time (e.g., linear and non-linear models) and compute the surprise score as a weighted sum of the distances calculated from the expected behaviors. Based on the correctness of the predictions in the past generations, we can then change the weights proportionally to the accuracy of the different prediction models.

Another possible direction for future investigation is coupling surprise search with surrogate models. Indeed, one of the most important drawbacks in EC is the computation time needed for the evaluation of the generated solutions. A typical run might require from thousands to million evaluations of the individuals before finding an acceptable solution to the problem. When the computational cost is prohibitive, we can approximate the fitness landscape with a surrogate model, in order to accelerate the evaluation phase at the cost of less precision (Jin, 2011). Several approaches have been proposed, as in Karavolos et al. (2017) where a machine learned model is used to predict the balance of a game level in a first-person shooter. Surrogate models might be particularly effective for the soft robot evolution domain, as the VoxCad simulations are extremely heavy computationally (Cheney et al., 2013). Using a surrogate model may make the discovery of efficient robots more difficult given the deceptiveness of this domain, but it would certainly boost the number of evaluations and experiments which could be performed. Therefore, a trade-off between efficiency and computational cost should be explored.

Prediction Locality of Surprise Search

The algorithm presented in this dissertation 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_{SS} which is set by the algorithm designer, as noted in Chapter 3. Prediction locality can be derived from the behavioural space but also from the genotypic space. Experiments conducted in Chapter 4 (see Fig. 4.4) show that surprise is slightly more sensitive to the change of hyperparameters compared to other kind of search (i.e. for $k_{SS} < 100$ in the maze navigation domain). However, the performed experiments show that the performances stabilize as long as k_{SS} is sufficiently high for the problem at hand. Nonetheless, future investigations should test more in-depth the effect of locality for surprise search across different domains and investigate the sensitivity of k_{SS} extensively.

An important limitation is related with the clustering algorithm used for the prediction locality, k -means. Surprise search, in the form presented in this thesis, requires some form of behavioural clustering. While k -means was investigated in the reported experiments for its simplicity and popularity, it should be noted that for high values of k_{SS} the k -means

algorithm might end up not assigning any data point to a particular cluster; the chance of this happening increases with k_{SS} and the sparseness of data (in particular in datasets containing outliers) (Hartigan and Wong, 1979). Further, the seeding initialization procedure we follow for k -means 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 during the assignment step of k -means, as a data point 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 data point 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_{SS} . Comparative studies between approaches are needed, including different ways of dealing with (or taking advantage of) empty clusters.

As an alternative to k -means, we may consider using a different clustering algorithm, such as agglomerative clustering or DBSCAN (Rokach and Maimon, 2005). Another possible solution is to use a supervised learning algorithm to learn how to compute the locality of the solutions. This model can be implemented by collecting data from human users, and then compute the prediction locality through the trained model, for instance by means of *preference learning* (Yannakakis et al., 2018).

Behaviour Characterization and Deviation

Experiments in this thesis focus on diversifying the behaviour of the evolved ANNs (in the maze navigation testbed) and soft robots’ trails (Chapter 4); to do this, we use several behaviour characterizations (for the objective, for the distance, and for prediction in surprise search) which have featured extensively in previous work (Mouret and Doncieux, 2012; Cheney et al., 2013). However, the results might be influenced by the chosen behaviour characterization. Other behaviour characterizations have been proposed (Methenitis et al., 2015) and could be potentially used to measure distance, e.g., $d_n(i, j)$ in Eq. 2.3 and $d_s(i, j)$ in Eq. 3.2. The current algorithm allows for any degree and type of deviation from the expected behaviour. Inspired by novelty search, this thesis 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 Grace et al. (2015).

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 7.1 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 aggregate the final positions of all robots ($k_{SS} = 1$), 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.

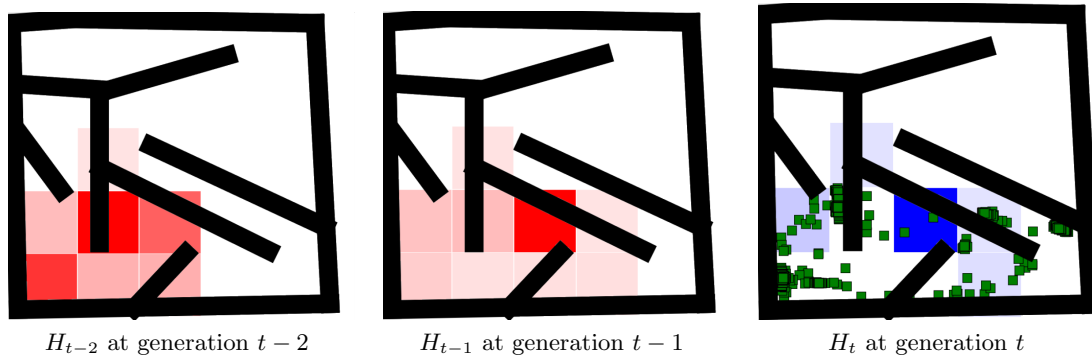


Figure 7.1: **Deviation in maze navigation:** Surprise search using heatmaps, at generation t . The first two heatmaps are computed in the last two generations by using the final robot positions, H_{t-2} and H_{t-1} . Using linear interpolation, the difference $H_{t-1} - H_{t-2}$ is computed and applied 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 .

7.1.2 Limitations of Novelty-Surprise Search

Extensive work has been done to test the hypothetical orthogonality of novelty and surprise (Chapter 5). The evidence confirmed our hypothesis, as we showed that the two approaches are more effective when combined compared to their base components, i.e., novelty search and surprise search. However, based on their algorithmic properties, using novelty and surprise search implies that the two dimensions change throughout the evolutionary process. Therefore, the analysis of the dynamics of the two divergent measures is hard, as both surprise and novelty are dynamic and their values depend on the state of the population. Indeed, novelty search depends on the state of the population and the novelty archive (Lehman and Stanley, 2011a,b), while surprise search depends on the historical trends and predictions which are recomputed in every generation (Gravina et al., 2017c, 2016b). Nonetheless, more theoretical and experimental work is required to understand the dynamics behind the beneficial synergy of the two divergent processes.

Towards that end, different combinations of the two rewards can be probed and tested. For example, Gomes et al. (2017) propose novelty-driven cooperative coevolutionary algorithms, where novelty helps the coevolutionary process to not converge towards a stable (and local optimal) solution. Extending on this idea, we can imagine that coupling novelty and surprise in a cooperative coevolution setting may help the coevolutionary algorithm to perform even better and, at the same time, may help to understand the underlying dynamics when these two divergent techniques are run together. Another possible implementation can couple novelty and surprise in a probabilistic fashion. Correll and Heer (2017) propose to use Bayesian Surprise (Itti and Baldi, 2006) to highlight unexpected patterns in data visualization. Also the notion of novelty can be formulated in a probabilistic manner, as described in (Li and Príncipe, 2018). Therefore, it is possible to test these two alternative formulations of novelty and surprise and compare them with the algorithms proposed in this dissertation. We can envision that a user-based model of novelty and surprise can be also realised. Inspired by interactive novelty search (Woolley and Stanley, 2014), we can imagine to compute surprise and novelty rewards based on the user preference and apply

them in an interactive evolution scenario (Liapis, 2014).

7.1.3 Limitations of Surprise for Quality Diversity

Surprise search has shown to be an advantageous quality diversity approach when coupled with novelty and local competition, as shown in Chapter 6. However, we found two main limitations during the analysis of the results. First, the quality diversity variants of surprise search have been tested only in one domain (i.e., maze navigation), which limits the generalization of the results obtained. The second domain, soft robot evolution, has not been tested due to the high computational time required to run the experiments (Cheney et al., 2013). The VoxCad simulations are indeed particularly expensive, especially as the resolution increases: indicatively, for resolutions of $10 \times 10 \times 10$, one evolutionary run takes 88.2 CPU hours. Second, we only tested one variant of the surprise-based quality diversity algorithms. Chapter 4 and Chapter 5 test two variants of SS, NSS and NS-SS (steady-state and generational), while all algorithms employed in Chapter 6 use a steady-state implementation of NEAT, as in (Lehman and Stanley, 2011a). In particular, the multi-objective algorithms (NS-LC, SS-LC, NSS-LC, NS-SS-LC) use a steady-state implementation of NSGA-II (Li et al., 2017). There are three main reasons behind this choice: first, for fair comparisons with the baselines used in this dissertation (NS, SS, NSS) which use a steady state implementation for the maze navigation testbed (Lehman and Stanley, 2011a; Gravina et al., 2016b, 2017a); second, due to evidence in (Lehman et al., 2013) that the generational counterpart of novelty search does not perform equally well in a maze navigation scenario, likely due to a “less informative gradient” for novelty search given by a generational reproduction mechanism (Lehman et al., 2013); third, due to arguments in recent studies (Durillo et al., 2009; Nebro and Durillo, 2009; Buzdalov and Parfenov, 2015) that a steady-state multi-objective implementation can be beneficial in terms of convergence and diversity for particular problems. Based on the aforementioned studies we assume that a steady state implementation is more suitable for the maze navigation testbed. Nevertheless, future work will test the degree to which this assumption holds by comparing current findings against a generational implementation of the proposed QD algorithms.

Another interesting insight obtained from the results in Section 6 has revealed that the three-dimensional solution NS-SS-LC does not perform as expected. We can assume that the simultaneous search for three dimensions (compared to all other QD approaches which search along two dimensions) was the primary cause of its subdued performance. As the dimensions increase, so does the number of non-dominated solutions (Purshouse and Fleming, 2007), which makes the search process slower. A further complication (as noted in the previous subsection) is that all three of the dimensions have dynamic, fluctuating scores: novelty search is sensitive to both the population and the novelty archive and the same individual may receive a different novelty score from one generation to the next. The same applies for surprise score, which depends on the ever-changing recent trends and predictions which are recalculated in every generation, and the local competition which again depends on other individuals in the population and the novelty archive. How multi-objective algorithms handle such dynamic fitness dimensions has not been sufficiently examined, and is an interesting direction of inquiry. Due to the high performance of NS-SS, however, we expect that an algorithm that can in principle handle more objectives more efficiently will lead to improved performance for NS-SS-LC. A possible way to address this problem could be implementing and testing different multi-objective algorithms for the surprise-based QD algorithms. We can envision that a reference point based algorithm, such as NSGA-III (Deb

and Jain, 2014), might mitigate some of the problems encountered and will be considered for future work. Further, the results reported in Chapter 6 shows that surprise with local competition underperforms compared to the other QD approaches. This might be due to the local competition reward formulation. Local competition is closely related to the inner workings of the novelty search algorithm, as it computes the localized reward based on the concept of the behavioural neighbourhood, while surprise search uses the prediction space. Therefore, we can imagine that a different model of local competition, working on both dimensions (the behavioural space used by novelty and the prediction space used by surprise), may be beneficial and improve even further the capabilities of the QD approaches proposed.

Finally, as noted in Chapter 6, we focused on enhancing one quality diversity approach with surprise: a multi-objective blend of a divergence score and a measure of local superiority. Other approaches for quality diversity such as MAP-Elites (Mouret and Clune, 2015; Cully et al., 2015) could also be considered, either as another QD algorithm used as a benchmark for NSS-LC, or to introduce surprise as a way of searching within the space of MAP-Elites. MAP-Elites is a QD algorithm where a descriptor space (a space defined by the designer of the algorithm) is discretized and stored as a grid. Given this N -dimensional space, MAP-Elites searches for highest performing solutions for each bin. The algorithm starts by generating a random set of solutions; these solutions are then evaluated and their associated descriptors recorded. Every generated solution is then potentially saved in their corresponding bin, computed based on the recorded descriptor. If the selected bin is empty, the solution is stored in the grid, otherwise the performances of the stored solution and of the new solution are compared, and only the best one is kept. After this initialization procedure, the stored solutions are uniformly selected to generate a new individual, by means of crossover or/and mutation. The new solution is then evaluated and then potentially stored in the corresponding bin if its performance is better compared to the solutions stored, following the same procedure as in the initialization. These three phases—selection, mutation, and evaluation—are repeated a number of times until the grid reaches the desired coverage or a predefined number of evaluations is reached.

As noticed the selection phase of the MAP-Elites algorithm chooses randomly the parents from the solutions stored in the grid. In order to aid the exploration of underrepresented solutions, we can imagine selecting the individuals stored in the grid based on their unexpectedness. Surprise could be integrated into MAP-Elites, for instance, by storing two different maps: one used as in the standard algorithm and the other one used for the surprise computation, with deviation from predictions used as a selection process for the algorithm (see Fig. 7.2). In (Gravina et al., 2019c) we perform a first step towards this direction in the soft robot domain, where surprise search is used to diversify the selection process of MAP-Elites.

