Applied Soft Computing 136 (2023) 110121

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Tools for Landscape Analysis of Optimisation Problems in Procedural Content Generation for Games

Vanessa Volz [a,*], Boris Naujoks [b], Pascal Kerschke [c], Tea Tušar [d]

[a] modl.ai, Denmark
[b] TH Köln – University of Applied Sciences, Germany
[c] TU Dresden & Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI) Dresden/Leipzig, Germany
[d] Jožef Stefan Institute, Slovenia

ARTICLE INFO

Article history:
Received 24 July 2020
Received in revised form 9 February 2023
Accepted 10 February 2023
Available online 13 February 2023

Keywords:
Optimisation
Search-based procedural content generation
Exploratory Landscape Analysis
Mario level generation

ABSTRACT

The term Procedural Content Generation (PCG) refers to the (semi-)automatic generation of game content by algorithmic means, and its methods are becoming increasingly popular in game-oriented research and industry. A special class of these methods, which is commonly known as search-based PCG, treats the given task as an optimisation problem. Such problems are predominantly tackled by evolutionary algorithms.

We will demonstrate in this paper that obtaining more information about the defined optimisation problem can substantially improve our understanding of how to approach the generation of content. To do so, we present and discuss three efficient analysis tools, namely diagonal walks, the estimation of high-level properties, as well as problem similarity measures. We discuss the purpose of each of the considered methods in the context of PCG and provide guidelines for the interpretation of the results received. This way we aim to provide methods for the comparison of PCG approaches and eventually, increase the quality and practicality of generated content in industry.

© 2023 Elsevier B.V. All rights reserved.

1. Introduction

Search-based procedural content generation is a very popular approach for generating various types of content for games, such as levels (for example for Super Mario Bros., see Fig. 1) and weapons [1]. They work by formulating the generation process as an optimisation problem, where the task is to identify content that fulfils a given objective best. According to [1], in order to define this optimisation problem, the following three components need to be specified: (1) a suitable *representation* / search space for the content that can be searched (more details in Section 3.1) by (2) an (optimisation) *algorithm*, which in turn is guided by (3) a suitable *fitness function* (more details in Section 3.2). Most previous publications on search-based PCG focus on one aspect of the problem definition and treat the remaining components as given [2], which limits our understanding of the problem as a whole.

Of course, focusing only on one component allows more in-depth analyses and streamlined discussions. However, taking a more holistic and application-agnostic approach to analysing search-based PCG systems instead comes with several potential benefits. This means that all interconnections between the different components of a PCG system are accounted for in the analysis. For example, a better understanding of the fitness landscape, which is a product of representation and fitness function, can allow to select a suitable search algorithm [3–7]. This selection is important as, according to the *no free lunch* theorem [8], no single algorithm can perform well on all types of problems. Aspects such as noise levels, number of local and global optima, or the landscape's ruggedness should be considered when selecting an algorithm [4,9–11]. In addition, knowledge about global optima

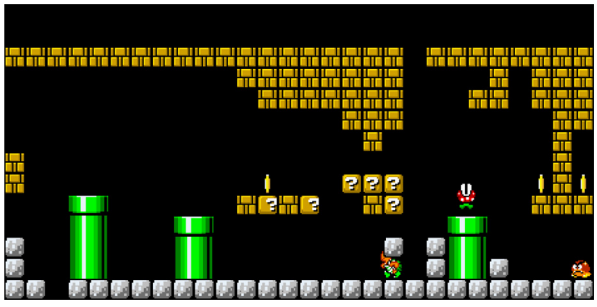


Fig. 1. Exemplary (underground) level of Super Mario Bros.

* Corresponding author.
E-mail address: vanessa@modl.ai (V. Volz).

https://doi.org/10.1016/j.asoc.2023.110121
1568-4946/© 2023 Elsevier B.V. All rights reserved.

can help to assess the success of the PCG approach and the potential for further optimisation.

This information is not only helpful for the selection of an optimisation algorithm, but can also be utilised in order to choose the best representation and fitness function possible. For example, if the dimension of the representation is variable, information about the scalability of the PCG approach can help to determine the smallest (i.e. lowest dimensional) representation that still offers sufficient detail and variety. Usually, the smaller the representation, the easier it is to find fit solutions. In addition, the content is usually intended to fulfil objectives around abstract game-related concepts, such as, e.g., game difficulty. These concepts, however, can usually be expressed and implemented in several ways, for example using different Artificial Intelligence (AI) game-playing agents for simulation. These different implementations might not create fitness landscapes of the same type, and consequently influence the attainable content.

Finally, a more extensive analysis of search-based PCG applications from the standpoint of optimisation algorithms also produces information on the robustness of the proposed approach. In this context, robustness is especially important with regards to the reliability of a content generator to produce similar content and thus fulfil expectations, for example regarding its playability. Further, different training examples and initialisation procedures could be tested to investigate the applicability of a given PCG algorithm to different games. This is especially true if all three components, i.e., representation, fitness function and search algorithm can be varied. Results from this type of analysis also facilitate comparisons between different content generators.

Research in evolutionary computation has been applying several methods of analysis for understanding and improving the behaviour of optimisation algorithms for several years [12,13]. In this paper, we

1. present and discuss how one can apply several of these landscape analysis methods to (search-based) PCG;
2. summarise how the gained insights can – and should – be used to evaluate and improve PCGs; and
3. demonstrate the methods' applicability by means of an exemplary use case in which we analyse generated Mario levels.

In order to keep the paper concise, the methods we discuss in more detail in this paper are all intended for the analysis of fitness landscapes of black-box problems, independent of the applied optimisation algorithm. This type of analysis is often a first step and especially relevant for PCG approaches due to the lack of suitable existing information helpful for selecting an optimiser.

We are further only describing the proposed methods in the context of continuous search spaces, since that is the type of problem we have chosen as an example application. All the required concepts also exist for optimisation problems with non-continuous search spaces, however, and are thus transferable.

Our contribution in this paper is thus a **vision and tutorial to study optimisation problems in search-based PCG from the perspective of systematic analysis tools**. To support our vision, we further provide exemplary results for a benchmark called GBEA (Game Benchmark for Evolutionary Algorithms [14]), which is based on PCG applications. We are able to demonstrate that the analysis we propose is useful to strengthen the interpretability and thus usefulness of the aforementioned benchmark.

In the following, we first give some background information, starting with related work on landscape analysis as well as search-based PCG in Section 2. We also specifically survey research that is intended to obtain more information on PCG approaches and identify a definite lack of such work. The general experimental setup for the experiments conducted in the following sections is provided in Section 3. In the second part of the paper, we propose several analysis tools and demonstrate their usefulness using an exemplary problem from the GBEA benchmark. Tools that are discussed are diagonal walks in Section 4, the estimation of structural high-level properties in Section 5, and problem similarity measures in Section 6. Each of these sections contains a short explanation of the method in question, followed by a description of its purpose in the context of PCG. An overview of the discussed tools and their purpose in PCG is given in Fig. 2. We then demonstrate each tool's applicability by applying it to our example. Each section is concluded by a short discussion of the respective method's suitability, strengths and weaknesses. We conclude the paper in Section 7 with a summary of our concrete findings. Finally, we discuss our vision of using optimisation analysis tools to improve PCG research in the future.

2. Related work

In the following, we give an overview of related work. We start with an overview of Exploratory Landscape Analysis, which we use as an example of current approaches to understanding fitness landscapes in black-box optimisation problems. Next, we conduct a brief survey of the state-of-the-art of search-based PCG, focusing specifically on work that analyses the optimisation problems in PCG in more detail.

2.1. Exploratory landscape analysis

Exploratory Landscape Analysis (ELA), sometimes also called fitness landscape analysis, stands for a sophisticated method that employs automatically computable (mainly numerical) values to extract representative information from a problem's landscape [10,11,15]. These numbers, the so-called *features*, can be used to derive more tangible properties of the landscapes, such as whether they are rugged [16], possess an underlying funnel structure [17], or contain plateaus [10]. As they provide cheap and explicitly measurable surrogates for these (hardly quantifiable) high-level properties, automatically computable landscape features enable more sophisticated analyses on the problems at hand such as examining the underlying problem spaces [18–21], characterising algorithmic search behaviour [22,23] as well as selecting and configuring well-performing optimisation algorithms ([4–7,24–27], respectively). Note that in order to save valuable resources, features are ideally computed on a very small set of sampled points [17,28].

