Exploiting Exploration: A Survey of Quality Diversity Search

Andrew Pozzuoli

Department of Computer Science

Faculty of Mathematics and Science,

Brock University St. Catharines, Ontario

2021

Contents

List of Figures

List of Algorithms

Abstract

Quality diversity search algorithms are designed to illuminate the search landscape showing a collection of high-performing solutions each defined by unique behaviour features. While promoting diversity helps keep the population from falling into pits of local optima and premature convergence, the benefits of quality diversity search have proven to reach farther than this. This review covers quality diversity search and the steps that led to it from the pure diversity of novelty search to the objective-divergent combination seen today in MAP-Elites. The review also covers its applications demonstrating the benefits it brings to creative inspiration and adaptable design.

Chapter 1

Introduction

Objective-based optimization aims to search a landscape of solutions for a peak of quality. The ideal image of a computer process that automatically scans and finds a perfect solution unfettered by human bias or input comes to mind. The reality, however, can be far more disappointing. With the power of objective-based search algorithms to solve complex problems, there come a set of new problems limiting their effects. One such problem that particularly can affect the evolutionary branch of these algorithms is that of premature convergence, where the fitness of solutions in a population make no further improvements but the solution they have settled on is sub-optimal. Many different approaches have been used to handle this with varying success.

A more recent approach to handling this issue came from Lehman and Stanley's *novelty search* [2] which abandoned using an objective to determine solution fitness and instead rewarded individuals who were the most different from others in the population. This proved to be highly effective at solving deceptive tasks such as maze navigation which can easily place an objective-based search into an attractive local optimum diverting attention away from actually solving the task viably.

The new concept of measuring novelty and, more generally, diversity of a population caused a stir and an array of new search algorithms cropped up in the following years.

Among these were Gravina et al.'s *surprise search* [4] and Mengitsu et al.'s *evolvability search* [5].

By abandoning objectives entirely, there was no method for exploiting good solutions found by novelty search. Researchers eventually took to combining diversity and objectives. The most successful of these methods being Mouret and Clune's MAP-Elites algorithm [6] which spawned an entirely new genre of search algorithm now known as *quality-diversity algorithms*. MAP-Elites first defines a behaviour grid made up of dimensions where each dimension is some feature defined by the user. As the search progresses, each cell of the grid is filled with the highest performing solution that fits the features mapped to it. For example, in the task of designing a robot that walks as far as possible, one feature may be leg length and another could be the number of joints. The best performing robot with legs that are 10 cm and has one joint may occupy one cell while another cell contains the highest performing robot with 5 cm legs and two joints. The result is a collection of high-performing solutions (*elites*), each behaviourally unique from one another. This has applications in engineering areas where robots may need a repertoire of movements so that in the event of damage, they can switch the the highest performing movement pattern that does not require the damaged part. Another application is in creative domains where creators of video game levels [3] or characters [7] can see a map of diverse creations that can inspire their designs.

This survey provides the stepping stones that led to quality-diversity search (chapter 2), the problem of premature convergence (chapter 3), early diversity-promoting techniques (section 3.2), pure diversity searches (section 4.1), and quality-diversity searches (chapter 4). The survey will also cover key concepts to consider with diversity search (chapter 5) and various applications (chapter 6).

Chapter 2

Background

The following chapter goes over fundamental background concepts related to the ideas explored in the subsequent chapters. This includes evolutionary computation, genetic algorithms, genetic programming, and NEAT.

2.1 Evolutionary Computation

The role of nature as an endless source of inspiration is not limited to the artistic endeavours of painters and poets. Even computer science, a field whose reputation is steeped in mathematical logic and hand-coded engineering, and whose association with nature seems more distant than almost any other, has looked to the natural world for guidance.

Nature's evolution via natural selection continues to produce seemingly unbounded creativity and ingenuity, inspiring researchers to adapt it for optimization purposes. Using the model of survival of the fittest and natural selection, pioneering researchers Fogel et al. (1966), Holland (1975), and Rechenberg (1973), independently used the evolutionary methods towards problem solving [1]. This motivated a population-based approach where many randomly generated solutions are guided over generations of genetic crossover and mutation towards better solutions with the eventual goal of finding a global optimal solution. The process is stochastic and finding the global optimum or even a good solution is not guaranteed nor can be predicted, however the strategies have proved effective for a wide array of problems over the years of research dedicated to evolutionary computation [8].

Rechenberg, for example, used the strategy to evolve reduction pieces for pipes. The end result being oddly shaped pieces that were more optimized for flow than any human-engineered piece [1]. This demonstrates one of evolutionary computation's advantages—the ability to distance solutions from human bias and optimize solutions in ways that a human would not be able to think. The method is not free from human bias and requires a number of things to be hand-tuned as the following subsections will show.

2.1.1 Genetic Algorithms

A genetic algorithm (GA) is a population-based, stochastic search method inspired by the principles of natural selection [1, 9].

Defining the problem, solution representation, and fitness

Before running the algorithm, it is important to first consider the problem. What is being optimized? What are the constraints? Consider the trivial problem of enclosing a plot of land with a fence where the goal is to achieve an area of at least 400 m^2 while minimizing the cost of fencing used. In this case, the objective and constraints are clear: minimal cost for maximal area. Having defined this, it next necessary to determine a representation (or chromosome) for candidate solutions. The width and length of the fencing would be a good way to represent solutions for the trivial example. Selecting a good representation is important. It is important to think about how the genetic operators of mutation and crossover will affect the representations. A poor representation may result in these operations destroying potential solutions. In order to preserve a chromosomal representation for this trivial example, a bit string representing each dimension could work. At this stage, it is possible to incorporate constraints in the dimensions that keep the area above the minimum set in the problem although this could be checked at the fitness stage instead. After finding a sufficient representation, there must be some way to evaluate its fitness. This determines how good a given solution is and will be used to compare solutions against each other. In this trivial example a possible fitness function could be $f(l, w) = c(2l + 2w)$ where c is the cost per meter of fence. In other words, the fitness is the cost of the surrounding fence with the goal of minimizing f. To constrain the dimensions, it is possible to assign a fitness that prevents any solution with an area lower than 400 m^2 from being better than a solution within the constraint. Since this example aims to minimize the fitness (lower cost fence results in a better solution for the problem), assigning a fitness of the maximum possible value to anything less than 400 m^2 will achieve this. Having defined the problem, solution representation, and fitness function, the core of the genetic algorithm comes into play.

Population size

The algorithm begins by initializing a population of candidate solutions with randomly generated chromosomes. In our trivial example, these are randomly generated bit strings representing the dimensions of the fence. The population size is determined by the programmer ahead of time. Too small a population and there is little genetic diversity resulting in a small portion of the search space being explored. Too large a population and the computing time needed to evaluate all the individuals slows the evolution down. Once this population has been created, each candidate solution is evaluated for its fitness. Naturally, the initial, random set would have low fitnesses with some solutions possibly being infeasible. These fitnesses are used to compare solutions to one another ultimately for selection.

Fitness-Proportional Selection

Fitness-proportional selection aims to preserve the best genetic material while weeding out the worse performing ones. There are a number of ways to determine selection such a using a tournament, roulette, ranking, etc. Each has its strengths. Suppose for the trivial example, a tournament selection is used with paired match-ups. In this case, members of the population are paired up and the one with the better fitness is selected to move on to the next round. Two solutions could be paired up, both having an area of 400 m^2, but the one with the lower cost of fencing will move on. Selection continues until the population of the next generation reaches the user-decided population size. It is important to notice here that weaker solutions are less likely to make it to the mating pool since survival in the tournament selection depends on dominating competing solutions in fitness. Ultimately, a successful selection will drive bad genes out of the population and preserve good genes for future generations to use.