7.1.4 Limitations of the Domains

The two domains used throughout in this dissertation present some limitations as well. In the maze navigation domain, a shared characteristic across the various mazes is the static environment used in the simulations and the use of only one goal per maze. A possible extension is to add more than one goal in the maze navigation problem, as previously tested in (Pugh et al., 2016). Another interesting direction is including dynamic obstacles in the maze: every n timesteps, we may add an obstacle in the maze, making the problem dynamic and more difficult (see Fig. 7.3). We can implement *dynamic maze navigation* in

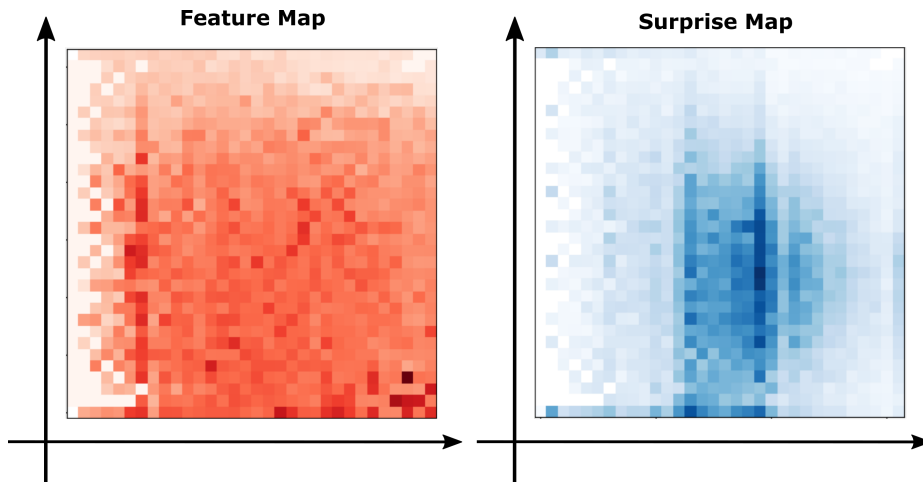


Figure 7.2: **A mockup of Surprise-based MAP-Elites:** the idea is to use two different descriptor spaces: the first one is computed as in Mouret and Clune (2015) while the second space acts as a probabilistic distribution model of surprise (*Surprise Map*) that can be used to evaluate the unexpectedness of the solutions in a probabilistic fashion.

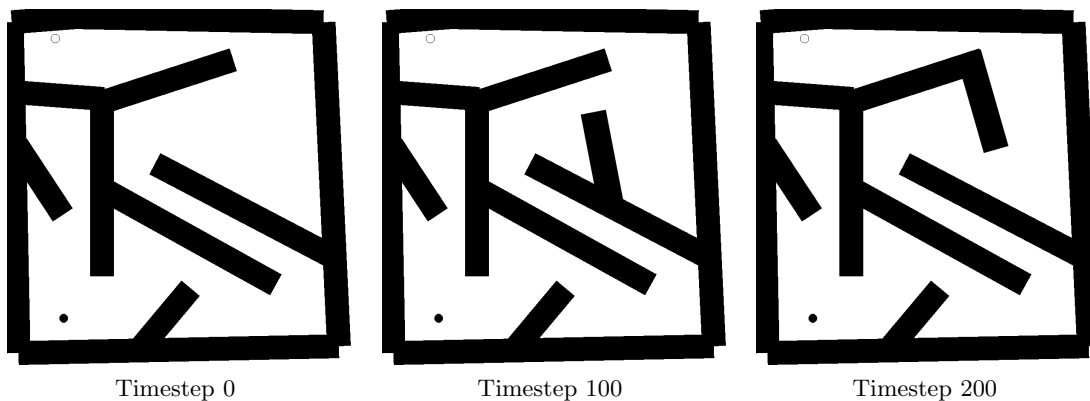


Figure 7.3: **Dynamic maze navigation:** a dynamic maze navigation problem, where the obstacles between the starting point and the goal are placed during the simulation with the wheeled robot, e.g., in this example at the timestep 100 and timestep 200.

two ways. The first way is to make the changes in the maze deterministically, i.e., all the times the wheeled robot traverses the mazes, the modifications of the maze will appear at the same place and at the same timestep. The second way is to place the obstacles in the maze randomly, making the problem also noisy. Another limitation is related to the soft robot domain. It is worth mentioning that analysis of the structural properties performed in Chapter 4 and 5 has focused on the rate of non-active materials and filled voxels as in (Mouret and Clune, 2015), but other structural properties could be relevant, such as the rate of active materials or more sophisticated distance measures. Furthermore, we can imagine changing the behaviour characterization employed. Instead of using the behavioural trails of the evolved robots, surprise-based algorithms may use their structural features directly, similarly to (Mouret and Clune, 2015).

A further limitation of the results reported in this work is related to the use of a specific

neuroevolution algorithm, NEAT (Stanley and Miikkulainen, 2002), for both domains. This poses the question whether the performances of the tested algorithms might depend on the features of neuroevolution. To address this limitation, it would be interesting to test the capabilities of surprise-based algorithms with different representations and testbeds. The proposed algorithms should be tested against a broader set of state of the art evolutionary algorithms, such as CMA-ES (Hansen, 2006) or multimodal optimization algorithms (Preuss, 2015).

To a degree, the experiment with the generated mazes (Chapter 4, Chapter 5 and Chapter 6) and the two independent implementations for each approach tests how generalizable the proposed algorithms are. Since results from that experiment indicated that surprise-based algorithms scale better to more deceptive problems, we need to further test the potential of the algorithm through more deceptive and complex environments. For instance, the capacity of these algorithms needs to be tested in other domains such as gameplaying or procedural content generation (Yannakakis and Togelius, 2018). Several works have studied the potential use of divergent search and quality diversity in computational creativity domains (Liapis et al., 2015; Liapis, 2016; Khalifa et al., 2018). This limitation has been partially addressed in (Gravina et al., 2016a), which will be described in more detail in the following section. However, a broader study on the performances of the surprise-based algorithms in non-NEAT domains is an important and necessary step for future work.

7.2 Extensibility: Beyond Mazes and Soft Robots

This thesis has put a great deal of effort in exploring the performance of the proposed algorithms across different setups and domains. However more directions wait to be explored, both concerning potential enhancement of the algorithms (e.g., prediction model or deviation model) and domains to which the algorithmic contributions of this work could be applied. This section offers an overview of the main directions that could be taken in the future.

7.2.1 Surprise Search for Procedural Content Generation

As other divergent search approaches (e.g., novelty search) have been successfully used for computational creativity and open-ended search, we can foresee that surprise could be an interesting addition in the search for unconventional artefacts, especially in the context of content generation and computational art. Surprise-based search is particularly well-suited, for example, in procedural content generation (PCG). Procedural content generation has been used since the 1980s in the game industry to quickly and computationally efficiently create elaborate structures such as the dungeons of *Rogue* (Toy and Wichman 1980) or the universe of *ELITE* (Acornsoft 1984). Commercially, procedural content generation is used primarily for two reasons: (a) to cut down on development effort and time, and (b) to create unexpected, unique experiences every time the game is played, thus increasing its lifetime and replayability value. Due to the former, small teams of game developers have been able to (procedurally) create grandiose gameworlds such as those in *Minecraft* (Mojang 2011) and *No Man's Sky* (Hello Games 2016). Due to the latter, games such as *Civilization V* (Firaxis 2010) have been immensely successful in retaining a userbase engaged despite the lack of e.g., an overarching campaign. In literature, different branches have been recognized for PCG: constructive, generate-and-test, search-based and Procedural Content Generation via Machine Learning (PCGML) (Togelius et al., 2011; Summerville et al., 2018)

Historically, constructive algorithms were the first algorithms capable of generating content for games automatically. Constructive algorithms are ad-hoc approaches that generate content without any quality control over the generated content. These algorithms feature two key advantages: they are fast and highly controllable. Given their advantageous characteristics, they are ubiquitous in commercially acclaimed games, such as in the action-based game *Bloodborne* (FromSoftware, 2015). Compared to the historical use of procedural content generation in games, academic interest in PCG from the perspective of artificial intelligence is relatively recent. Nonetheless, a number of constructive algorithms have been proposed and successfully tested in different genre such as puzzle games (Smith et al., 2013), platformer levels (Smith et al., 2011), mazes, dungeons (Horswill and Foged, 2012) and strategy game maps (Smith and Mateas, 2011).

Academic interest in PCG has often used *search-based* processes (Togelius et al., 2011) such as evolutionary computation to create game content which optimizes one or more game qualities deemed relevant by the designers. PCG research focuses on expanding the generative algorithms, going beyond *constructive* approaches (Togelius et al., 2011) While the majority of PCG research focuses on creating one final artifact which exemplifies the desired properties of its type, there are several attempts at creating a diverse set of content using, e.g., multi-objective optimization (Togelius et al., 2013), multi-modal optimization (Preuss et al., 2014) and novelty search (Liapis et al., 2015). PCG research has used many different sets of algorithms, often revolving around evolutionary computation and constraint satisfaction, among others. Broadly, evolutionary computation under the umbrella term *search-based PCG* (Togelius et al., 2011) evolves a large population of artifacts towards a certain objective, usually pertaining to in-game quality. Constraint satisfaction, instead, uses a carefully selected set of constraints to ensure that all of the generated content is playable (Smith et al., 2013).

More recent approaches involve the use of machine learning: PCGML is defined as the “generation of game content using machine learning models trained on existing content” (Summerville et al., 2018). The main difference from the approaches mentioned above is that the generation happens directly from the machine learning model, i.e., from the latent space learned from the data used to train the model. While PCGML can be used as in the search-based approaches, it complements the constructive and search-based PCG with specific capabilities more suited for machine learning, such as repair, critique and content analysis of the generated content. Given a sufficiently large database of representative content, PCGML can recognize issues in the generated content and suggest a possible way to fix it, e.g., an unbalanced map for a first-person shooter (Karavolos et al., 2017). Not only functional hints but also high-level suggestions can be generated: PCGML might act as a critique of the content and suggest possible improvements based on the knowledge acquired through the data used to train the model. For instance, Karavolos et al. (2018b) use deep learning to automate the design of adversarial shooter game and Karavolos et al. (2018a, 2019) couples SB-PCG and PCGML to automatically train a game designer and evolve a game creator for the FPS genre.

Procedural Content Generation through Quality Diversity

Being a recent trend in evolutionary computation, quality diversity algorithms are underexplored as procedural content generators, despite that diversification and quality are highly valued qualities from a user’s perspective (Gravina et al., 2019a). The game industry tends to focus on the controllable generation of content (e.g., constructive methods (Togelius et al.,

2011)), and the consequence is that content tends to be similar. Generally, a trade-off between quality and diversity has to be taken: if the search space is too restricted, quality can be guaranteed, but the generated content will be similar. On the one hand, such gameplay qualities are required from the generated content in order to ensure the game is playable and balanced across players (e.g., in a competitive game). On the other hand, a core motivation of commercial PCG is the element of surprise it can elicit from players. For example, although astonishing for the quality of the generated planets, *No Mans Sky* (Hello Games, 2016)—a commercial game that featured PCG as its main selling point—has been criticized for the limited variety of the generated content. While constructive approaches have a limited range of possible outcomes, diversity-only approaches can lead to poorly suboptimal solutions or yield unusable content. Constrained novelty search by Liapis et al. (2015) solves this problem by subdividing the search space in two: feasible and infeasible. The algorithmic process rewards infeasible solutions based on their distance from the feasibility and then, once the solutions have reached the desired features, the algorithm explores the feasible space by rewarding novel solutions. Another interesting take on this problem is by Khalifa et al. (2018). In that work, another affirmed QD algorithm, MAP-Elites (Mouret and Clune, 2015) is applied to generate new levels for a *bullet hell* game. As in the previous example, a feasible-infeasible approach is employed, and the dimensions of dexterity and strategy are explored to generate challenging and playable levels. We can imagine applying surprise-based algorithms to augment the creative potential of QD approaches. An initial investigation in this direction has been performed by Gravina et al. (2016a), where a set of balanced and surprising weapons for first-person shooters were evolved through a quality diversity approach named *constrained surprise search*.