Over the decades, a plethora of features have been proposed, which makes a detailed discussion in this work infeasible. Instead, we refer the interested readers to [3,5,11,29] for comprehensive overviews of recent works on this topic. For now, we only provide two features to give an example of typical features:

1. the coefficient of determination $R^2 \in [0, 1]$ (i.e., the quality) of a quadratic model that has been fitted to the sampled data [10], and
2. the ratio between (i) the average distance of the sample points to their respective nearest neighbour, and (ii) the average distance to their nearest *better* neighbour (i.e., the nearest neighbour among all observations with a better fitness value) [17].

Both features are useful when trying to distinguish between rather unimodal functions ($R^2 \approx 1$) and random landscapes that rather look like an egg box ($R^2 \approx 0$).

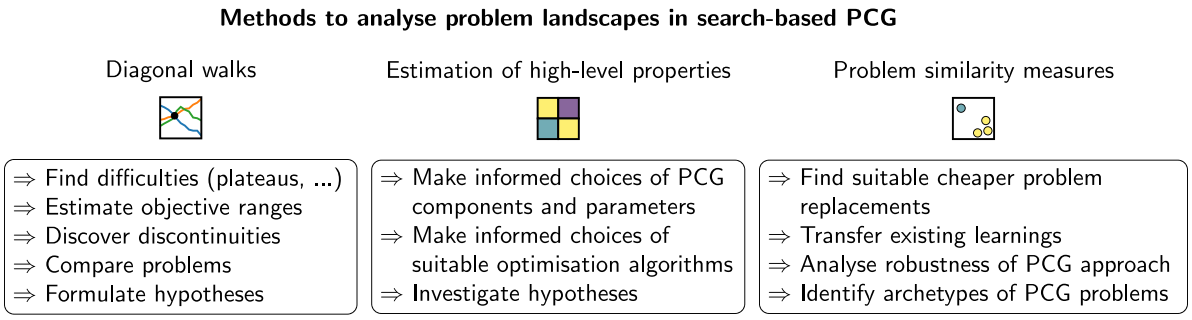


Fig. 2. Overview of the proposed methods and their purpose in the context of PCG. Diagonal walks are described in Section 4, the estimation of structural high-level properties in Section 5, and problem similarity measures in Section 6.

2.2. Search-based procedural content generation

The (semi-)automatic generation of game content through algorithmic means is commonly referred to as procedural content generation (PCG) [1]. Various approaches to PCG exist, but an especially popular subset is dubbed search-based PCG [1,30]. In the corresponding approaches, some notion of fitness is required to evaluate generated content. The "fittest" content can then be discovered by exploring the content-representing search space, for which the fitness values are given by the fitness function. Most commonly, the search is performed by evolutionary algorithms, likely due to their ability to handle black-box fitness functions [1, 31].

As in any optimisation problem, the difficulty of finding good content in search-based PCG depends on the structural challenges posed by the fitness landscape (i.e., the combination of representation and fitness function), as well as the search algorithm operating on it. However, previous work on the analysis of search-based PCG as optimisation problems often lacks a holistic approach.

For example, numerous publications address the difficulty of finding suitable representations for game content. Novel representations for different subsets of game content are proposed regularly (see [32] for an overview). One often addressed challenge of representations is the dimension of the search space [30], as the chosen representation is required to map to relatively complex content (e.g. a whole level). Still, if the dimension is reduced too much, the variety of valid content might be limited, resulting in less interesting gameplay. Other properties of representations that are often addressed are the locality [30] and the scarcity of valid solutions [33, pp. 93–95]. In [34], finding good representations is identified as an open problem. Recent efforts have been targeting the creation of such rich search spaces for game content [35].[1] Building on this, newer efforts have been aimed at transforming these search spaces to facilitate exploration of interesting parts of the search space [36].

The problem of defining fitness functions, however, is usually addressed separately from the choice of representation. This is because finding an automatic evaluation function for generated content without feedback from human players is an ill-posed problem, as subjective human perception needs to be expressed and formalised [30]. For this reason, numerous fitness functions have been proposed in literature, especially for heavily researched games, such as board- [37] and platform games [38]. Several (largely agreeing) attempts have also been made in order to characterise these evaluation functions [33, pp. 122–125], [1,30,34].

Existing evaluation approaches of both components (fitness function and representation) largely revolve around exploratory assessment of generated content, especially by visualising its variety — for example via expressive range analysis [2]. The improvement of the search algorithm's fitness over iterations is usually reported as well, but this only offers limited interpretability if the fitness landscape is unknown.

It is therefore indispensable to analyse the fitness landscapes of search-based PCG. However, related work in this area is sparse. There are a few publications where search spaces are characterised based on the performance of search algorithms. For example, in [39], several versions of an evolutionary algorithm are used to optimise landscape automata based on two fitness functions. The study, however, focuses on identifying good parameters for the evolutionary algorithms and for the representation, instead of executing a landscape analysis. Nevertheless, this type of analysis allowed the authors to characterise the landscape as multimodal, i.e., containing multiple local and/or global optima [40]. Another publication [41] uses a similar approach in order to describe the available gameplay to a beginner in a popular trading card game. The fitness function of evolving decks was shown to be jagged (i.e., it showed small local irregularities) and in their experiments, but the results are mainly anecdotal and tied to the optimisation approach

The only formal landscape analysis of search-based PCG we were able to find is based on a specific representation for maze levels, called apoptotic cellular automata [42]. The study is based on a survey of fitness landscape analysis methods [43], which were adapted appropriately for application in this context. The authors find that the fitness landscape is rugose, i.e., it possesses a multitude of local optima and large plateaus with low fitness. Based on their analysis, the authors see a similarity to Shekel's foxhole function [44].

Unfortunately, the results are difficult to generalise. The problem in the study uses a specific, non-standard representation, while in search-based PCG, continuous representations are usually preferred [30]. In addition, only a single fitness function is investigated. As evidenced by taxonomies and surveys [34], there is a variety of fitness functions, which will likely produce very different landscapes.

In this paper, we extend previous work in several ways. We analyse a problem with a more common representation, in conjunction with several (28) fitness functions.

Some literature can also be found on analysing the landscapes of games from the perspective of a game-playing agent. A recent example of this type of work is [45], where various games are characterised based on graph representations of playthroughs. The authors propose to use the resulting representation to be able to compare different games and to facilitate the understanding of game agents. In contrast, in this paper, we compare different problems in search-based PCG to facilitate the understanding of PCG algorithms. The results and representations are thus unfortunately not transferable.

[1] Brief video explanation: https://www.youtube.com/watch?v=NObqDuPuk7Q

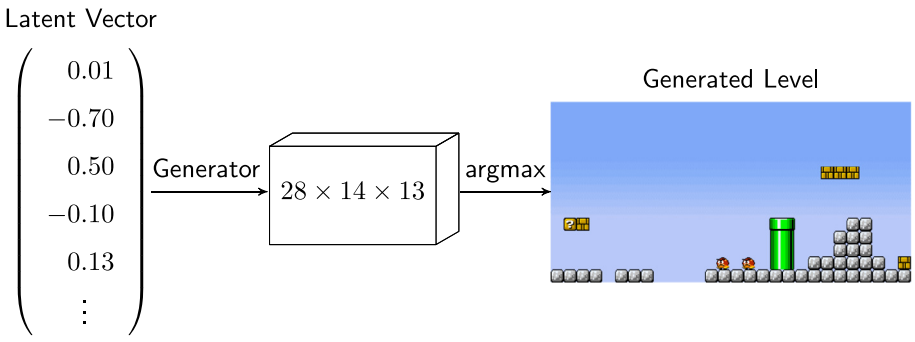


Fig. 3. A schematic representation of the level generation process using MarioGAN.

3. Example application MarioGAN

Super Mario Bros., or Mario for short, is a classic platform game ("platformer") and has been targeted by various research efforts, including competitions and publications on game AI [46], as well as procedural generation of levels [47]. A survey of several fitness functions that have been used in previous research can be found in [38].

As is clear from several surveys, a multitude of level generators for Mario [47] and similar platformers exist. However, as described in the previous section, the search landscapes of the corresponding generators have not been analysed in detail. Several publications address different aspects of the problem, such as fitness functions [38] or search space [35]. As we are not aware of any holistic analyses of optimisation problems related to Mario level generation, we are using this application as an example to demonstrate the methodology we propose in this paper. We thus select a previously published search-based PCG approach (dubbed MarioGAN [35]) that has certain representative characteristics, i.e., the representation is a continuous vector [30] and suitable for evolutionary algorithms [1,31].