Crossover and mutation

The two main reproduction operators are crossover and mutation. The selected solutions will participate in crossover where they create child solutions by recombining the genes of the parent solutions. Once again, there are many ways of doing this and it is important to select a crossover method that works best for the representation. One crossover type is known as *single-point crossover* where a single point in the chromosome is selected and the child inherits everything left of that point from one parent and everything right of that point from the other. Not all individuals selected have to have their genes recombined. Crossover rate is a parameter that can be set that determines how likely individuals participate in crossover. A crossover rate of 90% ensures that 10% of the population is copied from the last generation with no recombination. Crossover does not guarantee better solutions; some children may even become worse than the parents. However, selection will ensure that weaker solutions are removed and better solutions move on. Over many generations of this, the population will ideally move towards areas of the search space more likely containing a global optimum.

The next step for the child solutions is mutation where genes are randomly modified with a small chance. This helps to improve the diversity and possibly allows for solutions to explore more than parent genes will allow. In a bit string this could simply be flipping a bit.

Elitism

It is also possible to guarantee that a certain amount of the top individuals move on via *elitism.* With elitism the best individual or a percentage of the top individuals are selected to move on with no tournament competition, crossover, or mutation. This preserves the best genes from each generation but should be used carefully to avoid greediness which could result in sub-optimal convergence (section 3.1).

Overall these operators of selection, crossover, and mutation work together to remove bad solutions and prioritize good solutions guiding the population towards better and better solutions. Once these operators are complete and the new population has been generated, the process repeats until a pre-determined number of generations have passed or until the population has converged and makes no further improvement. A flowchart demonstrating the basic GA algorithm is shown in figure 2.1 from Deb's *An introduction to genetic algorithms* (1999) [1].

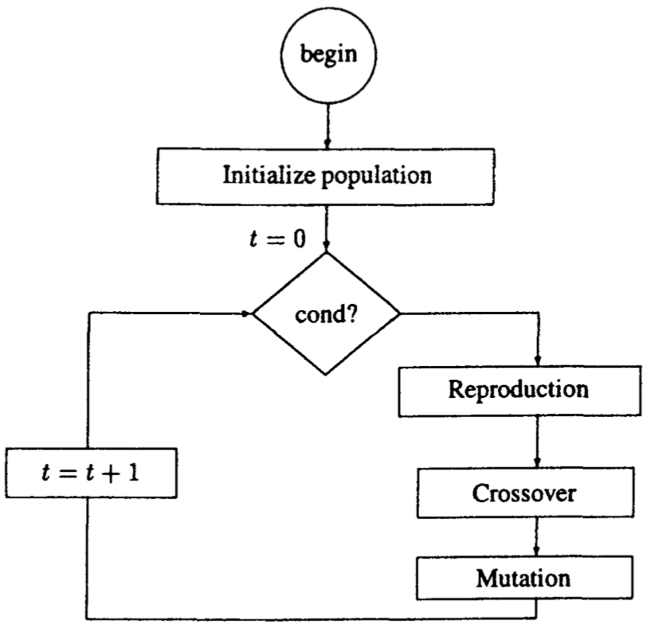

Figure 2.1: Flowchart of the basic Genetic Algorithm from Deb's *An introduction to genetic algorithms* (1999) [1]

2.1.2 Genetic Programming

The same guiding principles of evolutionary computation can also be extended to create entire computer programs. Applying these techniques to create computer programs leads to the field of genetic programming (GP) [10, 11, 12].

Like GAs, GP also begins with an initial population of randomly generated computer programs. Each of these programs are evaluated in terms of their ability to achieve some goal. Their fitnesses are compared and fitter programs have a higher chance to pass their genetic material to the next generation. The key distinguishing feature of GP is in individual representation. Candidates are represented as trees with leaves of terminal nodes connected to function nodes. Consider for example a regression problem where the goal is to find a mathematical function to fit as closely to a set of data points as possible. The terminal leaf nodes can be the input data for example and the function nodes can be the various mathematical operators such as addition, subtraction, multiplication, and division. Constructing a tree using only these nodes would create various mathematical functions each can be evaluated by how far the function output is from the plotted data. An example of a simple GP tree is shown in figure 2.2.

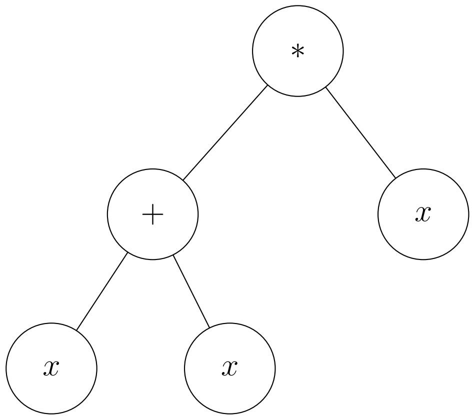

Figure 2.2: Example GP tree representing the mathematical expression $(x + x) * x$

Closure

GP trees require a property known as *closure* [10] which ensures that terminal types are consistent with the function nodes. The mathematical operator '+' takes in two integer types as arguments and returns one integer as a result. Since crossover can cause any subtree to attach to any node, the terminal types must remain consistent with the functions, or the function must have some way of converting improper types into proper types [12]. Another aspect of closure is to ensure that functions can be evaluated without causing errors. Consider a division function, if the terminals of a tree cause a division by zero then there must be some way for the function to handle the case without throwing an error [12].

Sufficiency

Another important property of a GP language is *sufficiency* which means that the functions and terminals defined are sufficient for solving the problem at hand [12]. The language set $\{+, -, 0, 1\}$ would be insufficient for solving a problem that requires e^x since the set has no functions to express it beyond a crude approximation. Sufficiency can be difficult to guarantee unless expert knowledge or theory can determine the necessary language set.

Tree Initialization

Unlike with genetic algorithms, GP trees do not have a fixed size which can pose problems for initialization. Two methods known as *full* and *grow* have a user-specified depth that initial trees do not exceed [12]. The *full* method builds trees by adding function nodes until the specified depth is reached, then terminals must be used. The *grow* method allows for both terminals and function nodes to be selected until the maximum depth is reached.

To allow for more variation in trees, the *ramped half-and-half* method [10] has half the population initialized with *grow* and half with *full*. The depth of the initialized trees increases over a range of values.

Reproduction

A common crossover operation for tree reproduction is *subtree crossover* [12] where a node is selected as a crossover point in parent trees and the offspring is created by taking a copy of the first parent and replacing the subtree rooted at the crossover point with the subtree of the second parent rooted at its crossover point. Selecting a crossover point is typically chosen randomly but in order to move selection away from choosing mostly small subtrees or terminals, Koza suggested selecting functions 90% of the time and terminals 10% of the time [10].

A typical mutation operator is *subtree mutation* where a node is selected randomly and the subtree is replaced with a randomly generated one [12].

Evaluation

GP trees are evaluated in a depth-first approach where arguments in the deeper leaves are known before the function node higher up evaluates them. The function node then returns the value up the tree for the next function node until the root knows the values of all its arguments to evaluate and return as output [12].