Constrained Surprise Search

Constrained surprise search is inspired by earlier work on creating sets of diverse artifacts (Gravina and Loiacono, 2015), and applies surprise search on the task of procedural content generation. In particular, the goal is generating pairs of weapons for a competitive first-person shooter game: the two weapons must be usable and balanced between them, but also exhibit surprising behavioral properties (i.e., different weapon pairs would allow different types of gameplay or strategies to emerge). Towards that end, constraints on usability and balance are satisfied via a feasible-infeasible two-population approach (FI-2pop GA) which guides infeasible content towards feasibility (Kimbrough et al., 2008). In the feasible population, however, the weapon pairs evolve towards surprising behaviors, i.e., behaviors that were not predicted based on the previous generations. 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.

The weapons are used in the commercially successful *Unreal Tournament III* (Epic Games 2007) game (UT3). Besides its commercial appeal, UT3 has well-designed game levels and AI modules which allow for simulations of game matches in order to derive behavioral properties of the weapons. Weapons in UT3 are already quite diverse, which allows the surprise-based algorithm to explore different sets of parameters such as bouncing bullets, grenades affected by gravity, or exploding projectiles. The two weapons evolved in this scenario are tested by two AI-controlled agents competing for the highest number of kills in UT3. Simulations are used to create a map of the death locations of each player; these act as *behavioral characteristics* of the weapons and are used to assess unexpected

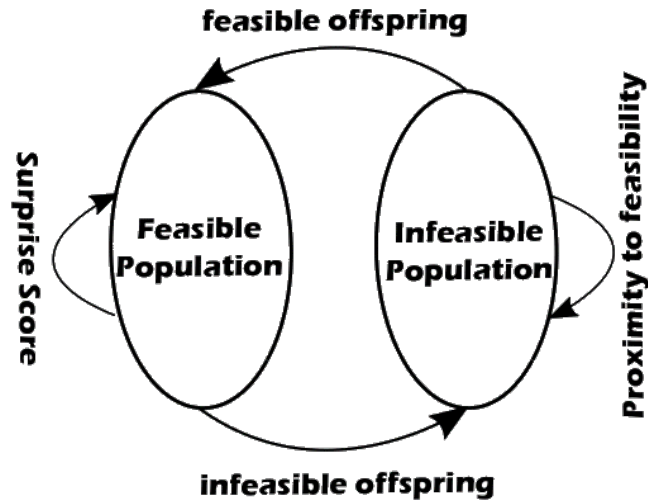


Figure 7.4: **Surprise-based FI-2pop**. Diagram of the two-population genetic algorithm augmented by surprise search used in (Gravina et al., 2016a).

behaviors in the surprise search algorithm.

Algorithm: this approach fuses the properties of FI-2pop constrained optimization (Kimbrough et al., 2008) with surprise search (Gravina et al., 2016b), by using the latter to evolve the feasible population. The proposed algorithm uses two populations which evolve towards different goals, one for the feasible individuals and the other for the infeasible ones (Fig. 7.4). In the genotype, each weapon is represented by 11 parameters with different value ranges and in-game properties; the generator evolves pairs of weapons (one per player in a deathmatch FPS game)

There are certain playability requirements for the generated weapons: balance, effectiveness and safety. Each of these properties can be evaluated as a scalar value, via heuristics discussed below, based on simulations between AI controlled agents. A pair of weapons is considered playable (i.e. feasible) if each property is above a specific threshold. Moreover, for infeasible individuals the heuristics can be used to derive the distance from feasibility with regards to each constraint. *Balance* is computed as the Shannon Entropy (Shannon, 2001) of the kills obtained by the two agents; *effectiveness* is calculated by dividing the total number of kills obtained in the simulation by the maximum *score limit*; *safety* is computed as the exponential of 0.9 elevated for the number of suicides (i.e. a death which was not scored as another player’s kill). The feasibility constraint is satisfied if balance is ≥ 0.9 , effectiveness is ≥ 1 (i.e. if exactly 20 kills are scored) and safety is ≥ 0.9 (i.e. if there’s at most one suicide). The rationale for the strict thresholds for effectiveness and safety are to avoid creating sparse heatmaps of death locations (due to low effectiveness) or death locations originating from suicides (due to low safety).

The feasible population contains individuals which satisfy all constraints listed above, while the infeasible population contains individuals which have at least one of safety, balance and efficiency below the minimal threshold. The infeasible population assigns its members a fitness equal to $balance + effectiveness + safety$, regardless of whether some of the values of these properties are above the feasibility threshold. This favors individuals which satisfy more constraints to others which satisfy no constraints, although some averaging

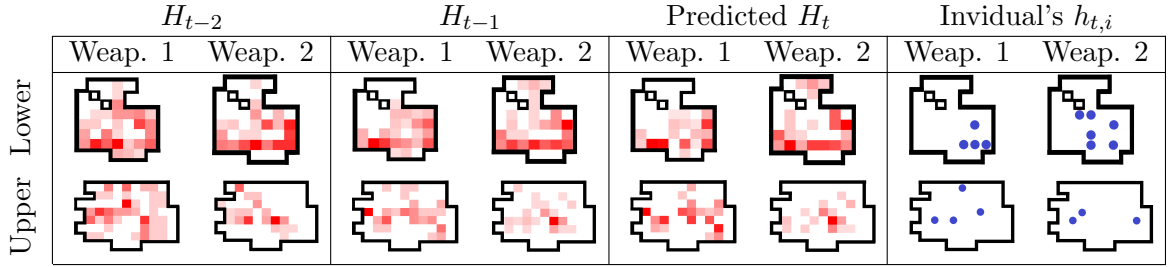


Figure 7.5: **An example of the prediction model of constrained surprise search.** The first two sets of heatmaps are computed in the last two generations, H_{t-2} and H_{t-1} ; the death location density is always normalized per floor. Using linear interpolation, the difference $H_{t-1} - H_{t-2}$ is computed and applied to H_{t-1} to derive the predicted current population's H_t truncated to $[0, 1]$. An individual's death locations are mapped to H_t to calculate the surprise score.

artifacts may occur. Unlike traditional FI-2pop approaches, the infeasible population attempts to maximize this value, as the three properties act as objectives (with minimal value constraints).

In the feasible population, surprise search attempts to deviate from predicted behavioral trends of the current population. *Behavior* of a weapon is considered to be the playtraces of the player who wields it, and in particular the locations where their opponent died in this one-versus-one deathmatch game. Since the genotype contains two weapons and the level used for the simulation consists of two floors, this creates a total of 4 heatmaps of death locations of each player. These *heatmaps* assign each death on a tile of a low-resolution grid (10 by 13 tiles per floor), incrementing the value (or heat) of that tile; example heatmaps are shown in Fig. 4.1. Note that heatmaps are normalized to a range of $[0, 1]$ based on the maximum heat value of each map (i.e. per floor and per player).

Surprise search attempts to deviate, therefore, from the expected heatmaps of this generation: i.e., have death locations which are unexpected based on the current evolutionary trends. Surprise search focuses on diverging from predictions \mathbf{p} (see Eq. 3.1) of the current population, calculated by observing the previous generations' behavioral changes. We use only the populations of the last two generations ($h = 2$; Eq. 3.1) to predict the current population, applying a linear interpolation (m is a linear regression model in Eq. 3.1). The model, m , considers the population as a whole ($k_{SS} = 1$; Eq. 3.1). In short, when predicting the heatmaps H_t for a population at generation t , the heatmaps of the population at $t - 2$ (H_{t-2}) is subtracted from those of the population at $t - 1$ (H_{t-1}) to calculate ΔH . The prediction of H_t is obtained by adding ΔH to H_{t-1} , ensuring that its values fall within $[0, 1]$ in all 4 heatmaps. Figure 7.5 illustrates this procedure.

The primary goal was to discover feasible and diverse content via constrained surprise search, thereby achieving quality diversity. The results obtained indicate that constrained surprise search tends to evolve diverse pairs of weapons, which have unexpected in-game uses. The FI-2pop paradigm also allows this method to discover feasible individuals quickly and consistently. As an example, Figure 7.6 shows a couple of weapons generated by constrained surprise search.

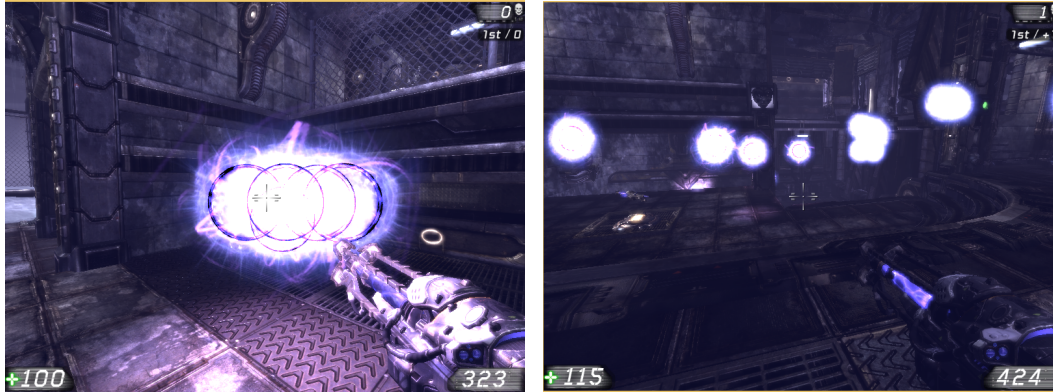


Figure 7.6: **Two example weapons created by constrained surprise search.** The weapons achieve quality diversity as they both respect balance constraints (quality) and, at the same time, maximize their surprise score (diversity). The weapon on the left creates ‘mines’ around the map: its bullets are extremely slow, with a large blast area (explosive, high collision radius) and they can also bounce on walls or the level’s floor. Moreover, these ‘mines’ are fired in clusters (high shot cost) thus costing a lot of ammo (of which the weapon has little): the first weapon requires its wielder to move around the level, laying ‘mines’ in chokepoints when the other player is nearby. Meanwhile, the weapon on the right is very similar to a rifle: high-damage fast bullets which shoot straight (trivial gravity effects) with a very low collision radius, thus requiring precise aiming. Unlike traditional rifles, however, the weapon’s bullets have some explosive qualities.

7.2.2 Surprise Search for Reinforcement Learning

Another potential domain for surprise-based search is Reinforcement Learning (Sutton and Barto, 2018). The idea of applying the intuition behind quality diversity within reinforcement learning is not new. As described in Chapter 2, we can argue that intrinsic motivation shares some common ground with the notion of quality diversity. However, the idea of applying stochastic search coupled with reinforcement learning has largely unexplored potential and only recently it has been explored with some promising results (Conti et al., 2017). For instance, Colas et al. (2018) introduce the idea of *decoupling* exploration and exploitation in RL.

Goal Exploration Process - Policy Gradient is composed of two different phases. During the first phase, a method called *goal exploration process* (Forestier et al., 2017) is used to explore efficiently the state-action space and populate a replay buffer. In the second phase, a Deep Reinforcement Learning (Deep RL) algorithm (Mnih et al., 2015) uses the replay buffer to sample diverse and efficient policies for the given problem. Therefore, this approach exploits the advantages of the two paradigms: while quality diversity is an efficient method to explore search spaces and circumvent deceptive environments, deep reinforcement learning is used to fine-tune the policies found during the first phase. Interestingly, Ecoffet et al. (2018) argue that *backtracking* to previously visited states is of fundamental importance to solve sparse reward systems. We argue that the unique properties of surprise search shown in this work—orthogonality with novelty and value, emerging backtracking behaviour—can be helpful to address difficult problems in RL. A surprise-based approach could be a useful

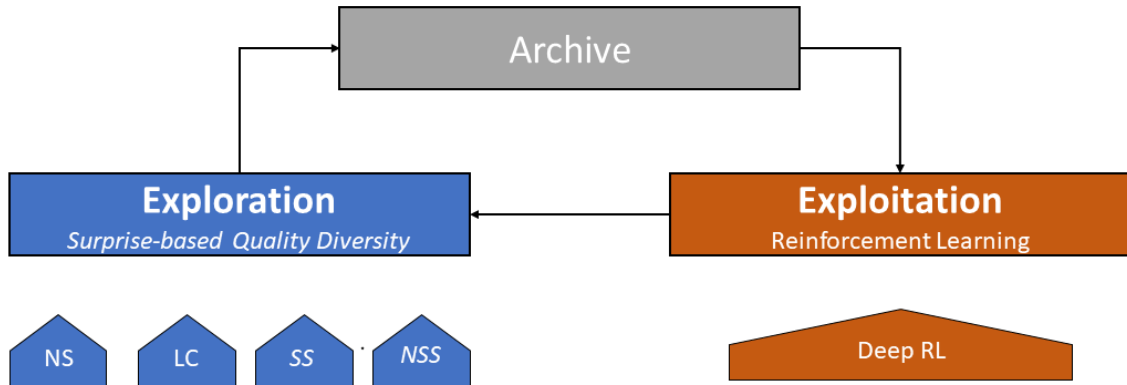


Figure 7.7: **Surprise-based Reinforcement Learning**: the flowchart illustrates the two key phases involved in reinforcement learning through surprise search. The figure include the general principles of the exploration-exploitation trade-off (bold) and the algorithmic contributions of this thesis (in italics).

aid to increase the exploration capabilities of reinforcement learning algorithms.