3.1. Search space

For our analysis, we use the level generator proposed in [35], which we will call MarioGAN in the following. The generator is a neural network that takes an input vector with values between -1 and 1 and produces 13 matrices of size 28×14. The search space is thus bounded and continuous even though the representation of the levels is discrete. The size of the input vector just depends on the structure of the neural network and can be chosen arbitrarily. These are translated to 28×14-dimensional Mario levels using a binary encoding of 13 different tile types. A visualisation can be found in Fig. 3. The generator is trained using an adversarial approach (Generative Adversarial Networks — GANs) as suggested in [48,49] and is trained on Super Mario Bros. original levels contained in the video game level corpus [50].

Using this approach, generators for different input space dimensions can be trained. This allows us to analyse the scaling behaviour of the executed search algorithms. Furthermore, the levels are generated within milliseconds, resulting in practical experiments in terms of computational resources.

3.2. Fitness functions

For our analysis, we are using the fitness functions proposed in the game-benchmark for evolutionary algorithms described in [14]. These functions are based on the state of the art of platformer evaluation [38] and selected to produce diverse fitness landscapes. A short description of the measures computed for the functions along with their original sources is found below and a more formal definition with more details on how they are transformed into minimisation problems can be found in Appendix:

enemyDistribution: Horizontal distribution of enemies across the level. The more enemies are grouped together, the more difficult it is to evade them. Measured as standard deviation of the enemies' x-axis coordinates [38].

positionDistribution: Vertical distribution of platforms across the level. Platforms at various different heights typically seem interesting to a player. Measured as standard deviation of y-axis coordinates of tiles one can stand on [38].

decorationFrequency: Amount of non-standard tiles in the level, i.e., tiles that make the level interesting. Measured as fraction of *pretty tiles* := {Tube, Enemy, Destructible Block, Question Mark Block, or Bullet Bill Shooter Column} [38].

negativeSpace: Amount of empty space in the level. This characterises the way the player moves through the level. Measured as the fraction of tiles one cannot stand on [51].

leniency: A quantification of how easy it is to pass the level. Measured as weighted sum of subjective *leniency* of tiles as defined in [52].

basicFitness: Difficulty for an AI to pass the level (according to the score used in the MarioAI championships [46]). Measured as a linear combination of several performance-related aspects, such as the distance of the level that was covered and the amount of collected coins.

airTime: Describes how much the AI agent jumped to pass through the level. Jumping is the main mechanic in Mario, so a level should require a considerable amount of it. Measured as the ratio between time in the air divided by time spent on the level. If the level is not completed, a penalty value is returned instead [35].

timeTaken: Time required by the AI to navigate through the level — longer times likely involve some backtracking or other challenging parts. Measured as the ratio between time taken and total time allowed for the level. If the level is not completed, a penalty value is returned instead [35].

3.3. Optimisation problems

Our set of MarioGAN optimisation problems is defined by combining the search space (Section 3.1) with the fitness functions (Section 3.2). Note that, without loss of generality, all problems have been turned into minimisation problems. In addition, several variations of each problem are provided to investigate

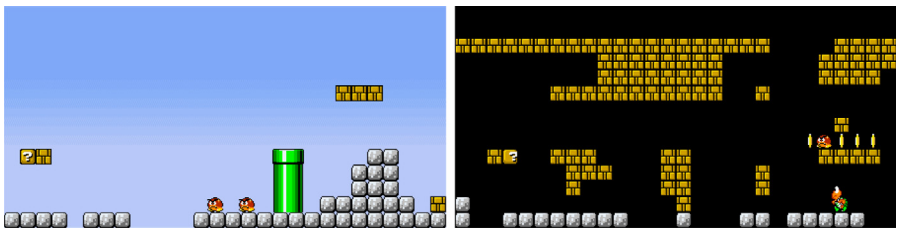


Fig. 4. Exemplary overworld (left) and underground (right) levels.

Table 1
Overview and characterisation of the 28 fitness functions in the Mario suite ($m_1 - m_{28}$). A* refers to the winner of the first MarioAI gameplay competition [46], which is based on an A* algorithm. Scared refers to *ScaredAgent*, one of the default agents within the MarioAI competition framework, which tries to avoid all obstacles and enemies by jumping. Letters [o] and [u] specify the training levels used and [c] whether level segments need to be concatenated.

Fitness measure	AI	o	u	oc	uc
enemyDistribution	–	m_1	m_2		
positionDistribution	–	m_3	m_4		
decorationFrequency	–	m_5	m_6		
negativeSpace	–	m_7	m_8		
leniency	–	m_9	m_{10}		
basicFitness	A*	m_{11}	m_{12}	m_{13}	m_{14}
	Scared	m_{15}	m_{16}		
airTime	A*	m_{17}	m_{18}	m_{19}	m_{20}
	Scared	m_{21}	m_{22}		
timeTaken	A*	m_{23}	m_{24}	m_{25}	m_{26}
	Scared	m_{27}	m_{28}		

different questions, resulting in a total of 28 problems listed in Table 1. Only variations resulting in functions with interesting and diverse fitness landscapes are included in the benchmark. These variations are:

- AI: Two different AIs (Baumgarten's A* and Scared [46]) are implemented to simulate player behaviour for functions that require it.
- Training levels: GANs are trained separately on two disjoint sets of Mario levels, overworld (o) and underground (u) (see the images in Fig. 4).
- Concatenation: Multiple level segments generated by a GAN can be concatenated (c) to create a longer level.

Instances of each problem (as defined in [53]) are created by varying the random seed used to initialise the training of the generators. In this paper we consider seven instances for each MarioGAN problem.

4. Diagonal walks

In the following, as well as in the two upcoming sections, different methods for analysing PCG approaches are discussed in detail (they are also presented in Fig. 5). We start with the most basic method, diagonal walks, which are a simple tool for a first analysis of optimisation problems.

4.1. Method description

According to [15,43], landscape walks are a useful tool for landscape analysis. Following the example from [14], we perform a type of landscape walks called diagonal walks through a random point. We generate a random anchor point that represents a valid solution and a random vector representing a direction. Together, they define a line through the search space that is limited by the search space boundaries. Then, we "walk" on this line from one end to the other in equidistant steps passing through the anchor point. As the randomly chosen directions are generally not axis-aligned, this means that all variable values are changed concurrently. Because the position and the direction of such walks are random, the number of available steps differs from walk to walk, the anchor point is not necessarily at the centre of the walk and the starting and ending points of the walk are not necessarily on the search space boundary. These walks are repeated several times by using different directions for the same anchor point in order to put the different walks into context of each other. Furthermore, to increase coverage, this procedure should be repeated with various anchor points.

In our example, we perform diagonal walks with a single anchor point and three different directions. In their visualisations (see Fig. 6), the x-axis shows the number of steps along the diagonal line (with the anchor point being positioned at $x = 0$), and the y-axis depicts the corresponding function values. To enable a direct comparison of different optimisation problems, we use the same anchor point and directions throughout our analysis in this paper.

4.2. Purpose in context of PCG

As complex game content must be presented in a way that is comprehensible for evolutionary algorithms, the genotype-phenotype mappings used in the context of search-based PCGs are often very complex. In addition, the selected genotype might require the definition of non-standard variation operators in order to ensure that all generated content is valid.

For this reason, it is helpful to gain insights into how an algorithm explores the space of the generatable content. Diagonal walks are an effective tool to achieve this purpose. While they certainly do not provide a holistic picture, these walks can still serve as early indicators of difficulties that the search algorithm may face, such as the existence of plateaus or many narrow basins of attraction (multi-modality). If such issues are discovered, different options for representation and variation operators can be explored. Otherwise, a suitable optimisation approach can be chosen by considering the expected difficulties. For example, restarts are a common approach for handling multi-modal landscapes.

Furthermore, it is usually difficult to judge what is a comparatively good fitness value. Diagonal walks are a useful tool for a rough approximation of the achieved value ranges. Also helpful in this regards are statistics on the distribution of fitness values from random samples of the search space. Diagonal walks, however, give an impression of the relative positions of the sample solutions in the search area. This can help to identify whether the fitness landscapes contain discontinuities in objective space or whole areas with good objective values.