Fitness is measured by comparing the result of the evaluated GP tree to a known goal (e.g. the end result of a maze navigation in relation to the exit), or in comparison to other GP trees (e.g. the amount of distance a gaited robot walks) [12].

2.1.3 NeuroEvolution of Augmenting Topologies

NeuroEvolution of Augmenting Topologies is a method for evolving neural networks introduced by Stanley and Miikulainen in their 2002 paper *Evolving Neural Networks through Augmenting Topologies* [13]. The networks act as controlling mechanisms for agents based on their sensory inputs. This method begins with a population of a single, simple network topology for all individuals and gradually increases the complexity over generations resulting in speciation into complex behaviours. A tag is used to keep track of components and how long they have been in the population. Components with similar tags are aligned during crossover which provides a simpler way to compare topologies than the NP-hard method of graph comparison. Tag alignment also protects new components during crossover since only similar structures crossover with each other.

Hypercube-based NEAT (HyperNEAT) is an extension of NEAT which takes a generative network that evolves using the NEAT method to indirectly map evolved genotypes to another neural network [14].

Chapter 3

Diversity and The Problem of Premature Convergence

3.1 Premature Convergence

Evolutionary computation can run into the problem of premature convergence where candidate solutions of the population get stuck in a local optimum and no further improvements are made over more generations. Often these local optima are not the global optimum. To tackle this issue, one area to look at is diversity of candidates. More diversity leads to more of the search space being explored, and this larger search area is more likely to include the global optimum.

Lehman and Stanley argued that this problem of premature convergence could be linked to the fitness function itself especially for deceptive search spaces with many local optima [2]. In this case, the objective function can inhibit the development of candidate solutions since the stepping stones leading to the goal are not rewarded. Modifying the fitness to also reward these stepping stones can work against evolutionary computation since this could add human biases to the path towards the goal. Therefore it is necessary to apply selective pressure towards finding these stepping stones that may not benefit the end goal fitness but rather will lead to it.

One example illustrating the stepping stones issue was displayed in Gomes et al's exploration of swarm robotics behaviours [15]. The task was in the domain of resource sharing where the swarm of robots share one charging station and must coordinate to allow every member of the swarm periodic access to the charging station. Failure to reach a charging station in time will cause the robot's energy to deplete. With a fitness based on the robot's lifespan, no viable solution was found. One of the most successful swarm behaviours observed had all of the robots remaining still until their energy depleted. The reason why this became the most successful under this fitness is that movement caused energy to deplete quicker. Any robot early in the run who initially stood still would have a higher fitness than any robot who moved and did not find a charging station so stillness was selected as fitter. In this case, the stepping stones leading robots to move toward the charging station resulted in a lower fitness and thus are not rewarded even though they are necessary for a better solution than what the run found.

3.2 Before Diversity Search

Pugh, Soros and Stanley [16] identified two search algorithms related to, but pre-dating, diversity search. The earlier one, multi-model function optimization [17], aims to find multiple optima in a search space leading to a diversity of solutions. The diversity here is genotypic (individual representation) rather than behavioural, which can be limiting (see chapter 5). The other search is multi-objective optimization, where the algorithm finds an optimal set of solutions with trade-offs between multiple objectives (the Pareto front). Pugh, Soros and Stanley highlight that this search drives solutions towards specific goals, while diversity search drives solutions away from previously searched regions.

An extensive study comparing ways of increasing diversity in evolutionary robotics was conducted by Doncieux and Mouret in a 2012 survey [18]. The survey compared diversity encouraging methods on three tasks in the field of evolutionary robotics. The methods being fitness sharing (a diversity method where individuals who are more different but have a lower fitness obtain a fitness similar to those who are better fit but less unique), multi-objective diversity, and multi-objective fitness sharing. The tasks tested on were a deceptive maze, a light seeking task, and a ball collecting task. It was found that explicitly encouraging behavioural diversity led to significant improvements in the evolution of robots for these tasks. They found that tasks like the deceptive maze which were difficult to solve by classical means became easy to solve by encouraging behavioural diversity. They also found that using behavioural distance as a diversity measure outperformed genotype distance.

An important finding of their survey [18] was that diversity is a far more important factor than any other parameter in terms of experiment success. The encoding in terms of genotype had no effect on the rankings nor did parameter tuning. These and similar findings led certain researchers to question the established norms of objective-based searches and see how far diversity could go.

Chapter 4

Diversity Search

4.1 Pure Diversity Search

Pure diversity search is characterized by the counter intuitive notion of discarding objectives and optimizing strictly for diversity, which can indirectly find solutions to problems better than an objective-based search for some problems.

4.1.1 Novelty Search

In their 2008 paper *Exploiting Open-Endedness to Solve Problems Through the Search for Novelty* [2] Lehman and Stanley proposed that, instead of searching for the objective of a solution, it is possible to ignore the fitness entirely and only select candidates whose functionality is significantly different from other individuals. This new search was called *novelty search*. A novelty measure was introduced allowing comparison of how different solutions are from one another, rewarding candidates who stand out from their peers. Having a sufficiently high novelty score allows candidates to get added to an archive so the algorithm can not only compare to the current generation but also to previous candidates. It is easy to see how this approach promotes diversity, since it is able to directly select for it with no other objective.

Pseudocode of the basic novelty search algorithm (algorithm 1) is similar to the standard genetic algorithm but with fitness evaluation replaced with novelty evaluation. A population is initialized with random solutions. Each member of the population is then evaluated by some measure of novelty. Novelty can be measured in many ways (see chapter 5). For the example algorithm, the average distance between the individual and k-nearest neighbours is used. A higher average distance indicates that this individual is in a sparser region of the search space and, consequently, is more novel. The k-nearest neighbours not only compares to members within the current population, but also to those in the archive of past novel behaviours. If the sparseness of the individual is evaluated to be higher than a minimum threshold value, the individual enters the archive. The genetic algorithm continues normally with selection, crossover, and mutation, but with fitness replaced with the novelty score.

Algorithm 1: Novelty Search

```
1  Initialize population of randomly generated individuals
2  Initialize empty archive
3  generation ← 0
4  Initialize maxGenerations
5  Initialize k value for k-nearest neighbours calculation
6  Initialize minSparseness threshold for archive entry
7  while generation < maxGenerations OR stopping criteria not met do
8      foreach individual i in population do
9          evaluations ← 0
10         distanceTotal ← 0
11         while evaluations < k do
12             $\mu_{evaluations}$ ← next nearest neighbour of i in the population or the archive
13             $distanceTotal = distanceTotal + distance(i, \mu_{evaluations})$
14         end
15         $sparseness = distanceTotal/k$
16         if sparseness <= minSparseness then
17             Add i to archive
18         end
19     end
20     while population of next generation is not filled do
21         Apply selection with bias towards individuals with higher sparseness
22         Apply genetic operators crossover and mutation to selected individuals
23     end
24 end
```

Without objectives, it may appear that this new search is simply a dressed up random search. What separates this approach from an exhaustive search is that there is no back-tracking over previous solutions thanks to the archive. Lehman and Stanley also argued that the number of novel behaviours in many task domains is limited enough to reasonably use novelty search without any further constraints offered by a fitness function.

Novelty search can also be used with more than one behaviour measure. Pugh *et al.* highlighted this variant, known as multi-behaviour novelty search (NS-NS) in their quality diversity review [16]. NS-NS uses multiple archives and every individual in the population is evaluated with a novelty score for each behaviour. The breeding potential is determined by a Pareto ranking across all behaviour novelty scores.