Inspired by Colas et al. (2018), we can imagine transferring the methodology described in this thesis in the context of RL. A possible implementation can use surprise-based quality diversity as a way to find unexpected behaviours in the state-action space (*exploration* phase) and store them in a separated archive, which can be used in the second phase (*exploitation* phase) as Fig. 7.7 shows. Surprise search, in particular, can explore the state-action space in an uninformed manner (as in divergent search), or explore both high-performing and diverse solutions as for surprise-based QD. The explored behaviours are stored subsequently in a separated archive as pairs of policy parameters and outcomes (based on the chosen behaviour characterization). This archive of pairs is then used in the second phase, where we populate the replay buffer used by the Deep RL algorithm, which can be implemented as plain Deep RL or coupled with other approaches. Extending on (Colas et al., 2018), we can imagine to alternate the two phases every N epochs, in order to fully exploit the exploration capabilities of the surprise-based QD algorithm and make the overall process more dynamic.

The introduction of surprise-based RL can broaden the scope of surprise search to reinforcement learning environments. Thanks to the fusion of surprise-based evolutionary approaches with Deep RL, we can envision to solve deceptive problems and sparse reward domains. For instance, there are a number of deceptive problems where Deep RL algorithms have difficulties in finding optimal solutions, such as games with deceptive reward systems (Anderson et al., 2018). Deceptive games are games where the reward system is designed to lead the agent away from the globally optimal policy. Anderson et al. (2018) introduced a set of deceptive games in the General Video Game Artificial Intelligence framework (Perez-Liebana et al., 2019), where they implement a number of games with different types of deception. In particular, they introduce several categories of deception, such as the *Greed Trap*, where local reward gradients lead to suboptimal solutions or the *Smoothness Trap* where the global optimum is surrounded by low performing areas of the solution space. A possible solution to this problem is to aid the exploration of the RL agent with surprise search: we argue that the surprise-based RL algorithm can be a promising approach for these deceptive problems, as surprise search may help the RL algorithm to overcome the

deceptiveness of the reward system across different deceptive categories.

7.3 Summary

This chapter has offered an overview of the main insights gained by testing the proposed surprise-based algorithms in several benchmarks. Based on the evidence collected, we argue that surprise is an effective evolutionary tool and, sided with novelty search and local competition, it contributes the search for high-quality solutions in an efficient and robust way, especially in deceptive problems. We performed a number of experiments to validate the proposed algorithms thoroughly, but some limitations have been found in the design choices and experiments performed. Chiefly, these limitations are based on two main aspects: the generalization of the reported results and the design of the algorithms' subcomponents. To address these limitations we proposed two main directions for future work. First, given the general formulation described in Chapter 3, we proposed several extensions for the surprise-based algorithms, e.g., in terms of prediction model or behaviour characterization used. Second, we proposed to corroborate the algorithms introduced in this dissertation in other domains. In particular, we can envision to use surprise-based search to generate unexpected content for games and to aid the exploration capabilities of deep reinforcement learning algorithms.

Appendices

Appendix A

Experimental Parameters

A.1 Parameters for Maze Navigation Experiments

In the experiments conducted in this dissertation, surprise search and surprise-based algorithms have been implemented as an extension of NEAT. To facilitate a fair comparison, all the tested algorithms are tested with the same NEAT parameters. The software package employed is based on the original Novelty Search C++ implementation provided by Lehman and Stanley (2011a). The source code of the described framework is available online¹. Table A.1 shows the parameters used in experiments reported in this work. Additionally Table A.2, Table A.3 and Table A.4 show the parameters chosen for the experiments conducted in in Chapter 4, Chapter 5 and Chapter 6. A description of the parameters is also provided:

- Population size: the number of artificial neural networks evolved.
- c_1 : weight for the excess genes used in the compatibility metric computation (Stanley and Miikkulainen, 2002).
- c_2 : weight for the disjoint genes used in the compatibility metric computation (Stanley and Miikkulainen, 2002).
- c_3 : weight for the connection strength used in the compatibility metric computation (Stanley and Miikkulainen, 2002).
- C_t : compatibility threshold (Stanley and Miikkulainen, 2002). This threshold is used to check if two individuals are part of the same species.
- Add Link probability: this parameter specify how often a mutation can add a link between two nodes of the ANN.
- Add Node probability: this parameter specify how often a mutation can add a new node in the ANN.
- Initial Archive Threshold: this parameter specify the minimum novelty value to enter the novelty archive. This parameter is used only for the novelty search based algorithms.
- Simulation steps: the number of simulation steps run in the maze for the wheeled robot.

¹<https://gitlab.com/2factor/QDSurprise>

Table A.1: **NEAT parameter setting.** This table shows the NEAT parameters used for the maze navigation experiments conducted in Chapter 4, Chapter 5 and Chapter 6.

Parameter	Value
Population size	250
c_1	1.0
c_2	1.0
c_3	3.0
C_t	<i>variable</i>
Probability Add Link	0.1
Probability Add Node	0.005
Initial Archive Threshold	6.0

Table A.2: **Maze navigation parameters.** Parameters used for the experiments reported in Chapter 4.

Parameter	Medium	Hard	Very Hard	Extremely Hard	Generated Mazes
Simulation steps	300	300	500	1000	300
Generations	300	300	1000	1000	600
n_{NS}	15	15	15	10	15
k_{SS}	200	100	200	220	200
h	1	1	2	2	2

- Generations: maximum number of generations allowed for the experiment.
- n_{NS} : a novelty search parameter that establishes the number of nearest neighbours to consider when computing the novelty measure (See Eq. 2.3).
- k_{SS} : a surprise search parameter that define the locality of the behaviour used to compute the predictions (See Eq. 3.1).
- h : parameter that defines the number of previous generations used by surprise search (See Eq. 3.1).
- m : parameter that defines prediction model used by surprise search (See Eq. 3.1).
- \mathbf{p} : predictive behaviours used by surprise search (See Eq. 3.1).
- d_{SS} : domain-dependent measure of behavioural distance used by surprise search (See Eq. 3.2).
- n_{SS} : a surprise search parameter that define the number of closest predicted behaviour to consider to compute the surprise score (See Eq. 3.2).
- λ : a novelty-surprise search parameter that defines the weight used to compute the linear aggregation between novelty and surprise (See Eq. 3.3).

A.2 Parameters for Soft Robot Evolution Experiments

The experiments reported with the Soft Robot Evolution testbed were run with an extension of the source code provided by Methenitis et al. (2015) and Cheney et al. (2013). The source

Table A.3: **Maze navigation parameters.** Parameters used for the experiments reported in Chapter 5.

Parameter	Medium	Hard	Very Hard	Extremely Hard	Generated Mazes
Simulation steps	300	300	500	1000	300
Generations	300	300	1000	1000	600
n_{NS}	15	15	15	10	15
k_{SS}	100	50	200	220	200
h	2	2	2	2	2
λ	0.6	0.6	0.3	0.6	0.6

Table A.4: **Maze navigation parameters.** Parameters used for the experiments reported in Chapter 6; n_{LC}^1 is used for NS-LC, NSS-LC and NS-SS-LC, n_{LC}^2 is used for SS-LC.

Parameter	Medium	Hard	Very Hard	Extremely Hard	Generated Mazes
Simulation steps	300	300	500	1000	300
Generations	300	300	1000	1000	600
n_{NS}	15	15	15	15	15
k_{SS}	200	200	200	200	200
h	2	2	2	2	2
λ	0.7	0.7	0.7	0.7	0.7
n_{LC}^1	5	5	5	5	5
n_{LC}^2	10	10	10	10	10

code of the described algorithms is available online². Table A.5 shows the parameters used for the experiments conducted in this work. A detailed description of these parameters can be found in (Stanley, 2007) and (Cheney et al., 2013).

²<https://gitlab.com/2factor/Soft-Robots-SurpriseSearch>

Table A.5: **Soft Robot Evolution parameter setting.**

	Parameter	Value
VoxCad	Gravity	-27.6 m/s^2
	Simulation time	0.4 seconds
	Signal rate	40Hz
CPPN-NEAT	Population size	30
	Generation	1000
	c_1	2.0
	c_2	2.0
	c_3	1.0
	C_t	6.0
	Probability Add Link	0.05
	Probability Add Node	0.03
	Compatibility Modifier	0.3
	Species Size Target	8.0
	Dropoff Age	15.0
	Age Significance	1.0
	Survival Threshold	0.2
	Mutate Link Weights Probability	0.8
	Mutate Only Probability	0.25
	Mutate Link Probability	0.1
	Smallest Species Size With Elitism	5.0
	Mutation Power	2.5
	Adult Link Age	18.0
	Force Copy Generation Champion	1.0
	Generation Dump Modulo	1.0
	Extra Activation Functions	1.0
	Signed Activation	1.0
	Extra Activation Updates	9.0
	n_{NS}	10
	k_{SS}	15
	h	4
λ	0.6	

Appendix B

Maze Navigation Generated Mazes

Table B.1: **60 generated mazes**: set introduced in Chapter 4. The starting position (grey filled circle) is at the bottom left corner; the goal position (black empty circle) is at the top right corner.