An additional important benefit of diagonal walks comes from the fact that many approaches rely on simulation-based measures computed during playthroughs with AI players [54, Appendix A]. Thus, comparing the fitness values computed from different AI

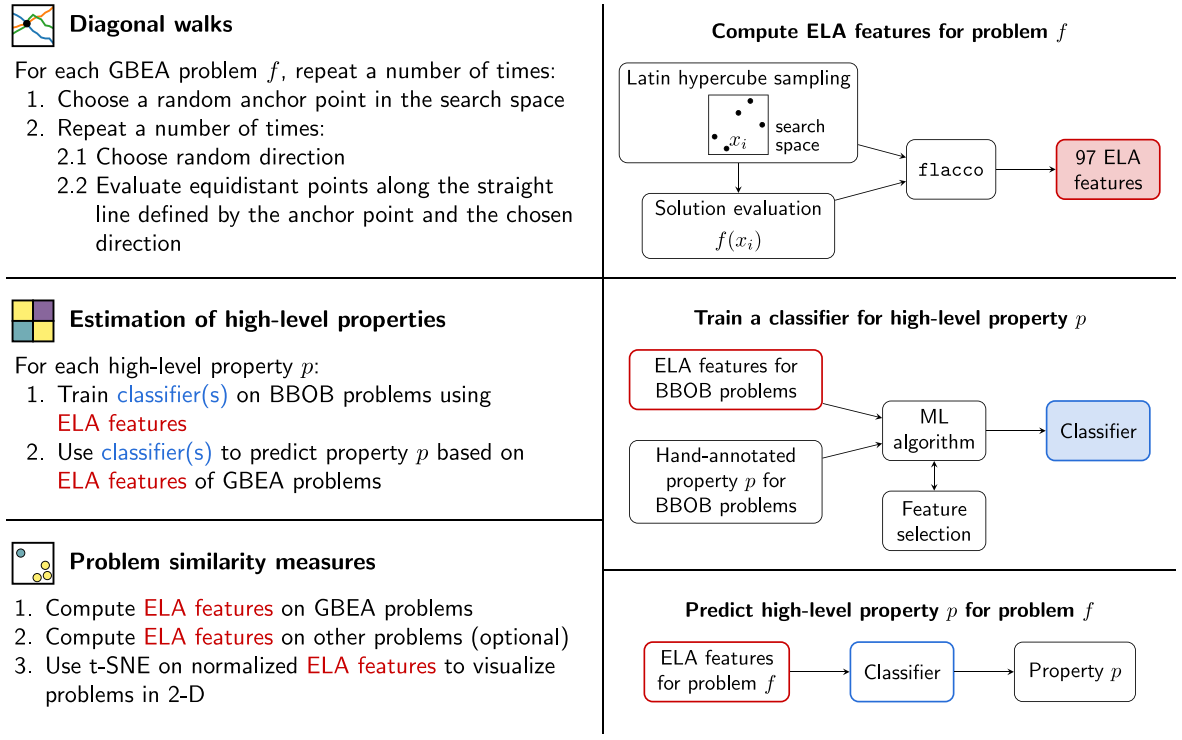
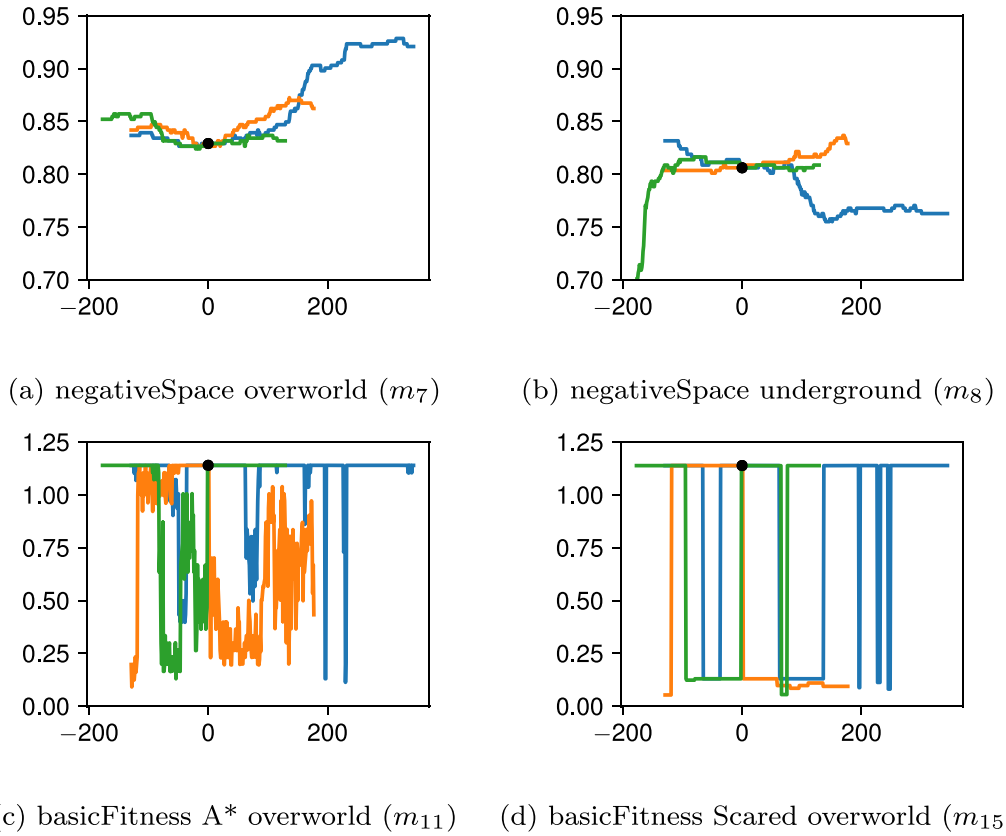


Fig. 5. Pseudocode for diagonal walks, estimation of high-level properties and problem similarity measures on the left hand side with additional explanation of the nontrivial steps on the right hand side. The derivation of ELA features (red) and classifiers (blue) is highlighted in the right hand part as well as their utilisation in the whole figure.



(a) negativeSpace overworld (m_7) (b) negativeSpace underground (m_8)

(c) basicFitness A* overworld (m_{11}) (d) basicFitness Scared overworld (m_{15})

Fig. 6. Diagonal walks for (the first instance of) different fitness functions. Their problem IDs (see Table 1) are given in brackets. The different colours indicate separate walks through the same anchor point (black dot, centred at $x = 0$). The x-axes show the number of steps taken in each direction from the anchor point, whereas the y-axes show the respective fitness values.

simulations at the same points (i.e., same generated content) can give first insights into differences and similarities of player behaviours. This analysis is especially meaningful if diagonal walks from AI playthroughs are compared with corresponding human ones.

4.3. MarioGAN results

4.3.1. Analysis

The diagonal walks for some representative problems in MarioGAN can be found in Fig. 6. Several interesting observations can be made based on these plots. For example, the genotype-phenotype mapping as described in Section 3.1 maps a continuous latent space to a discrete level. Steps in the search space thus translate to the addition, removal, or swapping of one tile in a level to another. If a fitness function (such as negativeSpace) is computed directly on the level encoding, we would thus expect a step-like landscape with small steps indicating when tiles change. Exactly this behaviour can be observed in Fig. 6(a) and (b). Although the size of the steps differs for different instances and areas of the search space, it is present for all problems with representation-based fitness functions ($m_1 - m_{10}$).

Despite being based on the same genotype-phenotype mapping, steps cannot be observed when using simulation-based fitness functions (problems $m_{11} - m_{28}$). This is likely because changing a single tile in the level does not necessarily result in different agent behaviour. As such, the fitness measure basicFitness (i.e., the performance of the AI) as depicted in Fig. 6(c) and (d), for example, does not necessarily change if a platform is added that cannot be reached by Mario. However, the addition of a single enemy can significantly affect player behaviour and thus the resulting score.

The size of these effects also depends on the agent used for simulations. This can be seen when comparing the scores that A* and the ScaredAgent received for the same levels as shown in Fig. 6(c) and (d), respectively. The naive ScaredAgent is more sensitive to changes, producing a widely varying performance. While the basicFitness still varies significantly for the A* agent overall, the variation per step is much smaller. Still, the lack of smoothness in both functions should be a large concern when picking an optimisation algorithm. Problems like the ones discussed above with a low locality are difficult for standard evolutionary algorithms [55], for example.

Another observation is that the achieved fitness values for negativeSpace tend to be lower for underground levels (m_8) than overworld levels (m_7). This is expected, as underground levels mimic tunnels in dungeons and are therefore capped by ceilings. We have visually verified that this is indeed the case for the generated levels (see Fig. 4). Through further experiments – for which a detailed description would exceed the scope of this paper – we were also able to demonstrate the observed tendency empirically.