Archive-less novelty search has also been studied. A 2021 study by Salehi *et al.* [19] proposed *behaviour recognition novelty search (BRNS),* which uses an encoder to recognize if a behaviour has been seen before in the search. This approach was found to improve runtime with a significant increase in efficiency in higher dimensional spaces.

One major takeaway from novelty search is that the idea is general and can be applied to any evolutionary computation task domain where diversity is measurable. However, finding a sufficient diversity measure is not always easy.

4.1.2 Surprise Search

Inspired by novelty search and findings in the literature of computational creativity, Gravina *et al.*, proposed another divergent search called *surprise search* [4]. Similar to novelty search, surprise search takes into account past behaviours and does not account for any other objective function. Where it differs is that. while novelty search rewards individuals for novel behaviour based on past actions, surprise search uses past actions to model predicted behaviours and rewards individuals with unexpected functionality when compared to the predicted models.

4.1.3 Evolvability Search

A different approach to promote diversity is to increase evolvability of solutions. Evolvability, as defined by Mengitsu *et al.* in their paper on evolvability search [5], is measured in terms of an individual's ability to produce future behavioural variation. The algorithm generates simulated offspring of individuals and measures how unique the behaviour of the offspring are from each other. These simulated offspring are discarded and if the measure of offspring diversity is high enough, the parent has a higher chance of selection for reproduction. As with novelty search and surprise search, objectives are ignored. Mengitsu et al., found that increased selection pressure towards individual offspring diversity raises the overall population diversity.

4.1.4 Extinction Events

Lehman and Miikulainen explored augmenting evolutionary algorithms with extinction events [20]. The idea being that random extinction events would indirectly select for diversification since lineages that are able to survive multiple extinction events would also be able to quickly occupy many different niches. In addition to the extinction events, the algorithm was implemented with diversity search. They found that extinction events increased evolvability but only when combined with diversity search. Objective-based search with extinction events did not perform as well in terms of selecting evolvable lineages. When compared to evolvability search, extinction events are able to achieve similar goals—that of selecting for evolvability—without the cost associated with running evaluations on every individual's simulated offspring.

One of the discussion points brought up by Lehman and Miikulainnen was that their paper's results suggested that the best practices for objective search algorithms may not apply to diversity search algorithms. A new paradigm may need to be considered to best foster the growth of the field.

4.1.5 Deceptive Maze Navigation and the Limits of Pure Diversity

Lehman and Stanley's novelty search [2], Gravina et al.'s surprise search [4], and Mengistu's evolvability search [5] found considerable success in the experiments conducted for their respective papers. The task domain was maze navigation where robots must navigate a maze from a starting point to the exit. Along the way to the exit, the walls that make up the maze can lead to dead ends which may be near the goal. This landscape provides a fitting abstraction of a deceptive landscape where it is easy for candidate solutions to get stuck in local optima.

Consider an objective based search applied to this problem, where the fitness of a candidate solution is only determined by the distance it is from the goal. Solutions which guide the robots into these dead ends that are closer to the goal will be selected until the population converges and no further progress is made. This clearly demonstrates how unsuitable fitness functions can be in deceptive landscapes. A comparison from Lehman and Stanley's paper is shown in figure 4.1.

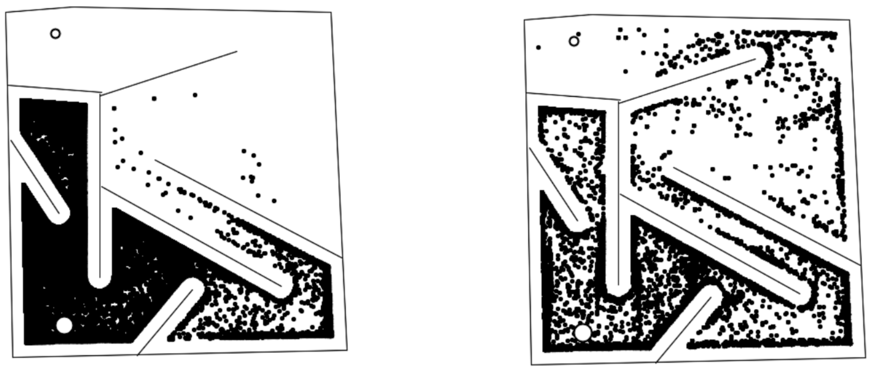
Figure 4.1: Comparison of end points of robots during the maze navigation task from Lehman and Stanley's 2008 paper *Exploiting Open-Endedness to Solve Problems Through the Search for Novelty* [2]. Fitness based search (left) cluster in the appealing dead end which is close to the goal. Novelty (right) manages to find the solution by rewarding the stepping stones along the path to the goal.

With a novelty metric that instead measures how different a solution is from others, even with no knowledge of the objective itself, Lehman and Stanley found that solutions were able to successfully navigate the maze task better than with an objective function. They mentioned that once every way to fail has been exhausted, a solution that finds the objective will appear novel.

Gravina et al. [4], found similar success in maze navigation with their surprise search. They found a comparable efficiency to novelty search and better performance than objective search. Compared to novelty search, surprise search was found to be more exploratory. This higher spatial exploration was credited to surprise search's ability to account for predicted behaviours not previously seen (and often not ever displayed).

Mengistu et al.'s evolvability search [5] found similar results when compared to novelty search with the cost of more evaluations needed due to simulating offspring. However, evolvability search was found to produce individuals with much higher capacity of offspring variation than novelty search even given the same computing power.

Lehman and Miikulainen [20] applied their combination of extinction events with a behavioural grid search for the task of maze navigation. The behavioural grid search places a grid over all possible behaviours and individuals that demonstrate a specific behaviour will fill the cell of the grid correspondingly. Each cell holds only a limited number of individuals so the search avoids convergence and more of the space must be explored. The researchers cite MAP-Elites [6] (section 4.2.4) as an inspiration behind the behavioural grid. In the task of maze navigation, behaviours are defined by the ending location of the robot traversing it. The behaviour grid makes up all possible ending locations. The results found that the extinction events improved the divergent search with more frequent extinctions having the most success.

Despite the success in this task, Lehman and Stanley noted that an optimized solution can be found only if an individual can reach a performance by appearing novel. The maze navigation task is really an idealized problem, tuned specifically to show off novelty search's success. They note that novelty search is limited in that there is nothing that tunes solutions towards an objective once good solutions are found. They suggest that using novelty along with objective functions could result in a balance of exploration and exploitation, where novelty can be used to approximate good solutions, and an objective function can polish solutions towards the goal.

4.2 Quality Diversity Search

Novelty search [2], surprise search [4], and evolvability search [20] are all examples of pure diversity searches. Objectives are discarded entirely in favour of putting selection pressure on population diversity. The success of these algorithms demonstrated that diversity can be used to drive solutions towards an objective with no other problem information. However, Pugh *et al.* [16], point out that these searches use diversity as a means to get to an objective, and ignore the intrinsic value that diversity brings.

With this, a new idea of *quality diversity* (QD) emerged, where the goal is to find a collection of behaviourally diverse individuals in all regions of the behaviour space, where each member of the collection performs with as high a fitness as possible. Since these algorithms can be said to highlight the fitness potential of areas of the search space, they have also been called *illumination algorithms* [6]. This idea of combining diversity with quality was alluded to in the novelty search paper. Chatzilygeroudis *et al.* also described the QD problem as a set of optimizations, each constrained by a different behaviour description [21]. Pugh *et al.* [16] mention possible applications that quality diversity offers beyond what optimization is capable of, such as with computational creativity and open-ended evolution.