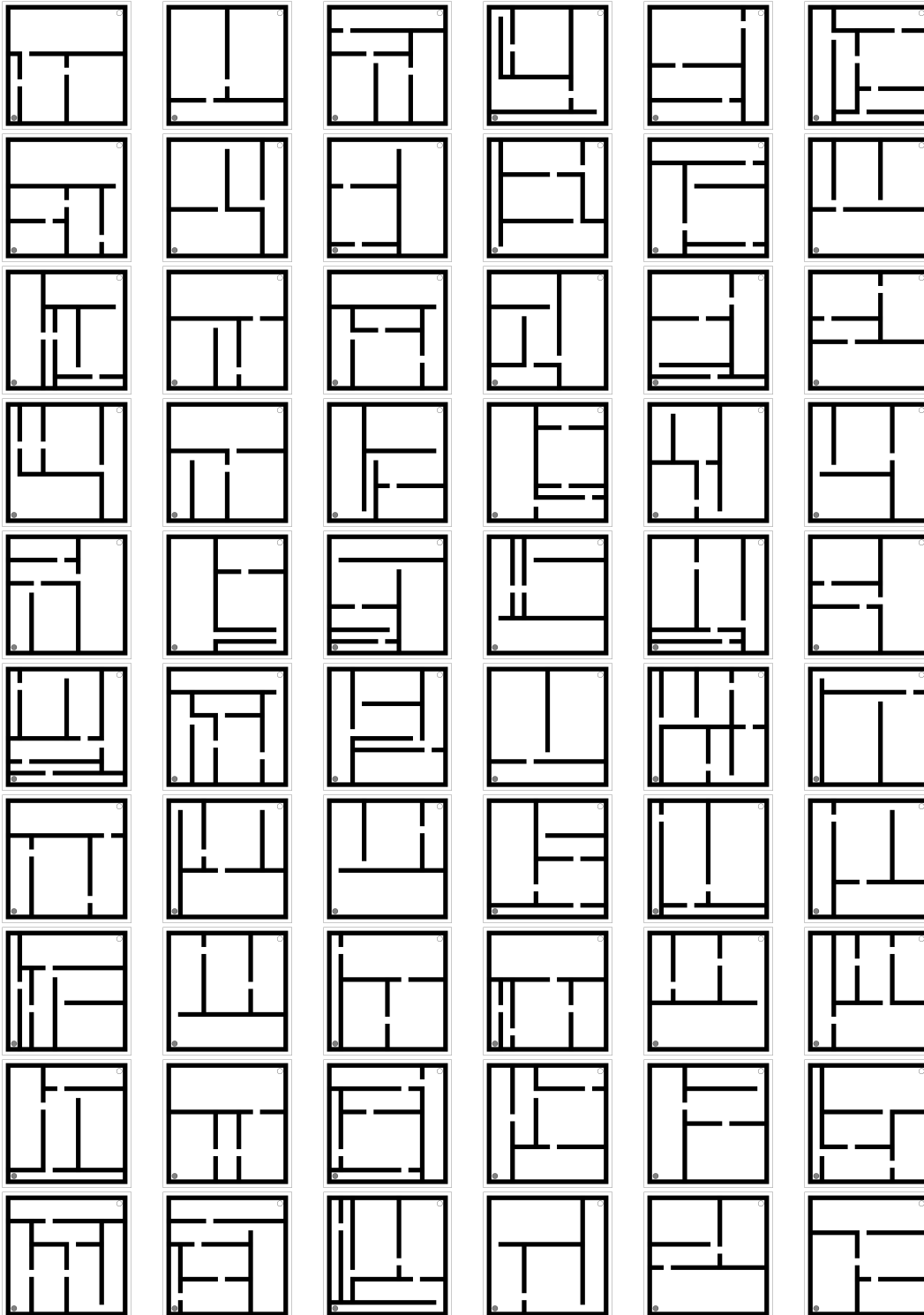
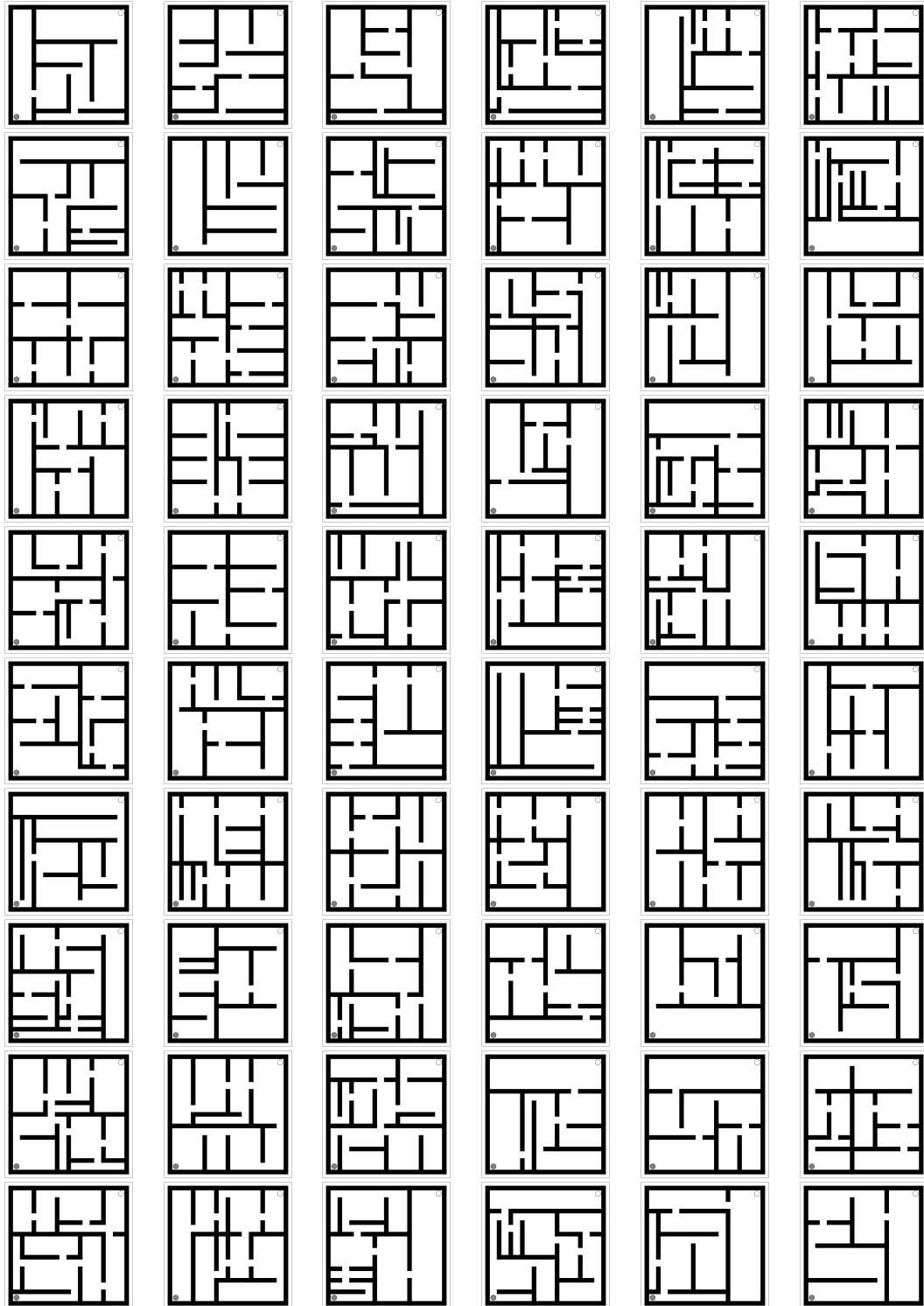


Table B.2: **60 generated mazes**: set introduced in Chapter 6. The starting position (grey filled circle) is at the bottom left corner; the goal position (black empty circle) is at the top right corner.



Appendix C

Soft Robot Behavioural and Structural Analysis

Table C.1: **Feature maps**: feature maps produced by the four methods, across the 8 resolutions considered. White bins do not have any robots, while colored bins denote the fitness of the best individual (blue for low fitness, red for high fitness).

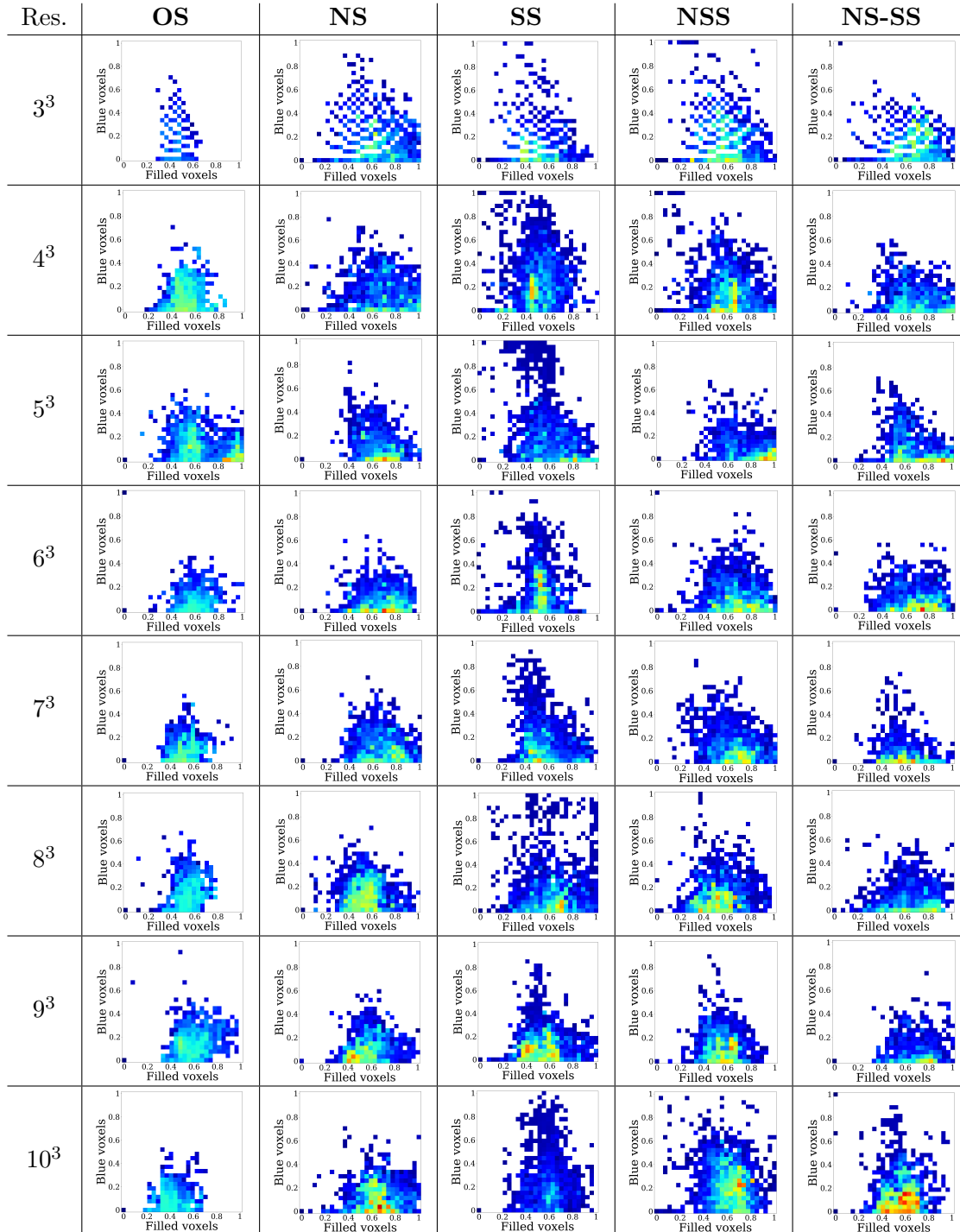


Table C.2: **Behavioural analysis:** behavioural performance metrics as the mean values of 90 independent runs (95% confidence interval in parentheses). Bold values are significantly different from all other methods.

3^3	Objective	Novelty	Surprise
Trajectory length	7.98 (0.48)	9.38 (0.20)	9.75 (0.22)
Deviation	0.19 (0.05)	0.39 (0.08)	0.35 (0.07)
Max velocity	44.86 (2.01)	44.77 (1.08)	46.69 (1.28)
Mean velocity	19.96 (1.21)	23.47 (0.50)	24.31 (0.56)
4^3	Objective	Novelty	Surprise
Trajectory length	8.33 (0.16)	10.71 (0.33)	11.28 (0.31)
Deviation	0.16 (0.02)	0.49 (0.08)	0.49 (0.08)
Max velocity	39.14 (0.76)	47.86 (1.72)	47.68 (1.36)
Mean velocity	20.84 (0.40)	26.79 (0.84)	28.20 (0.79)
5^3	Objective	Novelty	Surprise
Trajectory length	7.64 (0.44)	12.91 (0.33)	12.14 (0.36)
Deviation	0.22 (0.03)	0.59 (0.06)	0.63 (0.07)
Max velocity	34.14 (2.28)	57.24 (1.27)	53.57 (1.81)
Mean velocity	19.11 (1.11)	32.28 (0.84)	30.37 (0.92)
6^3	Objective	Novelty	Surprise
Trajectory length	8.44 (0.46)	12.52 (0.29)	12.45 (0.46)
Deviation	0.27 (0.04)	0.54 (0.06)	0.61 (0.08)
Max velocity	36.14 (2.45)	53.75 (1.11)	52.62 (1.56)
Mean velocity	21.11 (1.15)	31.30 (0.72)	31.13 (1.16)
7^3	Objective	Novelty	Surprise
Trajectory length	7.71 (0.48)	12.09 (0.36)	11.54 (0.30)
Deviation	0.19 (0.04)	0.58 (0.08)	0.58 (0.07)
Max velocity	29.01 (2.04)	49.54 (1.30)	47.37 (1.46)
Mean velocity	19.27 (1.20)	30.22 (0.91)	28.85 (0.76)
8^3	Objective	Novelty	Surprise
Trajectory length	8.93 (0.68)	12.93 (0.38)	11.70 (0.39)
Deviation	0.23 (0.03)	0.58 (0.07)	0.55 (0.09)
Max velocity	31.13 (2.31)	45.98 (1.50)	42.78 (1.41)
Mean velocity	22.31 (1.71)	29.81 (0.94)	29.23 (0.97)

Table C.3: **Behavioural analysis:** behavioural performance metrics as the mean values of 90 independent runs (95% confidence interval in parentheses). Bold values are significantly different from all other methods.