4.3.2. Interpretation

Based on the above analysis, it seems that evolutionary algorithms are well suited for optimising problems $m_1 - m_{10}$ with representation-based fitness functions, as locality assumptions are fulfilled. This seems to indicate that the chosen genotype-phenotype mapping is suitable for the generated content.

However, problems with simulation-based fitness functions show landscapes with large local variations. This might require the development and application of techniques that are suited to handle such variations appropriately. In addition, the cause for these large variations should be investigated in more detail. One possible explanation for the variations is the noise in the fitness evaluations that is not taken into account, caused either by the AI or the physics engine.

The large differences between fitness landscapes for different AI players also demonstrate how sensitive content evaluations react to the actual AI implementation used for the simulation. This illustrates the potential pitfalls of evaluating games content exclusively automatically, as discussed in [56]. Instead, it suggests that experiments using simulation-based fitness functions should at least be repeated with different AIs. Ideally, these AI agents resemble different player types in order to ensure diverse and meaningful behaviour [57].

4.4. Concluding remarks

Diagonal walks enable an investigation of how small changes in search space are reflected in objective space. Resulting observations of course only correspond to the selected random point and directions, and cannot offer reliable insights into global problem properties.

Still, this approach offers an efficient way to (visually) inspect characteristics of a high-dimensional space. Diagonal walks are easy to generate and interpret without the need for in-depth knowledge of evolutionary algorithms. They can thus serve as a standard method for sanity checking of PCG approaches. Since the number of samples required for this type of analysis is small, it is suitable for the often expensive fitness evaluations required for PCG. Furthermore, diagonal walks and their visualisation can serve as inspiration for the definition of hypotheses about fitness landscapes that can be tested with more sophisticated methods.

5. Estimation of high-level properties

If aimed at the general global structure of a fitness landscape, the aforementioned hypotheses can usually be formulated using high-level properties such as multi-modality, i.e., the (non-)existence of multiple optima. These properties are a way of describing fitness landscapes and are often used as a basis for determining suitable algorithms for black-box functions. Here, we propose to train a classifier for such properties using ELA features as input. That is, we draw a small sample of points in the optimisation problem's search space (e.g., using a Latin hypercube design or random uniform sampling) and compute the corresponding fitness values [28,58]. Subsequently, the fitness landscape of the particular optimisation problem is characterised by means of various numerical summary statistics – called ELA features [4,9,11] – that are computed based on the sampled points.

5.1. Method description

Our method, outlined in Fig. 5, is based on previous work by [10], in which various high-level properties of problem landscapes, such as (their degree of) multi-modality and global structure, were discussed. The authors further (manually) labelled the 24 artificial test functions from the well-known Black-Box Optimization Benchmark (BBOB) [59] with the defined high-level properties.

We use this data – i.e., the levels of the high-level properties (e.g., *none*, *low*, *medium* or *high* degree of multimodality) – as outcome or class labels when training classifiers for a total of eight[2] high-level properties. As input, each classification model (one per high-level property) takes a large set of cheap but informative ELA features. For our experiments we considered all 97 features – like the coefficient of determination R^2 that was mentioned in Section 2.1 as an exemplary ELA feature – from the following

[2] As all but one BBOB problem were said to be without plateaus, we removed this property, and instead considered the funnel characteristic from [17].

nine feature sets: dispersion, level set, meta model, y-distribution, angle, information content, nearest better clustering, basic, and principal component analysis [10,15,17,60,61]. A detailed, yet compact description of the considered feature sets can be found in the most-recent survey on ELA [11].

Our experiments are based on all 24 test problems from BBOB and rely on samples of 500 points that were created by means of a latin hypercube design (on the corresponding 10-dimensional search space $[-5, 5]^{10}$). For each of the problems, we then computed the aforementioned 97 ELA features using the R-package `flacco` [11,62]. In order to capture the general characteristics of a BBOB problem and not traits that are specific to one of its instances,[3] we considered (the first) five problem instances per problem.

Once we had generated appropriately labelled data, we trained a variety of classification models (classification trees [63], random forests [64], support vector machines [65] and gradient boosting models [66]) – all of which have proven to be suitable candidates in the context of feature-based studies (see, e.g., [5,67]) – to find strong classifiers for each of the different high-level properties. To reduce noise as well as redundancy among the features and thereby improve the quality of the models, each classifier was trained using different greedy (floating forward–backward and backward–forward selection, respectively) and stochastic (evolutionary algorithms with plus-strategy, population size $\mu = 10$, and $\lambda \in \{5, 50\}$ offspring) automated feature selection strategies as presented in [5]. All models were evaluated using leave-one-function-out cross-validation. This allows a fair comparison of the models and reduces the risk of overfitting.

In the end, we obtained at least one well-performing classification model per high-level property, capable of predicting the respective attribute(s) based on a set of ELA features as input. These classifiers can be applied to any black-box function, as long as an appropriate number of samples can be computed. Despite performing well, model predictions are no guarantees and rely on several assumptions. Still, they can serve as sophisticated and computationally efficient guesses for high-level properties of black-box functions.

5.2. Purpose in context of PCG

As most search-based PCG approaches employ complex genotype-phenotype mappings as well as simulation-based fitness functions, they must usually be considered as black boxes. In order to make informed decisions about the choice of the different PCG components (representation, fitness function, optimisation algorithm), it is thus crucial to obtain some information on the high-level properties of the resulting fitness landscapes. For example, if the high-level properties indicate a highly multimodal landscape, restarts of the evolutionary algorithm should be considered.

While diagonal walks (see Section 4) provide a basic way of approaching this lack of information, they can only offer insights into small slices of the complete landscape. Depending on how heterogeneous the landscape is, the information gathered from diagonal walks cannot be generalised to the entire landscape. A data-driven method, such as the classifiers proposed in this section, can provide information on whichever area of the landscape is sampled. It can thus be used as a way to investigate hypotheses formulated based on domain knowledge or initial observations.

5.3. MarioGAN results

5.3.1. Analysis

The heatmap in Fig. 7 illustrates the predictions for six (of all eight considered) high-level properties by previously trained classifiers for all MarioGAN problems. The rows indicate the problems (m_1 to m_{28}), whereas the columns subdividing each high-level property heading correspond to the seven instances per problem (see Section 3.3).

An immediately noticeable observation is that the high-level properties seem to be relatively similar across all the different MarioGAN problems. Two properties are not even shown because they were constant across all problems: none of the problems are separable (i.e., they cannot be broken down into smaller subproblems), and they all have a very homogeneous search space. However, when looking more closely, it is noticeable that for the agent-based problems ($m_{11} - m_{28}$) the attributes of the high-level properties are mostly consistent, whereas the characteristics of the representation-based problems display partly considerable differences. For instance, problems based on the fitness measures enemyDistribution (m_1, m_2) and leniency (m_9 and m_{10}) show no differences along the different search space dimensions as well as some evidence of a global structure, while all other problems indicate a moderate degree of variable scaling without signs of a funnel or global structure. Another characteristic we observed are the differences between over- and underworld levels for the remaining representation-based problems: The overworld problems (m_3, m_5 and m_7) apparently exhibit a higher degree of multimodality as well as a stronger global to local optima contrast compared to their underground counterparts (m_4, m_6 and m_8). Summarising across all 196 problems, we seem to be dealing with problems that mostly:

(1) behave differently in the different dimensions of the search space (*variable scaling*: medium) and are non-separable (*separability*: none), i.e., different dimensions of the search space cannot be treated separately;
(2) have no obvious global trends (*funnel*: none, *global structure*: none, *global to local optima contrast*: low) that could be utilised e.g., by estimation of distribution algorithms, such as the Covariance Matrix Adaption Evolution Strategy (CMA-ES) [68];
(3) have attraction basins of varying sizes (*basin space homogeneity*: none/low);
(4) have a relatively high number of local optima (*multimodality*: high).

Based on these results, we conclude that most of these problems are likely difficult to tackle. This largely aligns with the diagonal walk results discussed in Section 4.3.

5.3.2. Interpretation

Early results on the optimisation problems indicate that a majority of these problems are indeed difficult to tackle, thus validating the estimations from the classifier. For MarioGAN problems, for instance, random search has even shown to be competitive with a wide range of evolutionary algorithms [54].

This result suggests that the genotype-phenotype mapping used for all problems should be reconsidered, as the problems, independent of fitness function and training set, have mostly similar high-level properties. However, the goal of PCG approaches is usually not to identify a single best sample of content, but several ideally diverse examples of sufficient quality. Finding a global optimum is thus not the main priority in PCG. The same

[3] Each instance is a translated, rotated and/or shifted version of its original function.