4.2.1 Unifying Framework

Cully and Demeris proposed a single framework under which all quality diversity algorithms fall [22]. Each algorithm varies in a type of archive for previous solutions, selection operator, and the way solutions are scored. Once these are determined, the algorithm proceeds with an initial random population. Selection then creates a set of new solutions for evaluation.

Evaluation determines their scores and behaviour description. These solutions are compared to the container solutions and are added depending on if they meet the criteria for being added. Then the scores are updated and process repeats until a stopping criterion is reached.

4.2.2 Quality Diversity Performance

Chatzilygeroudis et al. identified two criteria for measuring performance in a QD algorithm [21].

- The fitness of each solution in the collection.
- The amount of behaviour space covered by the collection.

The first metric defines how the algorithm optimized to solve the task at hand. The second measures the amount of diversity.

4.2.3 Novelty Search with Local Competition

The first of these quality diversity algorithms comes from a 2011 paper by Lehman and Stanley which uses a combination of novelty search and objective search in an algorithm known as *novelty search with local competition* (NCLS) [23]. Combining novelty search with objective search may seem a simple case of multiple objectives with fitness and novelty on separate axes, but this ignores that different behavioural niches have different capacities for fitness. The paper uses the nature analogy that bacteria do not compete with bears, to illustrate the point further that global competition can impede diversity. In order to encourage diversity with the NSLC, competition between candidate solutions is restricted to those more similar to each other. Diversity is then rewarded indirectly since novel individuals can enjoy a new niche with less competition.

Lehman and Stanley highlight a trade-off with the quality diversity approach in that generally, solutions found in local competition tend to be less fit than those found through global competition. This is the cost of allocating more resources towards exploration and away from exploiting good solutions.

4.2.4 MAP-Elites

Another approach to quality diversity was proposed by Mouret and Clune known as multi-dimensional archive of phenotypic elites (MAP-Elites) [6]. With MAP-Elites, the user selects dimensions of variation in behaviour. For example, for a legged robot, one dimension could be the percentage of time each leg touches the floor, and another could be the height of the robot. This creates a multi-dimensional map of behaviours, known as a feature space, which is discretized into boxes. As the search space is explored, high performing individuals fill the cells of the feature map depending on if they meet the feature requirements for that cell and outcompete the current cell member in terms of fitness. The result is a multi-dimensional grid of high performing candidate solutions (called elites) where each cell is functionally different from every other. By creating a feature space displaying the highest performing elite at each point along multiple dimensions, the algorithm illuminates the effects of feature combination on solution performance, and uncovers the fitness potential in the feature map.

Algorithm 2 shows how MAP-Elites works. Initially, an N-dimensional map is populated with random solutions where each solution is placed into the cell corresponding to its behaviour description. Once the initial population is created, a random elite is copied from the map and the genetic operators (*crossover* and *mutation*) are applied to it. The resulting individual is then evaluated in terms of behaviour and the cell corresponding to that behaviour is checked. If that cell is empty, the individual is placed into it. If the cell is occupied, the performances are compared and if the new individual's performance is higher, then it replaces the current elite.

Some key differences between novelty search with local competition and MAP-Elites were highlighted in Mouret and Clune's paper [6]. For one thing, novelty search requires each candidate to be measured against every other candidate in terms of some diversity distance,

Algorithm 2: MAP-Elites

```
1  Initialize empty map of elites with $N$ dimensions, $M$
2  Initialize $maxIterations$
3  Initialize $initPopSize$ as the number of randomly generated individualls in the
     initial population
4  $i \leftarrow 1$
5  while $iterations\ i < maxIterations$ do
6      if $i < initPopSize$ then
7          Generate a random solution, $x'$
8      else
9          $x \leftarrow$ random elite from $M$
10         $x' \leftarrow$ copy $x$ and apply $crossover$ and $mutation$
11     end
12     $b' \leftarrow$ evaluate the behaviour of $x'$ according to the map behaviour dimensions
13     $f' \leftarrow$ evaluate the fitness of $x'$
14     if $M(b')$ is empty OR $f' > M(b').fitness$ then
15         $M(b') \leftarrow x'$
16     end
17 end
```

whereas with MAP-Elites, only the current occupant of the cell is compared. MAP-Elites does not need to keep track of an archive, which requires many more parameters that need to be fine-tuned or else performance suffers.

One domain that illustrates the effectiveness of MAP-Elites is in generating gaits for legged robots. Culley et al. [24] used the algorithm to create gaits for six legged robots. The feature space varied its dimensions by how long each leg touched the ground. The fitness was measured by distance covered in a fixed time interval. MAP-Elites was able to produce a large repertoire of high performing gaits. For example, the best performing gait with all six legs, the best performing gait with no front legs, etc. One example application of this in the real world would be to produce a robot for terrain traversal with many of these gaits programmed as options. If one of the robot's legs breaks down, change over to the gait that does not require that leg.

MAP-Elites + Novelty

MAP-Elites can be augmented with novelty. The variant known as MAP-Elites + Novelty (MENOV) has offspring added to an archive which allows for novelty scoring the members of the MAP-Elites grid [16].

MAP-Elites + Passive Genetic Diversity

MAP-Elites + Passive Genetic Diversity (MEPGD) addresses a potential pitfall in the elitism of MAP-Elites. Instead of only saving the best elite in each cell of the grid, two individuals are stored. The elite and another individual with a lower fitness. Any individual evaluated to have a lower fitness has a chance of being saved in the second spot of the slot no matter how low the fitness is [16].

Selection in MAP-Elites

An important factor driving the search in MAP-Elites is parent selection for producing offspring. Traditionally, this is done by selecting randomly from the elites. However, other selection methods have been proposed.

A 2020 survey on quality diversity by Chatzilygeroudis et al. [21] mentioned the *curiosity score* which models the probability that an individual's offspring is added to a cell. An individual with a higher curiosity score would then have a higher chance of being selected to reproduce. The survey found curiosity to be effective at balancing exploring interesting areas of the behaviour landscape while also optimizing solutions locally.

One paper by Sfikas et al. compared the traditional approach with using an upper confidence bound (UCB) balancing exploration and exploitation [25]. They compared two approaches with UCB. One prioritizing selection based on how many times an individual had already been chosen (individual-based) and the other prioritizing selection based on how many times and individual in a specific cell had been chosen (cell-based). Between these two, the individual-based approach was found to work better for deceptive landscapes and cell-based was better in other tasks. Exploitation of good solutions was not selected directly. Rather, an indirect approach was used factoring offspring survival, which refers to the rate that offspring are added to the collection either by discovering a new cell or by replacing another elite (similar to curiosity). The study found that the methods placing newer individuals or rarely selected individuals at a higher priority for selection improved the performance of MAP-Elites on all metrics.

Chapter 5

Measuring Diversity

All diversity search algorithms require some way to measure diversity. For solutions that only require tuning various parameters, this can easily be done by comparing the Euclidean distance between the parameters as vectors. For other solutions however, determining the distance between candidate solutions is not an easy or general task.

The earliest example of diversity, and the most intuitive, is genotypic. That is to compare the physical structures of the solutions. In a neural network, this would be to compare nodes and topology while in genetic programming, this would be comparing the tree structures of candidate solutions. Koza referred to the number of unique genotypes as *variety* [11].