9^3	Objective	Novelty	Surprise
Trajectory length	7.86 (0.51)	11.92 (0.38)	11.46 (0.37)
Deviation	0.21 (0.03)	0.51 (0.07)	0.57 (0.08)
Max velocity	27.46 (1.75)	44.03 (1.33)	40.78 (1.42)
Mean velocity	19.65 (1.26)	29.81 (0.96)	28.66 (0.92)
10^3	Objective	Novelty	Surprise
Trajectory length	9.08 (0.68)	11.60 (0.34)	11.34 (0.38)
Deviation	0.22 (0.04)	0.55 (0.08)	0.41 (0.06)
Max velocity	30.45 (2.25)	41.73 (1.38)	40.32 (1.41)
Mean velocity	22.70 (1.70)	29.01 (0.85)	28.34 (0.95)

Appendix D

Computational Effort

While the analysis of computational effort has been focused on the efficiency and robustness of the considered algorithms, it is worth investigating the computation time required to run the surprise-based evolutionary algorithms proposed in this work and compared them to two baselines, novelty search, and objective search. We report the obtained results in terms of CPU time per generation (in seconds) recorded in one indicative run for each algorithm in the two domains used in this thesis. Table D.1 and Table D.2 show the results in the maze navigation domain and in the soft robot domain respectively. All the experiments have been run independently in a 2.4 GHz 40-cores workstation.

Table D.1: **Maze Navigation Computational Effort.** CPU time (in seconds) per generation of one indicative run in the maze navigation domain.

	Medium	Hard	Very Hard	Extremely Hard
OS	0.210	0.287	0.461	1.410
NS	0.309	0.342	0.475	0.675
SS	0.234	0.235	0.952	0.967
NSS	0.423	0.491	0.734	1.201
NS-SS	0.775	0.787	0.960	1.301
NS-LC	0.835	0.828	0.929	1.543
SS-LC	0.739	0.746	0.899	2.075
NSS-LC	1.021	1.182	1.180	1.507
NS-SS-LC	1.034	1.158	1.133	1.644

Table D.2: **Soft Robot Computational Effort:** CPU time (in seconds) per generation of one indicative run in the soft robot domain.

	OS	NS	SS	NSS	NS-SS
3x3x3	1.872	4.419	5.761	5.495	6.487
4x4x4	9.707	11.659	12.128	12.647	13.164
5x5x5	18.070	22.052	22.999	22.294	25.496
6x6x6	28.551	40.387	31.591	38.291	32.850
7x7x7	26.879	64.178	53.246	50.220	58.146
8x8x8	74.586	91.975	83.791	85.687	79.129
9x9x9	90.697	117.067	127.328	112.114	121.617
10x10x10	162.423	156.618	140.949	156.700	172.854

Acronyms

- Deep RL** Deep Reinforcement Learning. 130
- EC** Evolutionary Computation. 1
- FI-2pop GA** Feasible-infeasible Two-population Genetic Algorithm. 127
- MOEAs** Multi-objective Evolutionary Algorithms. 18
- NEAT** Neuroevolution of Augmenting Topologies. 21
- NS** Novelty Search. 2
- NS-LC** Novelty Search with Local Competition. 2
- NS-SS** Novelty Search-Surprise Search. 37
- NS-SS-LC** Novelty Search-Surprise Search-Local Competition. 37
- NSGA-II** Non-dominated Sorting Genetic Algorithm II. 19
- NSS** Novelty-Surprise Search. 37
- NSS-LC** Novelty-Surprise Search with Local Competition. 37
- PCG** Procedural Content Generation. 125
- PCGML** Procedural Content Generation via Machine Learning. 125
- QD** Quality Diversity. 2
- SS** Surprise Search. 2
- SS-LC** Surprise Search with Local Competition. 37

Bibliography

- Timo Aaltonen, J. Adelman, T. Akimoto, M.G. Albrow, B. Álvarez González, S. Amerio, D. Amidei, A. Anastassov, A. Annovi, J. Antos, et al. Measurement of the top-quark mass with dilepton events selected using neuroevolution at cdf. *Physical Review Letters*, 102(15):152001, 2009. — Cited on page 22.
- Christoph Adami, Charles Ofria, and Travis C. Collier. Evolution of biological complexity. *Proceedings of the National Academy of Sciences*, 97(9), 2000. — Cited on page 1.
- Damien Anderson, Matthew Stephenson, Julian Togelius, Christoph Salge, John Levine, and Jochen Renz. Deceptive games. In *International Conference on the Applications of Evolutionary Computation*, pages 376–391. Springer, 2018. — Cited on page 131.
- 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. — Cited on page 24.
- Andrew Barto, Marco Mirolli, and Gianluca Baldassarre. Novelty or surprise? *Frontiers in psychology*, 4:907, 2013. — Cited on pages xi, 2, and 14.
- Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016. — Cited on pages 30 and 31.
- Peter Bentley and Sanjeev Kumar. Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann Publishers Inc., 1999. — Cited on page 20.
- Daniel E. Berlyne. Conflict, arousal, and curiosity. *McGraw-Hill series in psychology*, 1960. — Cited on page 13.
- S. Bertelli, L. M. Chiappe, and C. Tambussi. A new phorusrhacid (aves: Cariamae) from the middle miocene of Patagonia, Argentina. *Journal of Vertebrate Paleontology*, 27(2), 2007. — Cited on page 4.
- Margaret A. Boden. *The Creative Mind: Myths and Mechanisms*. Routledge, 2004. — Cited on pages 2 and 11.
- Josh C. Bongard and Rolf Pfeifer. Evolving complete agents using artificial ontogeny. In *Morpho-functional Machines: The new species*, pages 237–258. Springer, 2003. — Cited on page 35.

- Yossi Borenstein and Riccardo Poli. Fitness distributions and ga hardness. In *International Conference on Parallel Problem Solving from Nature*, pages 11–20. Springer, 2004. — Cited on page 23.
- Heinrich Braun and Joachim Weisbrod. Evolving neural feedforward networks. In *Artificial Neural Nets and Genetic Algorithms*, pages 25–32. Springer, 1993. — Cited on page 21.
- Dimo Brockhoff, Tobias Friedrich, Nils Hebbinghaus, Christian Klein, Frank Neumann, and Eckart Zitzler. Do additional objectives make a problem harder? In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2007. — Cited on page 24.
- Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018. — Cited on page 31.
- Maxim Buzdalov and Vladimir Parfenov. Various degrees of steadiness in nsga-ii and their influence on the quality of results. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2015. — Cited on page 122.
- Luigi Cardamone, Georgios N. Yannakakis, Julian Togelius, and Pier Luca Lanzi. Evolving interesting maps for a first person shooter. In *European Conference on the Applications of Evolutionary Computation*, pages 63–72. Springer, 2011. — Cited on page 33.
- Alastair Channon et al. Passing the alife test: Activity statistics classify evolution in geb as unbounded. In *ECAL*, pages 417–426. Springer, 2001. — Cited on page 1.
- Nick Cheney, Robert MacCurdy, Jeff Clune, and Hod Lipson. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2013. — Cited on pages 34, 35, 61, 64, 85, 119, 120, 122, 136, and 137.
- Jeff Clune and Hod Lipson. Evolving three-dimensional objects with a generative encoding inspired by developmental biology. In *ECAL*, pages 141–148, 2011. — Cited on page 22.
- Carlos A. Coello Coello, Gary B. Lamont, David A. Van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer, 2007. — Cited on pages 19 and 42.
- Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms. *arXiv preprint arXiv:1802.05054*, 2018. — Cited on pages 130 and 131.
- Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. *arXiv preprint arXiv:1712.06560*, 2017. — Cited on page 130.
- Michael Correll and Jeffrey Heer. Surprise! bayesian weighting for de-biasing thematic maps. *IEEE Transactions on Visualization Computer Graphics*, 23(1):651–660, 2017. — Cited on pages 12 and 121.
- Giuseppe Cuccu and Faustino Gomez. When novelty is not enough. In *European Conference on the Applications of Evolutionary Computation*, pages 234–243. Springer, 2011. — Cited on page 41.

- Antoine Cully and Yiannis Demiris. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22, 2018. — Cited on pages vi, 26, 28, and 30.
- Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015. — Cited on pages 27, 28, and 123.
- Y. Davidor. Epistasis variance: A viewpoint on ga-hardness. In *Foundations of Genetic Algorithms*. Morgan Kaufmann, 1991. — Cited on pages 17 and 23.
- Kenneth A. De Jong. *Evolutionary computation: a unified approach*. MIT press, 2006. — Cited on page 1.
- Kalyanmoy Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7, 1999. — Cited on page 24.
- Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001. — Cited on pages 18, 28, 41, and 42.
- Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4): 577–601, 2014. — Cited on page 122.
- Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. A. M. T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. — Cited on pages 19, 29, 42, 43, 44, 74, and 86.
- Emanuel Donchin. Surprise!... surprise? *Psychophysiology*, 18(5):493–513, 1981. — Cited on page 12.
- Juan J. Durillo, Antonio J. Nebro, Francisco Luna, and Enrique Alba. On the effect of the steady-state selection scheme in multi-objective genetic algorithms. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 183–197. Springer, 2009. — Cited on page 122.
- Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Montezuma’s revenge solved by go-explore, a new algorithm for hard-exploration problems (sets records on pitfall, too). <http://eng.uber.com/go-explore/>, 2018. — Cited on page 130.
- Paul Ekman. An argument for basic emotions. *Cognition & emotion*, 6(3-4), 1992. — Cited on pages 6 and 12.
- Jeffrey L. Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993. — Cited on page 24.
- Leon Festinger. *A theory of cognitive dissonance*, volume 2. Stanford university press, 1962. — Cited on page 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. — Cited on page 24.

- David B. Fogel. Phenotypes, genotypes, and operators in evolutionary computation. In *Evolutionary Computation, 1995., IEEE International Conference on*, volume 1, page 193. IEEE, 1995. — Cited on page 20.
- Sébastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*, 2017. — Cited on page 130.
- Justin Fu, John Co-Reyes, and Sergey Levine. Ex2: Exploration with exemplar models for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2577–2587, 2017. — Cited on page 31.
- Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. Data-efficient design exploration through surrogate-assisted illumination. *Evolutionary Computation*, 26(3):381–410, 2018. — Cited on page 118.
- Pascal Godefroit, Andrea Cau, Dong-Yu Hu, François Escuillié, Wenhao Wu, and Gareth Dyke. A jurassic avialan dinosaur from china resolves the early phylogenetic history of birds. *Nature*, 498(7454):359–362, 2013. — Cited on pages v and 5.
- David E. Goldberg. Simple genetic algorithms and the minimal deceptive problem. In *Genetic Algorithms and Simulated Annealing, Research Notes in Artificial Intelligence*. Morgan Kaufmann, 1987. — Cited on pages 1 and 23.
- David E. Goldberg and John H Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2), 1988. — Cited on page 1.
- David E. Goldberg, Jon Richardson, et al. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Hillsdale, NJ: Lawrence Erlbaum, 1987. — Cited on pages 1, 24, and 28.
- Jorge Gomes, Pedro Mariano, and Anders Lyhne Christensen. Devising effective novelty search algorithms: A comprehensive empirical study. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2015. — Cited on page 27.
- Jorge Gomes, Pedro Mariano, and Anders Lyhne Christensen. Novelty-driven cooperative coevolution. *Evolutionary computation*, 25(2):275–307, 2017. — Cited on page 121.
- Faustino J. Gomez and Risto Miikkulainen. Solving non-markovian control tasks with neuroevolution. In *Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2*, pages 1356–1361. Morgan Kaufmann Publishers Inc., 1999. — Cited on pages 20 and 21.
- M. Gong, J. Liu, H. Li, Q. Cai, and L. Su. A multiobjective sparse feature learning model for deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 26(12):3263–3277, 2015. — Cited on page 28.
- Goren Gordon and Ehud Ahissar. Hierarchical curiosity loops and active sensing. *Neural Networks*, 32:119–129, 2012. — Cited on page 31.