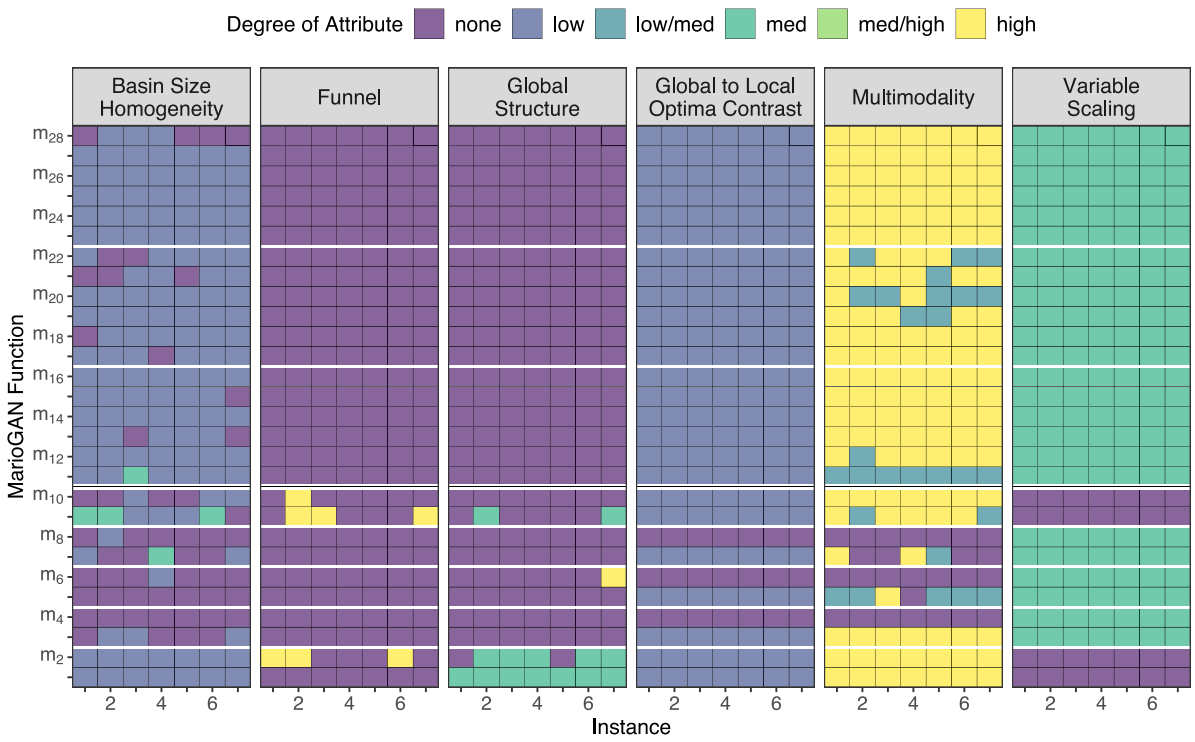


Fig. 7. Predicted high-level properties for all seven instances of each of the 28 MarioGAN problems. According to our models, *all* 196 instances are supposedly non-separable and possess highly homogeneous search spaces.

is also true for multi-modal problems, as well as optimisation in real-world contexts in general.

It should thus be investigated whether the competitive performance of random search can be explained by the abundance of suitable generated levels, which would result in very flat fitness landscapes. In this case, this rich representation should be kept and instead of applying sophisticated algorithms, simple samplers could suffice to identify suitable levels quickly. The same concerns naturally also arise for problems with similar fitness landscapes, which many problems in real-world applications might have. This should thus be a topic for further investigation.

5.4. Concluding remarks

The approach considered in this section compresses the information of all 97 landscape features into eight high-level properties. ELA features are intended for data-driven analyses, and the trained models are helpful to gain general insights based on them. In contrast to the previously discussed diagonal walks, the whole search space is considered and thus, global properties can be assigned with reasonable confidence. Obviously, the quality of the resulting classification (of the attribute values within each of these properties) strongly depends on (a) the appropriateness of the selected classification model, and (b) the representativeness of the sampled points. Moreover, it should be noted that depending on the complexity of the considered classifier(s), it can be (1) expensive to train a separate classifier (per property), and (2) hard to explain its decisions.

In consequence, the reliance on high-level properties has several issues. They describe complex abstract concepts which require a certain amount of existing familiarity with such properties in order to be interpretable by a user. They also inadvertently shape and thus limit the way in which problems are described. The accuracy of the classifier also depends, on the one hand, on the quality of the training data, which must be manually labelled with appropriate high-level properties, and, on the other hand, it is influenced by the similarity between the training data and the data of the test problem(s).

This method is thus suitable only for advanced users. For beginners, other, more intuitive methods are needed to characterise the properties of a given problem.

6. Problem similarity measures

Another way of characterising problems is to describe them by their similarity to others. Similarity can provide intuition and thus avoid reliance on complex abstract concepts such as high-level properties discussed in the previous section. In addition, similarity information can also be useful to decide whether findings from different problems are applicable in a new context.

6.1. Method description

ELA features (see Section 5) provide a way of characterising and ranking different optimisation problems numerically, and thus constitute a great basis for measuring similarity. They are, however, designed to work as input for data-driven modelling and hence should not be interpreted in isolation. We propose to use dimensionality reduction approaches to capture similarities recorded in ELA features (see Fig. 5). Here, we specifically consider t-distributed Stochastic Neighbourhood Embedding (t-SNE) [69] as a tool for finding a low-dimensional, visualisable representation of the problems — and their (dis)similarities.

As a preparatory step, ELA features are computed for all problems under analysis as described in Section 5.1. This allows a comparison between the problems under investigation. In addition, it is useful to also compute ELA features for other (well-known) problems that can serve as baselines for comparison. Ideally, these problems are well understood and diverse, so that the analysed problems can be characterised by their similarity to the baselines. Good candidates are artificial optimisation problems from benchmarks such as BBOB [59].

Next, some pre-processing of the data is required. Features with constant values across all problems need to be removed from the data. Given that many features are of different magnitude – and as we have no intention of interpreting any of them at this point – all feature vectors are normalised to ensure a more homogeneously scaled data set.

Finally, the dimensionality reduction method t-SNE [69] is applied to the (normalised) feature data. The resulting representation can be used as a basis for investigating the similarities between problems further.

6.2. Purpose in context of PCG

As discussed above, the ability to measure similarity can help in identifying problems that can serve as a suitable (i.e., comparable) test bed for the original problem. This can be helpful for the selection of suitable optimisation algorithms or even for tuning their hyper-parameters. Such a test bed is especially beneficial if it is much cheaper to evaluate than the original problem — which is often quite expensive in simulation-based PCG evaluations.

It might even be possible to identify problems that are similar enough to be able to act as a proxy for the original problem. This could for example be used as a way to identify interesting areas in the search space without having to evaluate the expensive problem.

Similarity measures are also helpful for comparisons of different versions of the same problem. The effects of modifying the representation within a PCG algorithm, for example, could be tracked this way. Conversely, it can also be used to verify the similarity of the optimisation problems encountered when applying a given PCG approach to different games. If the problems are indeed similar, we can reasonably expect that the problem landscapes exhibit similar characteristics. In this case, similarity can thus be used as a measure of the robustness of the PCG approach.

Finally, given a sufficient amount of suitable data from different search-based PCG approaches is collected, similarity measures in combination with clustering methods could be used to identify archetypes of PCG problems. This discovery would facilitate the generalisation of PCG approaches and make their outcomes more predictable. This would in turn improve their applicability in industry.

6.3. MarioGAN results

6.3.1. Analysis

For the analysis, we re-use the ELA features computed for MarioGAN (see Section 5.3). As baselines, we selected the 10-D problems from the BBOB benchmark [59] and a collection of 40 10-D instances that were generated using Shekel's foxhole function[4] [44]. In previous work, the foxhole function with its large plateaus and small spikes was identified to be potentially similar to a PCG approach based on cellular automata [44]. Based on the results from the diagonal walk analysis described in Section 4.3, foxhole functions do indeed seem like a good candidate for similarity based on their known landscape characteristics.

Fig. 8 provides a two-dimensional representation of the high-dimensional feature data (of all 356 considered problem instances) using t-SNE. The problems from MarioGAN (shown in green) form a mostly homogeneous group and do not have any spatial overlap with any of the other benchmark problems. On the other hand, the foxhole functions (yellow) form two groups, which are separated by a large cluster of BBOB problems (purple). According to this plot, the three different problem sets are not similar and we thus determine that the findings from [42] do not generalise to MarioGAN.