Mouret and Doncieux pointed out some issues with this approach in their study on behavioural diversity in evolutionary robotics [18]. Computing the edit distance between graph structures is an NP-hard problem, making it infeasible to compute for every candidate at every generation. Another issue is that similarly structured graphs can lead to drastically different behaviours and structurally diverse graphs can have identical behaviours.

The more recent approach came with the introduction of novelty search. Rather than measuring differences in the space of genotypes, measure the phenotypic (behavioural) differences. This way, the NP-hard issues comparing graphs of neural networks or genetic programming trees is side-stepped. This approach is not as general as comparing structural differences, and instead requires a task specific distance to be created and computed. With this, researchers working with behaviour distance need to be careful not to inject human biases onto the diversity of behaviours.

In order to achieve a more general behaviour measure, Gomez [26] compared five generic measures of behavioural diversity. These measures were fitness, Euclidean distance of genotypes, Hamming distance of a binary vector representing the behaviour, relative entropy, and normalized compression distance of the binary behaviour vectors. In his experiments, the normalized compression distance measure resulted in the highest fitness, but the Hamming distance only performed slightly worse with much less computational cost.

Mouret and Doncieux [18] also compared several generic and task specific distances and found Hamming distance to be the most efficient. Task specific distances were the next most successful. They concluded that Hamming distance was an effective generic diversity measure that can be as efficient as task specific distances.

5.1 Multiple Behaviour Distances

Doniceux and Mouret [27] proposed a method for removing the choice involved with selecting a diversity measure by selecting them all. In their 2013 paper on using multiple behaviour distances [27] they tried two approaches: the first being to take the average of all the behaviour distance measures; and the second, less expensive option of randomly alternating between each of them. For their experiments, they used a combination of general and *ad hoc* measures, and found that using multiple behaviour distance measures resulted in better performance while also offloading the task of choosing a behaviour distance from the user. Switching between multiple measures was found to be as effective as averaging while also being more efficient. Provided that the task has multiple possible behaviour measures, Doncieux and Mouret's findings suggest that using as many as are available can increase performance, with a lower bound of four measures being enough to see significant improvement.

5.2 Feasible and Infeasible Space

In many problems with physical constraints, the search space can often be split into feasible and infeasible solutions. Consider the task of creating an efficient wing design for a plane. Some candidate designs during search or in the initialization may not allow for flight and would thus be infeasible. In the context of fitness-based optimization, infeasible individuals can be handled by some penalty to fitness. With a diversity search, this is not as simple. One of the main ideas behind diversity search is that stepping stones leading to a solution can be explored even if they do not achieve a better fitness. In the context of constrained search, infeasible individuals may eventually lead to feasible solutions through evolution.

Liapis et al. [3] proposed two algorithms that use novelty search to explore both the feasible and infeasible space of game level generation. Both algorithms maintain two populations, one of feasible solutions and the other of infeasible solutions. This way the solutions do not compete with each other. The first algorithm, feasible-infeasible novelty search (FINS) has the feasible solutions evolve to maximize their novelty from other feasible solutions while the infeasible population evolves to minimize its distance to the feasible population. The other algorithm, feasible-infeasible dual novelty search (FI2NS) has a novelty search performed on both populations independently from one another. Ideally, with FI2NS, the novelty search conducted in the infeasible space would push solutions towards islands of feasibility. An illustration from the paper of both algorithms is shown in figure 5.1. Liapis et al. hypothesized that the efficient search through infeasible space of FINS would allow it to discover feasible solutions more quickly than other search methods. They also assumed that FI2NS would find more diverse feasible solutions since novelty search is also conducted in the infeasible space. For the application of game content generation, these hypotheses were validated.

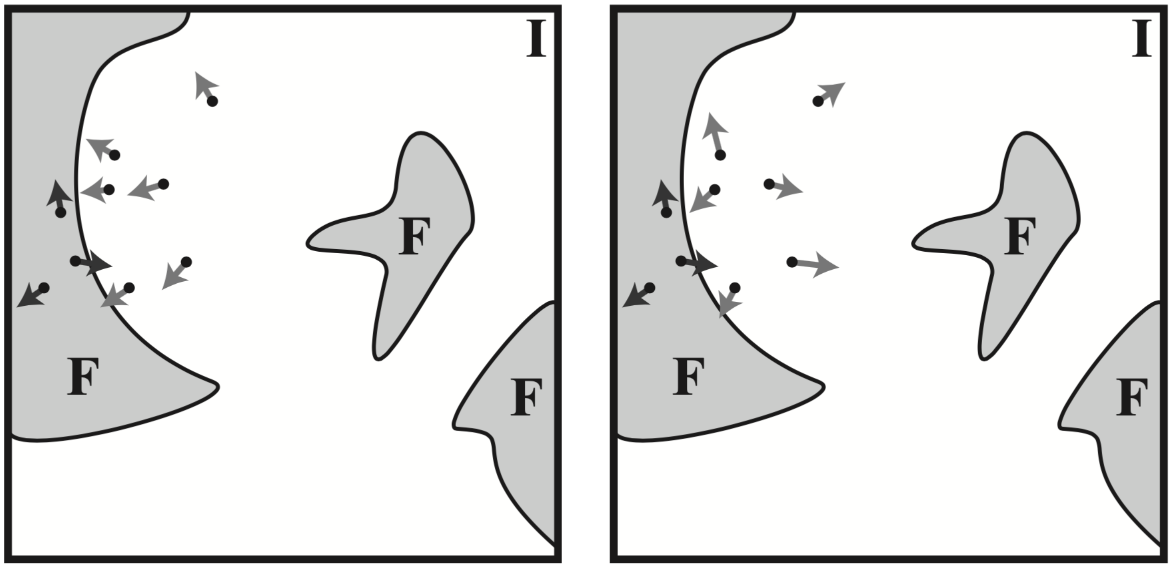

Figure 5.1: Illustration of FINS (left) and FI2NS (right) from Liapis et al.'s *Constrained Novelty Search: A Study on Game Content Generation* (2015) [3]

Chapter 6

Task Domains and Applications

Novelty search and diversity search have been applied to a number of task domains with great success. This section highlights some areas where the methods have been used.

6.1 Evolutionary Robotics

One of the most popular uses for diversity search is in evolutionary robotics. This field requires evolving morphologies, controllers, or both, that will accomplish some goal. Taking inspiration from natural evolution, researchers looked to this area to design robots in ways unbound by human biases. The stochastic elements that mutation brings can foster unexpected designs that improve the efficiency of robot morphologies. Stemming from this, it seems natural to apply diversity search to this domain in order to increase search exploration.

6.1.1 Swarm Robotics

Gomes et al. [15] explored using novelty search to evolve neuro controllers for swarm robotics. The first task they explored in this domain was swarm aggregation. This task requires a system of robots dispersed about the environment to form a cluster. This task is not deceptive like maze navigation, so an objective-based algorithm can perform well. The results found that using novelty search for this task led to fitness scores comparable to the objective-based search. Fitness-based search had robots exploring the arena in large circles until encountering another robot and then they would form a static cluster. If the cluster is small for long enough, the robots disperse and explore the arena again. Novelty search had robots moving towards the walls making sure to avoid other robots, then they move away from the wall in a trajectory that passes through the center. After some time, the robots form loosely near the center of the arena. Another wall behaviour had robots moving towards a wall and following it until they encountered another robot.