- Kazjon Grace and Mary Lou Maher. What to expect when you're expecting: The role of unexpectedness in computationally evaluating creativity. In *Proceedings of the International Conference on Computational Creativity*, 2014. — Cited on page 11.
- Kazjon Grace, Mary Lou Maher, Douglas Fisher, and Katherine Brady. Data-intensive evaluation of design creativity using novelty, value, and surprise. *International Journal of Design Creativity and Innovation*, 2014. — Cited on pages 2, 5, and 12.
- Kazjon Grace, Mary Lou Maher, Douglas Fisher, and Katherine Brady. Modeling expectation for evaluating surprise in design creativity. In *Design Computing and Cognition'14*, pages 189–206. Springer, 2015. — Cited on pages 2, 12, 14, 38, and 120.
- Daniele Gravina and Daniele Loiacono. Procedural weapons generation for unreal tournament iii. In *Games Entertainment Media Conference (GEM), 2015 IEEE*, pages 1–8. IEEE, 2015. — Cited on page 127.
- Daniele Gravina, Antonios Liapis, and Georgios N Yannakakis. Constrained surprise search for content generation. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2016a. — Cited on pages x, 16, 125, 127, and 128.
- Daniele Gravina, Antonios Liapis, and Georgios N. Yannakakis. Surprise search: Beyond objectives and novelty. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2016b. — Cited on pages 12, 25, 121, 122, and 128.
- Daniele Gravina, Antonios Liapis, and Georgios N. Yannakakis. Coupling novelty and surprise for evolutionary divergence. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017a. — Cited on pages 15, 16, and 122.
- Daniele Gravina, Antonios Liapis, and Georgios N. Yannakakis. Exploring divergence for soft robot evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2017b. — Cited on page 85.
- Daniele Gravina, Antonios Liapis, and Georgios N. Yannakakis. Surprise search for evolutionary divergence. *arXiv preprint arXiv:1706.02556*, 2017c. — Cited on pages 12, 25, and 121.
- Daniele Gravina, Antonios Liapis, and Georgios N. Yannakakis. Fusing novelty and surprise for evolving robot morphologies. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018. — Cited on page 15.
- Daniele Gravina, Ahmed Khalifa, Antonios Liapis, Julian Togelius, and Georgios N. Yannakakis. Procedural content generation through quality-diversity. In *Proceedings of the IEEE Conference on Games*, 2019a. — Cited on page 126.
- Daniele Gravina, Antonios Liapis, and Georgios N. Yannakakis. Quality diversity through surprise. *IEEE Transactions on Evolutionary Computation*, 23, 2019b. — Cited on pages 15, 16, and 19.
- Daniele Gravina, Antonios Liapis, and Georgios N. Yannakakis. Blending notions of diversity for map-elites. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2019c. — Cited on page 123.

- John J. Grefenstette. Deception considered harmful. In *Foundations of genetic algorithms*, volume 2, pages 75–91. Elsevier, 1993. — Cited on page 23.
- Frederic Gruau, Darrell Whitley, and Larry Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In *Proceedings of the 1st annual conference on genetic programming*, pages 81–89. MIT Press, 1996. — Cited on page 21.
- Haipeng Guo and William H. Hsu. Ga-hardness revisited. In *GECCO*, pages 1584–1585, 2003. — Cited on page 23.
- David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018. — Cited on page 119.
- Nikolaus Hansen. The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation*, pages 75–102. Springer, 2006. — Cited on page 125.
- John A. Hartigan and Manchek A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1): 100–108, 1979. — Cited on page 120.
- Erin J. Hastings, Ratan K. Guha, and Kenneth O. Stanley. Evolving content in the galactic arms race video game. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 241–248. IEEE, 2009. — Cited on page 22.
- Yuejun He and Jianxi Luo. The novelty ‘sweet spot’ of invention. *Design Science*, 3, 2017. — Cited on page 114.
- Jonathan Hiller and Hod Lipson. Dynamic simulation of soft heterogeneous objects. *arXiv preprint arXiv:1212.2845*, 2012. — Cited on pages 34 and 61.
- John Henry Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992. — Cited on page 16.
- Gregory 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. — Cited on pages 1 and 24.
- Gregory S. Hornby, Hod Lipson, and Jordan B. Pollack. Evolution of generative design systems for modular physical robots. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 4146–4151. IEEE, 2001. — Cited on page 35.
- Ian Douglas Horswill and Leif Foged. Fast procedural level population with playability constraints. In *AIIDE*, 2012. — Cited on page 126.
- Rein Houthoofd, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016. — Cited on page 31.
- J. Hu, E. Goodman, K. Seo, Z. Fan, and R. Rosenberg. The hierarchical fair competition (hfc) framework for sustainable evolutionary algorithms. *Evolutionary Computation*, 13(2), 2005. — Cited on pages 1 and 24.

- Marcus Hutter and Shane Legg. Fitness uniform optimization. *IEEE Transactions on Evolutionary Computation*, 10(5):568–589, 2006. — Cited on page 24.
- Laurent Itti and Pierre F. Baldi. Bayesian surprise attracts human attention. In *Advances in Neural Information Processing Systems*, pages 547–554, 2006. — Cited on pages 12 and 121.
- Yaochu Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing*, 9(1):3–12, 2005. — Cited on page 118.
- Yaochu Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011. — Cited on page 119.
- Yaochu Jin, Handing Wang, Tinkle Chugh, Dan Guo, and Kaisa Miettinen. Data-driven evolutionary optimization: An overview and case studies. *IEEE Transactions on Evolutionary Computation*, 2018. — Cited on pages 118 and 119.
- Terry Jones and Stephanie Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192, 1995. — Cited on page 23.
- Niels Justesen and Sebastian Risi. Automated curriculum learning by rewarding temporally rare events. In *IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2018. — Cited on page 31.
- Frédéric Kaplan and Pierre-Yves Oudeyer. Intrinsically motivated machines. In *50 years of artificial intelligence*, pages 303–314. Springer, 2007. — Cited on page 31.
- Daniel Karavolos, Antonios Liapis, and Georgios N. Yannakakis. Learning the patterns of balance in a multi-player shooter game. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, page 70. ACM, 2017. — Cited on pages 119 and 126.
- Daniel Karavolos, Antonios Liapis, and Georgios N. Yannakakis. Pairing character classes in a deathmatch shooter game via a deep-learning surrogate model. In *Proceedings of the FDG Workshop on Procedural Content Generation*, 2018a. — Cited on page 126.
- Daniel Karavolos, Antonios Liapis, and Georgios N. Yannakakis. Using a surrogate model of gameplay for automated level design. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2018b. — Cited on page 126.
- Daniel Karavolos, Antonios Liapis, and Georgios N. Yannakakis. A multi-faceted surrogate model for search-based procedural content generation. *IEEE Transactions on Games*, 2019. — Cited on page 126.
- S. A. Kauffman. Adaptation on rugged fitness landscapes. In *Lectures in the Sciences of Complexity*. Addison-Wesley, 1989. — Cited on page 23.
- Ahmed Khalifa, Scott Lee, Andy Nealen, and Julian Togelius. Talakat: Bullet hell generation through constrained map-elites. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018. — Cited on pages 125 and 127.

- Steven Orla Kimbrough, Gary J. Koehler, Ming Lu, and David Harlan Wood. On a feasible–infeasible two-population (fi-2pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch. *European Journal of Operational Research*, 190(2): 310–327, 2008. — Cited on pages 127 and 128.
- Joshua D. Knowles, Richard A. Watson, and David W. Corne. Reducing local optima in single-objective problems by multi-objectivization. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 269–283. Springer, 2001. — Cited on page 24.
- Teuvo Kohonen. *Self-organization and associative memory*, volume 8. Springer Science & Business Media, 2012. — Cited on page 13.
- Sylvain Koos, Jean-Baptiste Mouret, and Stéphane Doncieux. The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Transactions on Evolutionary Computation*, 17(1):122–145, 2013. — Cited on page 27.
- John R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4(2):87–112, 1994. — Cited on pages 16 and 17.
- Deepak Kulkarni and Herbert A Simon. The processes of scientific discovery: The strategy of experimentation. *Cognitive science*, 12(2):139–175, 1988. — Cited on pages 11 and 12.
- Max C. Langer, Martín D. Ezcurra, Oliver W.M. Rauhut, Michael J. Benton, Fabien Knoll, Blair W. McPhee, Fernando E. Novas, Diego Pol, and Stephen L. Brusatte. Untangling the dinosaur family tree. *Nature*, 551(7678):E1, 2017. — Cited on page 27.
- Michael S. Y. Lee, Andrea Cau, Darren Naish, and Gareth J. Dyke. Sustained miniaturization and anatomical innovation in the dinosaurian ancestors of birds. *Science*, 345(6196): 562–566, 2014. — Cited on page 4.
- Joel Lehman and Kenneth O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *Proceedings of the International Conference on Artificial Life*, 2008. — Cited on page 47.
- Joel Lehman and Kenneth O. Stanley. Revising the evolutionary computation abstraction: minimal criteria novelty search. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2010. — Cited on pages 16, 27, and 95.
- Joel Lehman and Kenneth O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2), 2011a. — Cited on pages vi, vii, 1, 2, 13, 14, 24, 25, 26, 32, 38, 39, 40, 42, 47, 48, 49, 51, 57, 74, 75, 93, 94, 95, 105, 121, 122, and 135.
- Joel Lehman and Kenneth O. Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2011b. — Cited on pages 2, 28, 29, 30, 33, 34, 42, 43, 61, 94, 95, 96, 100, and 121.
- Joel Lehman and Kenneth O. Stanley. Improving evolvability through novelty search and self-adaptation. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 2693–2700. IEEE, 2011c. — Cited on page 27.

- Joel Lehman and Kenneth O. Stanley. Novelty search and the problem with objectives. *Genetic Programming Theory and Practice IX*, pages 37–56, 2011d. — Cited on pages 57 and 95.
- Joel Lehman and Kenneth O. Stanley. Beyond open-endedness: Quantifying impressiveness. In *Proceedings of the International Conference on Artificial Life*, 2012. — Cited on pages 25 and 26.
- Joel Lehman, Kenneth O. Stanley, and Risto Miikkulainen. Effective diversity maintenance in deceptive domains. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2013. — Cited on pages 2, 24, and 122.
- Hao Li, Maoguo Gong, Deyu Meng, and Qiguang Miao. Multi-objective self-paced learning. In *Proceedings of the Thirtieth AAAI Conf. on Artificial Intelligence*, pages 1802–1808, 2016. — Cited on page 28.
- Kan Li and José C. Príncipe. Surprise-novelty information processing for gaussian online active learning (snip-goal). In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2018. — Cited on page 121.
- Ke Li, Kalyanmoy Deb, Qingfu Zhang, and Qiang Zhang. Efficient nondomination level update method for steady-state evolutionary multiobjective optimization. *IEEE Transactions on Cybernetics*, 47(9):2838–2849, 2017. — Cited on pages 42, 74, 94, 95, 100, 105, and 122.
- Antonios Liapis. *Searching for Sentient Design Tools for Game Development*. PhD thesis, Center for Computer Games, IT University of Copenhagen, Copenhagen, Denmark, 2014. — Cited on pages vi, 26, and 122.
- Antonios Liapis. Exploring the visual styles of arcade game assets. In *Proceedings of Evolutionary and Biologically Inspired Music, Sound, Art and Design (EvoMusArt)*. Springer, 2016. — Cited on page 125.
- Antonios Liapis, Héctor P. Martínez, Julian Togelius, and Georgios N. Yannakakis. Transforming exploratory creativity with DeLeNoX. In *Proceedings of the International Conference on Computational Creativity*, 2013. — Cited on pages 2, 13, 16, 22, 25, and 26.
- Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. Constrained novelty search: A study on game content generation. *Evolutionary Computation*, 23(1), 2015. — Cited on pages 16, 26, 27, 28, 125, 126, and 127.
- G. E. Liepins and M. D. Vose. Representational issues in genetic optimization. *Journal of Experimental and Theoretical Artificial Intelligence*, 2(101), 1990. — Cited on page 23.
- Emiliano Lorini and Cristiano Castelfranchi. The cognitive structure of surprise: looking for basic principles. *Topoi*, 26(1), 2007. — Cited on pages 2 and 12.
- Luís Macedo and Amílcar Cardoso. Modeling forms of surprise in an artificial agent. In *Proceedings of the annual Conference of the Cognitive Science Society*, 2001. — Cited on pages 3 and 12.
- Luís Macedo and Amílcar Cardoso. Assessing creativity: the importance of unexpected novelty. *Structure*, 1(C2):C3, 2002. — Cited on pages 3 and 12.