Analysing the MarioGAN problems in more detail, it seems that different instances from the same problem (indicated by the same problem IDs) tend to be similar. It also seems that problems $m_1 - m_{10}$ with the representation-based fitness functions concentrate on the bottom of the visualisation. They can thus be considered more similar to each other than the agent-based problems. And although this finding was expected, and could have been observed in the heatmap of high-level properties (see Section 5.3), the approach considered here is able to show this effect more clearly.

6.3.2. Interpretation

This result supports the assumption that the different instances of each problem are actually reasonably similar. It can thus be argued that the results of MarioGAN will be robust to different initialisations of the training process.

Furthermore, the Mario problems do not resemble any of the artificial problems we compared them against. In consequence, optimisation algorithms that perform well on e.g., BBOB functions will not necessarily perform well in search-based PCG approaches such as Mario level generation.

This finding reinforces the need for more systematic analyses of PCG problems. It might even require specifically suited optimisation algorithms to handle these specific fitness landscapes. A first step towards collecting suitable data for further investigations was made with the proposal of the GBEA [14], which contains PCG approaches for two different game-based applications. However, in order to allow for more general conclusions, further applications will be needed.

6.4. Concluding remarks

The characterisation of the fitness landscapes determined by dimensionality reduction of the ELA features as proposed here seems to offer a good trade-off between intuitive interpretability and the ability to gain insight into the fitness landscapes on a global level. However, it should be noted that thorough hypothesis testing is required to confirm the findings of the visual analysis.

We here suggested the usage of t-SNE, as it preserves the local proximity of similar observations instead of just the structure among dissimilar points. This is mainly achieved by minimising the Kullback–Leibler divergence between the (probability mass of the) observations from the high- and low-dimensional spaces [69]. Nonetheless, t-SNE is no silver bullet for dimensionality reduction as it also comes with drawbacks: (1) the initial low-dimensional sample created by t-SNE is stochastic and as such, its final result will differ (slightly) every time, (2) one cannot simply add a new instance to an existing plot, instead one has to re-run the entire divergence optimisation procedure, (3) it does not allow for statements regarding the proportion of explained variance (in contrast to classical dimensionality reduction approaches like principal component analysis), and (4) it is computationally much more expensive than its competitors.

7. Summary and future work

In this paper, we demonstrate how tools designed for the systematic analysis of optimisation problems can be employed to understand more about the interaction of the different components (representation, fitness function, search algorithm) of

[4] Eight problems generated with different numbers of peaks (3, 5, 7, 10, 20, 30, 40 and 50). To create five instances each, locations and widths of the peaks were sampled random uniformly from $[0, 10]^{10}$ and [0, 1], respectively.

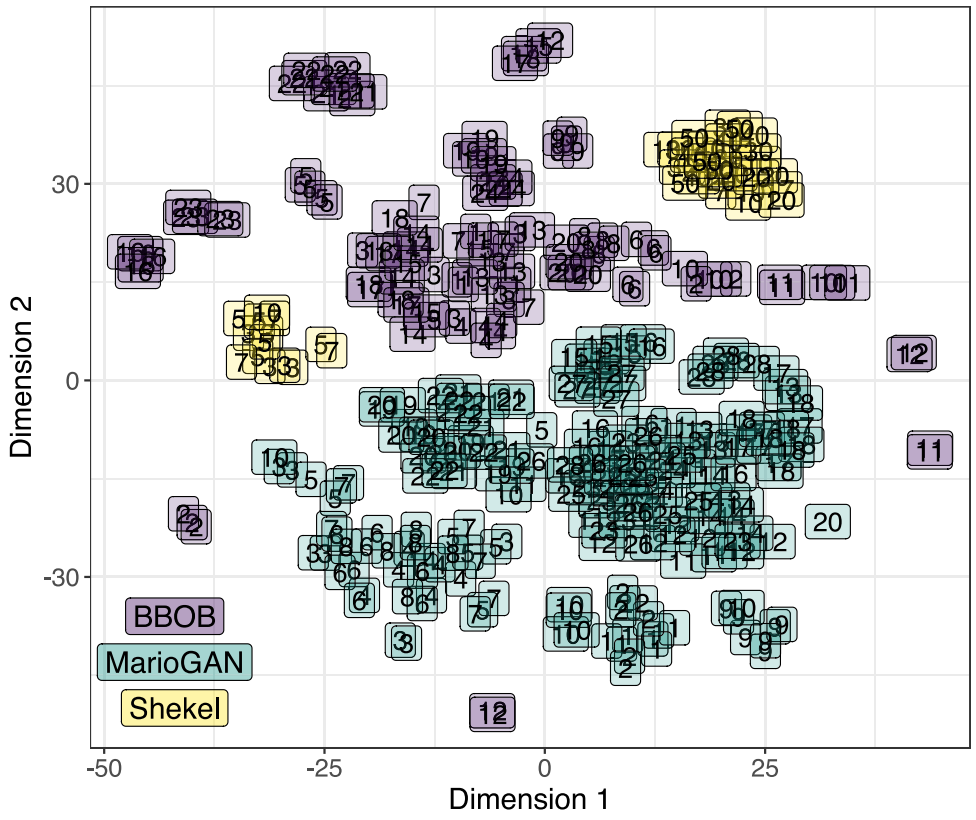


Fig. 8. Low-dimensional representation of all considered problems from BBOB (24 functions × 5 instances, shown in purple), MarioGAN (28 functions (see Table 1) × 7 instances, green) and Shekel (8 functions × 5 instances, yellow). The mapping was created by applying t-SNE to the corresponding normalised ELA feature vectors.

search-based PCG approaches. We further argue how the derived insights can be used to evaluate and improve these three approaches. This discussion is on the one hand executed on a general level, and on the other hand demonstrated with a level generator for Mario. These results are the basis for our conclusions on the suitability of the different tools we review. Within this study, we focus on analysis tools intended to identify suitable optimisation algorithms.

We first discussed diagonal walks, which are an easy way of gathering interpretable information on the fitness landscapes in specific areas of the search space. Their disadvantage is their locality and their therefore very limited perspective regarding global properties of a given optimisation problem.

Next, we investigated a machine learning-based method that is capable of estimating these high-level properties based on a suitably small number of samples of a PCG problem. Besides potential issues incurred through classification errors, the main downside of this tool is the required familiarity with the high-level properties used to characterise the problems. While this method can thus potentially provide very detailed insights, it is particularly suitable for advanced users. Moreover, feature-based approaches also highly depend on the quality of the sampled points, which includes their amount and distribution, as they are fundamental for the feature computation — which in turn forms the basis for various further investigations.

As a compromise between the very simple diagonal walks and the rather challenging identification of high-level properties, we studied another data-driven – yet more intuitively interpretable – method capable of recording an optimisation problem's global properties. We proposed using the similarity to archetypal baseline functions as a way to gain insights into black-box optimisation problems and employed the distance in a learned latent space as distance measure.

In consequence, the appropriate analysis tool thus needs to be chosen based on the given circumstances and the user's existing expertise. However, as demonstrated in this paper, a simple experiment with diagonal walks can already achieve interesting insights and serve as a baseline and sanity check for any work on search-based procedural content generation. Unfortunately, and despite the simplicity of the diagonal walks, even such approaches are rarely performed in related works.

We tested all the tools on an example application of MarioGAN, a content generator for Mario levels based on generative adversarial networks. Our main finding is that it is likely that the problems will prove difficult to optimise for standard evolutionary approaches. We plan to investigate this further by employing different analysis tools for additional insights from different perspectives in the future. Further, we will validate our assumption by comparing multiple optimisation algorithms on these problems.

Within this paper, we focused on a specific type of analysis and only discussed three selected analysis tools. This was necessary to allow for sufficient depth in our study and to demonstrate the suitability of the tools on a specific PCG-based use case. However, there exists a wide range of other appropriate tools for systematic analysis of PCG problems with varying degrees of complexity. Examples include the analysis of noise in fitness functions, as well as plots of the Empirical Cumulative Distribution Function (ECDF) for the visualisation of optimisation performance over time [12,13].

In the future, we plan to employ such additional tools to gather more information on PCG problems, starting with the ones that are already part of the game-based benchmark GBEA [14]. Combining observations that were produced with different tools promises further insights into these complex problems.