The second task examined was a resource sharing task. In this task, a swarm of robots share one resource and must coordinate to allow every member to access the resource periodically. The results of this experiment found that fitness-based search did not find a viable solution and often got caught in local optima. One of the behaviours observed had robots circling around. When one robot detects the charging station, it moves to it and remains there until all of the other robots die and the simulation ends. The other behaviour had all robots remaining still until they ran out of energy. Novelty search was able to achieve high fitness, since it has the ability to escape the local optima.

Three behaviours appeared in the successful runs. One of these had the robots in the charging station leave if they detected another robot approaching. They move about the arena for a while then return. The second behaviour had the robot in the charging station also leave if they detected an approaching robot but instead of moving about the arena randomly, they circle near the charging station until they need it again. The third behaviour had the robot in the charging station wait until its charge was above a certain value then it left and waited near the charging station without moving until it needed a charge again.

The paper concluded that novelty search was able to effectively sidestep the problem of convergence onto a local optimum. However, they did find that pure novelty search does not always find high-fitness solutions.

6.1.2 Underwater Robot Design

Corucci et al. [28] used novelty in the domain of robot design. The goal of this task was to improve upon a human-designed robot for underwater locomotion. The experiment used novelty search to explore the behaviour space of robots that resulted from morphological changes. Despite there being no objective defined for the novelty search, the researchers found that novelty search alone was capable of outperforming the objective-based search. For example, the horizontal speed achieved by novelty search outperformed that of the objective-based algorithm. More importantly, when comparing the exploration of the behaviour space, it was found that the objective-based search tended to have solutions that clustered into narrow regions where novelty search had a much more vast exploration.

Corucci et al. proposed that the strengths of the novelty search in this domain could be in a collaborative process where novelty search can offer a plethora of interesting morphologies for human designers to analyze and use as a starting point for their robot designs. This process does not have to be limited to underwater robots but to all areas of robot design.

6.1.3 Soft Locomoting Robot Morphology

The goal of soft locomoting robot morphology design is to evolve soft robots to move. Soft robots are made of deformable materials that are capable of adapting their shape. This allows for less rigid motion and much more degrees of freedom making them difficult for human designers.

Mouret and Clune [6] applied their MAP-Elites algorithm to this domain and compared it an objective-based algorithm. They found no significant difference in the global best solution found however the MAP-Elites algorithm found a much higher diversity of high performing solutions, illuminating interesting areas of the feature space that objective based search could not. True to the motivations of the algorithm, it is possible to use this map to discover the fitness potential of different features. One example of this was found to be that for any body size, if there is more bone in the soft robot, it moves slower.

6.1.4 Real Soft Robot Arm

The task of finding controllers for a real soft robot arm provides a different challenge. The robot arm is difficult to simulate which limits the evaluations and features that it can have. MAP-Elites was applied to this task with the goal of finding the boundaries of the arm in high, medium, and low ranges of movement [6]. MAP-Elites again was able to find many high performing solutions at all ranges demonstrating that the algorithm is effective, even with a small feature space and expensive evaluations.

6.2 Functional Generative Design

The domain of functional design in 3D printing is a challenging one. The search space is very high dimensional, and mechanisms that use 3D printed parts often require extensive postprocessing and reprints. On top of this, the software for designing is limited in terms of resolution, meaning that printing smaller, more detailed areas of the design is imprecise, and a lot of loss or unpredictable physical behaviours can come about as a result.

Wolfe et al. [29] took on this challenging evolving recurrent neural networks (RNNs) with novelty search to produce unique gear mechanisms. The RNNs add gears one at a time to a mechanism until a functional design is produced. Mechanisms with high novelty scores are added into the archive for future generations to refer to. Since this task requires real world designs, only valid gears should be produced. Any mechanisms that fall in the infeasible space are penalized heavily. This way, novel and feasible mechanisms will be produced.

One of the reasons Wolfe et al. gave for choosing novelty search over an objective function was due to the expense of producing each mechanism for evaluation. Also the low number of feasible solutions satisfying specific constraints on design makes exploiting a specific objective unproductive compared to looking for a diverse array of innovative designs.

The RNN model used with novelty search was found to be effective at generating diverse and feasible gear mechanism designs. Even with limited evaluations due to physical constraints, novelty search still managed to produce reliable and diverse designs.

6.3 Game Content Generation

A cheaper alternative to hand-designing levels in video games is to use procedural generation. This can happen during development to inspire level designers with interesting computer generated levels or during end-user gameplay where levels are automatically designed on-the-fly. Novelty search can be applied to this domain to always generate new levels that the user has never seen before. However, there has to be a balance between new content and levels that are both playable and fun for players. Liapis et al. [3] used their algorithms for constrained novelty search in the domain of game content generation. The novelty search was used to generate new levels by adding previously played levels to the archive. This way, players always have fresh content. The constraints were placed to provide satisfaction in playing as well as defining what a feasible or infeasible level was. Their FINS and FI2NS algorithms were able to find a diverse set of playable levels.

6.4 Competitive Coevolution Predator-Prey

In the domain of competitive coevolution, two populations evolve simultaneously. Each population's success depends on outcompeting the other population.

Gomes et al. [30] used novelty search in a predator-prey context. The predator agents evolve to capture the prey and the prey agents evolve to evade the predators. They note that often competitive coevolution can get caught in a cyclic convergence, where a limited set of strategies is used over and over with no new solutions being explored. Another convergence can occur if one population becomes so dominant over the other that no further improvement is possible.

Gomes et al. compared an objective-based competitive coevolution with novelty search on both populations, novelty search on only the predators, and a progressive minimal criteria novelty search on both populations, where novelty is explored only in regions found to have higher fitness. In terms of highest fitness, the pure objective-based search found higher fitness solutions than the other methods. However, the novelty-based approaches found a much higher diversity of solutions.

In terms of behaviours, the high fitness prey behaviours had little to no variation. They moved at full speed in circles around the arena using their proximity sensors to evade predators and avoid walls. Predator behaviours were more varied with the paper listing the following four as the highest fitness behaviours: (i) The predator moves backwards in a curved manner until it sees the prey. Then the predator stops and faces the prey. When the prey gets close, it attacks. (ii) The predator moves forwards in the arena at full speed avoiding walls until it detects the prey and then chases it at full speed. (iii) The predator moves slowly towards a wall and stops and turns to face the center of the arena. When it detects the prey, it chases it. (iv) The predator rotates until it sees the prey then it keeps it in its vision and moves backwards slowly. When the prey gets near, the predator attacks.

Interestingly, the progressive minimal criteria novelty search found comparable fitness to the purely objective-based search with the added benefit of finding a high diversity of behaviours. The paper concluded that when balanced with competitiveness, novelty is able to avoid convergence problems and consistently find new behaviours that a pure fitness-based search cannot.

Chapter 7

Conclusion

Quality-diversity search is a relatively new algorithm with the goal of highlighting areas of the search space with a diverse collection of solutions. Its benefits are farther reaching than promoting diversity to prevent premature convergence. This review has shown many different applications demonstrating benefits such as in creative inspiration where designers can use the diverse collection of solutions as starting points for designs [28, 3]. Quality diversity is also able to provide a repertoire of behaviours allowing for applications that can adapt with locally optimal behaviour constraints in unexpected conditions as with a legged robot who can pull one of these repertoires out in response to damage [6, 31].

In terms of future work, while emergent behaviour is an extensive area of research in the field of intelligent agents, relatively few studies have applied quality diversity techniques to intelligent agents and this may be an interesting avenue to explore.