- Luis Macedo, Amílcar Cardoso, Rainer Reisenzein, Emiliano Lorini, and C. Castelfranchi. Artificial surprise. *Handbook of research on synthetic emotions and sociable robotics: New applications in affective computing and artificial intelligence*, 2009. — Cited on pages 3 and 12.
- Mary Lou 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. — Cited on pages 2, 12, and 14.
- Mary Lou Maher and Douglas H. Fisher. Using AI to evaluate creative designs. In *Proceedings of the 2nd International Conference on Design Creativity*, volume 1, 2012. — Cited on pages v, 3, and 12.
- Mary Lou Maher, Katherine Brady, and Douglas H. Fisher. Computational models of surprise in evaluating creative design. In *Proceedings of the fourth international conference on computational creativity*, volume 147. Citeseer, 2013. — Cited on pages 3 and 12.
- Samir W. Mahfoud. Niching methods for genetic algorithms. *Urbana*, 51(95001):62–94, 1995. — Cited on pages 24 and 28.
- Markos Markou and Sameer Singh. Novelty detection: a review—part 1: statistical approaches. *Signal processing*, 83(12):2481–2497, 2003. — Cited on page 13.
- Yuliana Martínez, Enrique Naredo, Leonardo Trujillo, and Edgar Galván-López. Searching for novel regression functions. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 16–23. IEEE, 2013. — Cited on page 26.
- Georgios Methenitis, Daniel Hennes, Dario Izzo, and Arnoud Visser. Novelty search for soft robotic space exploration. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2015. — Cited on pages 61, 62, 63, 64, 86, 120, and 136.
- Wulf-Uwe Meyer, Rainer Reisenzein, and Achim Schützwohl. Toward a process analysis of emotions: The case of surprise. *Motivation and Emotion*, 21(3), 1997. — Cited on page 12.
- Zbigniew Michalewicz and Stephen J. Hartley. Genetic algorithms+ data structures= evolution programs. *Mathematical Intelligencer*, 18(3):71, 1996. — Cited on page 20.
- Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013. — Cited on page 1.
- Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998. — Cited on page 16.
- Melanie Mitchell, Stephanie Forrest, and John H. Holland. The royal road for genetic algorithms: Fitness landscapes and ga performance. In *Proceedings of the first european conference on artificial life*, pages 245–254, 1992. — Cited on page 23.
- Tom M. Mitchell et al. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45(37): 870–877, 1997. — Cited on pages 1 and 20.

- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015. — Cited on pages 30 and 130.
- Jean-Baptiste Mouret. Novelty-based multiobjectivization. *New horizons in evolutionary robotics*, pages 139–154, 2011. — Cited on pages 27, 29, and 47.
- Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015. — Cited on pages x, 2, 28, 66, 87, 123, 124, and 127.
- Jean-Baptiste Mouret and Stéphane Doncieux. Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evolutionary computation*, 20(1):91–133, 2012. — Cited on page 120.
- Enrique Naredo and Leonardo Trujillo. Searching for novel clustering programs. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2013. — Cited on page 26.
- B. Naudts and A. Verschoren. Epistasis and deceptivity. *Bulletin of the Belgian Mathematical Society*, 6(1), 1999. — Cited on page 23.
- Antonio J. Nebro and Juan J. Durillo. On the effect of applying a steady-state selection scheme in the multi-objective genetic algorithm nsga-ii. In *Nature-Inspired Algorithms for Optimisation*, pages 435–456. Springer, 2009. — Cited on page 122.
- Andrew Ortony and Derek Partridge. Surprisingness and expectation failure: what’s the difference? In *Proceedings of the Joint conference on Artificial intelligence*, 1987. — Cited on page 12.
- Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, volume 2017, 2017. — Cited on page 31.
- Gregory S. Paul. *Predatory Dinosaurs of the World: A Complete Illustrated Guide*. Simon & Schuster, 1988. — Cited on page 4.
- Diego Perez-Liebana, Jialin Liu, Ahmed Khalifa, Raluca D. Gaina, Julian Togelius, and Simon M. Lucas. General video game ai: a multi-track framework for evaluating agents, games and content generation algorithms. *IEEE Transactions on Games*, 2019. — Cited on page 131.
- Mike Preuss. *Multimodal optimization by means of evolutionary algorithms*. Springer, 2015. — Cited on pages 24, 28, and 125.
- Mike Preuss, Antonios Liapis, and Julian Togelius. Searching for good and diverse game levels. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2014. — Cited on pages 28 and 126.
- Justin K. Pugh, Lisa B. Soros, Paul A. Szerlip, and Kenneth O. Stanley. Confronting the challenge of quality diversity. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2015. — Cited on pages 27 and 30.

- Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016. — Cited on pages 2, 16, 27, 28, 30, 42, 47, 95, and 123.
- Robin C. Purshouse and Peter J. Fleming. On the evolutionary optimization of many conflicting objectives. *IEEE Transactions on Evolutionary Computation*, 11, 2007. — Cited on pages 24, 100, and 122.
- Chao Qian, Yang Yu, and Zhi-Hua Zhou. Pareto ensemble pruning. In *Proceedings of the AAAI Conf. on Artificial Intelligence*, pages 2935–2941, 2015a. — Cited on page 28.
- Chao Qian, Yang Yu, and Zhi-Hua Zhou. Subset selection by pareto optimization. In *Advances in Neural Information Processing Systems*, pages 1774–1782, 2015b. — Cited on page 28.
- Chao Qian, Jing-Cheng Shi, Ke Tang, and Zhi-Hua Zhou. Constrained monotone k-submodular function maximization using multi-objective evolutionary algorithms with theoretical guarantee. *IEEE Transactions on Evolutionary Computation*, 2017. — Cited on page 28.
- Ingo Rechenberg. Evolution strategy: Optimization of technical systems by means of biological evolution. *Fromman-Holzboog, Stuttgart*, 104:15–16, 1973. — Cited on page 16.
- Rainer Reisenzein. The subjective experience of surprise. *The message within: The role of subjective experience in social cognition and behavior*, pages 262–279, 2000. — Cited on page 38.
- Joseph Reisinger, Erkin Bahceci, Igor Karpov, and Risto Miikkulainen. Coevolving strategies for general game playing. In *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, pages 320–327. IEEE, 2007. — Cited on page 22.
- AM Reynolds. Maze-solving by chemotaxis. *Physical Review E*, 81(6):062901, 2010. — Cited on pages 58 and 96.
- Henry Gordon Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74(2):358–366, 1953. — Cited on page 23.
- Sebastian Risi and Kenneth O. Stanley. Confronting the challenge of learning a flexible neural controller for a diversity of morphologies. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2013. — Cited on pages 22 and 26.
- Sebastian Risi, Sandy D. Vanderbleek, Charles E. Hughes, and Kenneth O. Stanley. How novelty search escapes the deceptive trap of learning to learn. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2009. — Cited on page 25.
- Sebastian Risi, Charles E. Hughes, and Kenneth O. Stanley. Evolving plastic neural networks with novelty search. *Adaptive Behavior*, 18(6):470–491, 2010. — Cited on page 25.
- Graeme Ritchie. Some empirical criteria for attributing creativity to a computer program. *Minds and Machines*, 17(1), 2007. — Cited on pages 2, 11, 15, and 27.

- Duda R.O. and Hart P.E. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York, 1973. — Cited on page 13.
- Lior Rokach and Oded Maimon. Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer, 2005. — Cited on page 120.
- Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. *arXiv preprint arXiv:1810.02274*, 2018. — Cited on page 31.
- Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3), 2010. — Cited on page 31.
- Marc Schoenauer. Shape representations and evolution schemes. *Evolutionary Programming*, 5, 1996. — Cited on page 20.
- Marc Schoenauer and Zbigniew Michalewicz. Evolutionary computation at the edge of feasibility. In *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, pages 245–254, 1996. — Cited on page 114.
- John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015. — Cited on page 30.
- Jimmy Secretan, Nicholas Beato, David B. D. Ambrosio, Adelein Rodriguez, Adam Campbell, and Kenneth O. Stanley. Picbreeder: evolving pictures collaboratively online. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1759–1768. ACM, 2008. — Cited on page 22.
- Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001. — Cited on page 128.
- Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. *arXiv preprint arXiv:1810.12162*, 2018. — Cited on page 31.
- Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM, 1994. — Cited on pages 34 and 35.
- Adam M. Smith and Michael Mateas. Answer set programming for procedural content generation: A design space approach. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):187–200, 2011. — Cited on page 126.
- Adam M. Smith, Eric Butler, and Zoran Popović. Quantifying over play: Constraining undesirable solutions in puzzle design. In *Proceedings of ACM Conference on Foundations of Digital Games*, pages 221–228, 2013. — Cited on page 126.
- Davy Smith, Laurissa Tokarchuk, and Geraint Wiggins. Rapid phenotypic landscape exploration through hierarchical spatial partitioning. In *International conference on parallel problem solving from nature*, pages 911–920. Springer, 2016. — Cited on page 1.
- Gillian Smith, Jim Whitehead, and Michael Mateas. Tanagra: Reactive planning and constraint solving for mixed-initiative level design. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):201–215, 2011. — Cited on page 126.

- William M. Spears. Crossover or mutation? In *Foundations of genetic algorithms*, volume 2, pages 221–237. Elsevier, 1993. — Cited on page 17.
- Kenneth O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162, 2007. — Cited on pages 22, 35, 61, and 137.
- Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2), 2002. — Cited on pages vi, 17, 21, 22, 24, 28, 35, 38, 48, 49, 61, 74, 125, and 135.
- Kenneth O. Stanley and Risto Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of artificial intelligence research*, 21:63–100, 2004. — Cited on page 22.
- Kenneth O. Stanley, Bobby D. Bryant, and Risto Miikkulainen. Real-time neuroevolution in the nero video game. *IEEE Transactions on Evolutionary Computation*, 9(6):653–668, 2005a. — Cited on page 22.
- Kenneth O. Stanley, Nate Kohl, Rini Sherony, and Risto Miikkulainen. Neuroevolution of an automobile crash warning system. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2005b. — Cited on page 22.
- Christopher Stanton and Jeff Clune. Curiosity search: producing generalists by encouraging individuals to continually explore and acquire skills throughout their lifetime. *PloS one*, 11(9):e0162235, 2016. — Cited on page 1.
- Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. Procedural content generation via machine learning (pcgml). *IEEE Transactions on Games*, 10(3):257–270, 2018. — Cited on pages 125 and 126.
- Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018. — Cited on pages 30 and 130.
- Haoran Tang, Rein Houthoofd, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. #exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2753–2762, 2017. — Cited on page 31.
- Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):172–186, 2011. — Cited on pages 125 and 126.
- Julian Togelius, Mike Preuss, Nicola Beume, Simon Wessing, Johan Hagelbäck, Georgios N Yannakakis, and Corrado Grappiolo. Controllable procedural map generation via multi-objective evolution. *Genetic Programming and Evolvable Machines*, 14(2):245–277, 2013. — Cited on page 126.
- Deepak Trivedi, Christopher D. Rahn, William M. Kier, and Ian D. Walker. Soft robotics: Biological inspiration, state of the art, and future research. *Applied Bionics and Biomechanics*, 5(3):99–117, 2008. — Cited on page 34.

- Adriano Vinhas, Filipe Assunção, João Correia, Aniko Ekárt, and Penousal Machado. Fitness and novelty in evolutionary art. In *International Conference on Evolutionary and Biologically Inspired Music and Art*, pages 225–240. Springer, 2016. — Cited on page 16.
- 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. — Cited on page 24.
- Darrell Whitley, Timothy Starkweather, and Christopher Bogart. Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel computing*, 14(3): 347–361, 1990. — Cited on page 20.
- L. Darrell Whitley. Fundamental principles of deception in genetic search. In *Foundations of genetic algorithms*, volume 1, pages 221–241. Elsevier, 1991. — Cited on pages 1, 15, and 23.
- Geraint A. Wiggins. A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems*, 19(7), 2006. — Cited on pages 2, 11, and 12.
- Edward Orlando Wiley and Bruce S. Lieberman. *Phylogenetics: theory and practice of phylogenetic systematics*. John Wiley & Sons, 2011. — Cited on page 4.
- Brian G. Woolley and Kenneth O. Stanley. A novel human-computer collaboration: combining novelty search with interactive evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2014. — Cited on page 121.
- Larry Yaeger. Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or poly world: Life in a new context. *Santa Fe Institute Studies In The Sciences Of Complexity*, 17, 1994. — Cited on page 1.
- Georgios N. Yannakakis and Antonios Liapis. Searching for surprise. In *Proceedings of the International Conference on Computational Creativity*, 2016. — Cited on pages v, 3, 4, 12, 25, and 41.
- Georgios N. Yannakakis and Julian Togelius. *Artificial intelligence and games*, volume 2. Springer, 2018. — Cited on page 125.
- Georgios N. Yannakakis, John Levine, John Hallam, and Markos Papageorgiou. Performance, robustness and effort cost comparison of machine learning mechanisms in flatland. In *Proceedings of the Mediterranean Conference on Control and Automation*, 2003. — Cited on page 52.
- Georgios N. Yannakakis, Roddy Cowie, and Carlos Busso. The ordinal nature of emotions: An emerging approach. *IEEE Transactions on Affective Computing*, 2018. — Cited on page 120.