Chatzilygeroudis et al. also highlighted open areas where current work is being done [21]. These include: (i) finding less expensive fitness functions which can slow down the evolution time especially in design areas where many real-world tests have to be conducted to determine the effectiveness of certain design; (ii) finding a way to deal with higher dimensional behaviour spaces since even three behaviour characteristics can exponentially increase runtime when evaluating solutions needing to be added to the container; (iii) defining behaviours which can be challenging if this requires expertise in the problem at hand or which can also run the risk of constraining the search too much to the biases of who is defining the behaviours.

Another aspect to consider moving forward in quality diversity is a shift in thinking. Applying the same assumptions and biases to quality diversity as are applied to objective-based searches can cause misleading comparisons. Often it is seen that quality diversity does no better or worse in terms of global optimization. However, this ignores the values that diversity itself brings. The problems that quality diversity solve and the metrics used to measure it should be considered separately from strict optimizers if their benefit is to be properly appreciated.

This review, which is by no means comprehensive, looked at an overview of diversity searches, quality diversity, concepts, and applications. Overall, quality diversity search is an interesting, new search algorithm that is able to illuminate the search space for a diverse collection of solutions.

Bibliography

[1] K. Deb, "An introduction to genetic algorithms," *Sadhana*, vol. 24, pp. 293–351, Aug. 1999.

[2] J. Lehman and K. O. Stanley, "Exploiting Open-Endedness to Solve Problems Through the Search for Novelty," *Proceedings of the Eleventh International Conference on Artificial Life (ALIFE XI)*, 2008.

[3] A. Liapis, G. N. Yannakakis, and J. Togelius, "Constrained Novelty Search: A Study on Game Content Generation," *Evolutionary Computation*, vol. 23, pp. 101–129, Mar. 2015.

[4] D. Gravina, A. Liapis, and G. Yannakakis, "Surprise Search: Beyond Objectives and Novelty," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, (Denver, Colorado, USA), pp. 677–684, ACM, July 2016.

[5] H. Mengistu, J. Lehman, and J. Clune, "Evolvability Search: Directly Selecting for Evolvability in order to Study and Produce It," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, (Denver, Colorado, USA), pp. 141–148, ACM, July 2016.

[6] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," *arXiv:1504.04909 [cs, q-bio]*, Apr. 2015.

[7] K. Stuart, “Think, fight, feel: how video game artificial intelligence is evolving,” *The Guardian*, July 2021.

[8] A. Hoffmann, “Artificial and natural computation,” in *International Encyclopedia of the Social Behavioral Sciences* (N. J. Smelser and P. B. Baltes, eds.), pp. 777–783, Oxford: Pergamon, 2001.

[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* USA: Addison-Wesley Longman Publishing Co., Inc., 1st ed., 1989.

[10] J. R. Koza and J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT press, 1992.

[11] J. R. Koza and R. Poli, “Genetic Programming,” in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques* (E. K. Burke and G. Kendall, eds.), pp. 127–164, Boston, MA: Springer US, 2005.

[12] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming.* Lulu Enterprises, UK Ltd, 2008.

[13] K. O. Stanley and R. Miikkulainen, “Evolving Neural Networks through Augmenting Topologies,” *Evolutionary Computation*, vol. 10, pp. 99–127, June 2002.

[14] S. Didi and G. Nitschke, “Hybridizing novelty search for transfer learning,” in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, (Athens, Greece), pp. 1–8, IEEE, Dec. 2016.

[15] J. Gomes, P. Urbano, and A. L. Christensen, “Evolution of swarm robotics systems with novelty search,” *Swarm Intelligence*, vol. 7, pp. 115–144, Sept. 2013.

[16] J. K. Pugh, L. B. Soros, and K. O. Stanley, “Quality Diversity: A New Frontier for Evolutionary Computation,” *Frontiers in Robotics and AI*, vol. 3, July 2016.

[17] S. W. Mahfoud, *Niching Methods for Genetic Algorithms.* PhD thesis, University of Illinois at Urbana-Champaign, 1995.

[18] J.-B. Mouret and S. Doncieux, "Encouraging Behavioral Diversity in Evolutionary Robotics: An Empirical Study," *Evolutionary Computation*, vol. 20, pp. 91–133, Mar. 2012.

[19] A. Salehi, A. Coninx, and S. Doncieux, "BR-NS: an Archive-less Approach to Novelty Search," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '21)*, (Lille, France), pp. 172–179, ACM, July 2021.

[20] J. Lehman and R. Miikkulainen, "Enhancing Divergent Search through Extinction Events," in *Proceedings of the Annual Conference on Genetic and Evolutionary Computation (GECCO '15)*, (Madrid, Spain), pp. 951–958, ACM, July 2015.

[21] K. Chatzilygeroudis, A. Cully, V. Vassiliades, and J.-B. Mouret, "Quality-Diversity Optimization: a novel branch of stochastic optimization," in *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*, vol. 170 of *Springer Optimization and its Applications*, pp. 109–135, Springer, Cham, Jan. 2021.

[22] A. Cully and Y. Demiris, "Quality and diversity optimization: A unifying modular framework," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 245–259, 2018.

[23] J. Lehman and K. O. Stanley, "Evolving a diversity of virtual creatures through novelty search and local competition," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '11)*, (Dublin, Ireland), pp. 211–218, ACM, July 2011.

[24] A. Cully and J.-B. Mouret, "Evolving a behavioral repertoire for a walking robot," *Evolutionary Computation*, vol. 24, pp. 59–88, 2016.

[25] K. Sfikas, A. Liapis, and G. N. Yannakakis, "Monte Carlo Elites: Quality-Diversity Selection as a Multi-Armed Bandit Problem," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '21)*, (Lille, France), pp. 180–188, ACM, June 2021.

[26] J. Gomes, P. Mariano, and A. L. Christensen, "Devising Effective Novelty Search Algorithms: A Comprehensive Empirical Study," in *Proceedings of the Annual Conference on Genetic and Evolutionary Computation (GECCO '15)*, (Madrid, Spain), pp. 943–950, ACM, July 2015.

[27] S. Doncieux and J.-B. Mouret, "Behavioral diversity with multiple behavioral distances," in *2013 IEEE Congress on Evolutionary Computation*, (Cancun, Mexico), pp. 1427–1434, IEEE, June 2013.

[28] F. Corucci, M. Calisti, H. Hauser, and C. Laschi, "Novelty-Based Evolutionary Design of Morphing Underwater Robots," in *Proceedings of the Annual Conference on Genetic and Evolutionary Computation (GECCO '15)*, (Madrid, Spain), pp. 145–152, ACM, July 2015.

[29] C. R. Wolfe, C. C. Tutum, and R. Miikkulainen, "Functional Generative Design of Mechanisms with Recurrent Neural Networks and Novelty Search," Mar. 2019. arXiv: 1903.10103.

[30] J. Gomes, P. Mariano, and A. L. Christensen, "Novelty Search in Competitive Coevolution," in *Parallel Problem Solving from Nature – PPSN XIII* (T. Bartz-Beielstein, J. Branke, B. Filipič, and J. Smith, eds.), vol. 8672 of *Lecture Notes in Computer Science*, pp. 233–242, Springer International Publishing, 2014.

[31] J. Nordmoen, F. Veenstra, K. O. Ellefsen, and K. Glette, "MAP-Elites Enables Powerful Stepping Stones and Diversity for Modular Robotics," *Frontiers in Robotics and AI*, vol. 8, Apr. 2021.