Further down the line, we hope to encourage a best practice for making data from different systematic analyses of PCG problems publicly available. Such a dataset would be an invaluable resource in order to conduct research on the robustness and generalisability of search-based PCG approaches. This would allow for a much wider adaptation of such methods in the games industry as the expected performance could then be estimated before implementation. Decision makers in a practical context would thus be able to operate with more information and choose appropriate tools.

CRediT authorship contribution statement

Vanessa Volz: Conceptualization, Methodology, Software, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Boris Naujoks:** Methodology, Writing – original draft, Writing – review & editing, Validation, Resources. **Pascal Kerschke:** Methodology, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Tea Tušar:** Conceptualization, Methodology, Software, Investigation, Writing – original draft, Writing – review & editing, Visualization, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

Tea Tušar acknowledges financial support from the Slovenian Research Agency (projects No. Z2-8177 and N2-0254, and program No. P2-0209). Pascal Kerschke acknowledges support by the *European Research Center for Information Systems (ERCIS)*.

Appendix. Function definitions

We formally define here the functions from Section 3.2 that are implemented in the MarioGAN benchmark used for the analysis in the paper. Note that they are all mapped to a minimisation problem via transformation and scaled to fit into a value range of [0, 1] in order to better fit into the benchmarking framework which requires strict limits. In many cases, reaching one or both limits of the fitness value range is unrealistic for any meaningful game level.

enemyDistribution [38]: $1 - \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)}}{m_{sd}}$, where N is the number of enemies in the level, x_i is the x-axis coordinate of the ith enemy, m_{sd} is the largest standard deviation possible given the width of the level and $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$.

positionDistribution [38]: $1 - \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \mu)}}{m_{sd}}$, where N is the number of tiles in the level that a player can stand on, y_i is the y-axis coordinate of the ith such tile, m_{sd} is the largest standard deviation possible given the height of the level and $\mu = \frac{1}{N}\sum_{i=1}^{N} y_i$.

decorationFrequency [38]: $1 - \frac{n_{pt}}{n_{tot}}$, where n_{pt} is the number of *pretty tiles* in the level and n_{tot} the total number of tiles. Pretty tiles are *pretty tiles* := {Tube, Enemy, Destructible Block, Question Mark Block, or Bullet Bill Shooter Column}

negativeSpace [51]: $1 - \frac{n_{st}}{n_{tot}}$, where n_{st} is the number of tiles in the level that the player can stand on and n_{tot} the total number of tiles.

leniency [52]: $\frac{1}{2}\left(\frac{v}{n_{tot}} + 1\right)$, where $v = \sum_{i\in P} n_i - \sum_{i\in N} n_i - \frac{n_{gaps}}{2} - \overline{d_{gaps}}$ where P contains question blocks with power ups, and N contains bullet bill shooter stations, piranha plant tubes, and all enemy types and n_i is the number of tiles of type i. Further, n_{gaps} is the number of gaps the player has to jump over and $\overline{d_{gaps}}$ the average length of all gaps in the level.

The maximum value for $\frac{v}{n_{tot}}$ is 1 in the unrealistic case that $\sum_{i\in P} n_i = n_{tot}$ and $\sum_{i\in N} n_i = 0$ and $n_{gaps} = 0$, so all tiles in the level are power ups. Conversely, the minimum value for v is -1, if $\sum_{i\in N} n_i = n_{tot}$ and $\sum_{i\in P} n_i = 0$, so all tiles are of a type contained in N. Gaps can be neglected here because each gap would reduce the computed value by *height*. Therefore, the resulting value of the above formula is between 0 and 1.

basicFitness [46]: $\frac{v+0.04}{1.26}$, where $v = (d_{level} - t_{level} + n_{coins} + I_{won} * 5000)/5000$, d_{level} is the length of the level passed, t_{level} is the time spent on the level, n_{coins} is the number of gained coins and $I_{won} = 1$ if the level was completed and 0 otherwise. The constants used for normalisation have been determined via experimentation.

airTime [35]: $\begin{cases} \frac{t_g}{t_{tot}}, & \text{if won} \\ 1, & \text{otherwise} \end{cases}$, where t_g is the number of game ticks spent on the ground and t_{tot} the total number of game ticks played.

timeTaken [35]: $\begin{cases} 1 - \frac{t_{tot}}{t_{max}}, & \text{if won} \\ 1, & \text{otherwise} \end{cases}$, where t_{tot} is the time spent on the level and t_{max} is the total allotted time to finish the level.

References

[1] J. Togelius, N. Shaker, The search-based approach, in: N. Shaker, J. Togelius, M.J. Nelson (Eds.), Procedural Content Generation in Games: A Textbook and an Overview of Current Research, Springer, 2016, pp. 17–30, http://dx.doi.org/10.1007/978-3-319-42716-4.

[2] N. Shaker, G. Smith, G.N. Yannakakis, Evaluating content generators, in: N. Shaker, J. Togelius, M.J. Nelson (Eds.), Procedural Content Generation in Games: A Textbook and an Overview of Current Research, Springer, 2016, pp. 215–224, http://dx.doi.org/10.1007/978-3-319-42716-4_2.

[3] P. Kerschke, H.H. Hoos, F. Neumann, H. Trautmann, Automated algorithm selection: Survey and perspectives, Evol. Comput. 27 (1) (2019) 3–45, http://dx.doi.org/10.1162/evco_a_00242.

[4] M.A. Muñoz Acosta, Y. Sun, M. Kirley, S.K. Halgamuge, Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges, Inf. Sci. (JIS) 317 (2015) 224–245, http://dx.doi.org/10.1016/j.ins.2015.05.010.

[5] P. Kerschke, H. Trautmann, Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning, Evol. Comput. 27 (1) (2019) 99–127, http://dx.doi.org/10.1162/evco_a_00236.

[6] B. Bischl, O. Mersmann, H. Trautmann, M. Preuss, Algorithm selection based on exploratory landscape analysis and cost-sensitive learning, in: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation (GECCO), ACM, 2012, pp. 313–320, http://dx.doi.org/10.1145/2330163.2330209.

[7] R.P. Prager, H. Trautmann, H. Wang, T.H.W. Bäck, P. Kerschke, Per-instance configuration of the modularized CMA-ES by means of classifier chains and exploratory landscape analysis, in: Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2020, pp. 996–1003, http://dx.doi.org/10.1109/SSCI47803.2020.9308510.

[8] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82, http://dx.doi.org/10.1109/4235.585893.

[9] K.M. Malan, A.P. Engelbrecht, A survey of techniques for characterising fitness landscapes and some possible ways forward, Inf. Sci. (JIS) 241 (2013) 148–163, http://dx.doi.org/10.1016/j.ins.2013.04.015.

[10] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, G. Rudolph, Exploratory landscape analysis, in: Genetic and Evolutionary Computation Conference, GECCO, ACM, 2011, pp. 829–836, http://dx.doi.org/10.1145/2001576.2001690.

[11] P. Kerschke, H. Trautmann, Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the R-package flacco, in: N. Bauer, K. Ickstadt, K. Lübke, G. Szepannek, H. Trautmann, M. Vichi (Eds.), Applications in Statistical Computing, Springer, 2019, pp. 93–123, http://dx.doi.org/10.1007/978-3-030-25147-5_7.

[12] N. Hansen, A. Auger, D. Brockhoff, T. Tušar, Anytime performance assessment in blackbox optimization benchmarking, IEEE Trans. Evol. Comput. 26 (6) (2022) 1293–1305, http://dx.doi.org/10.1109/TEVC.2022.3210897.

[13] T. Bartz-Beielstein, C. Doerr, D. van den Berg, J. Bossek, S. Chandrasekaran, T. Eftimov, A. Fischbach, P. Kerschke, W. La Cava, M. López-Ibáñez, K.M. Malan, J.H. Moore, B. Naujoks, P. Orzechowski, V. Volz, M. Wagner, T. Weise, Benchmarking in optimization: Best practice and open issues, 2020, arXiv preprint arXiv:2007.03488.

[14] V. Volz, B. Naujoks, P. Kerschke, T. Tušar, Single- and multi-objective game-benchmark for evolutionary algorithms, in: Genetic and Evolutionary Computation Conference, GECCO, ACM, 2019, pp. 647–655, http://dx.doi.org/10.1145/3321707.3321805.

[15] M.A. Muñoz Acosta, M. Kirley, S.K. Halgamuge, Exploratory landscape analysis of continuous space optimization problems using information content, IEEE Trans. Evol. Comput. 19 (1) (2015) 74–87, http://dx.doi.org/10.1109/TEVC.2014.2302006.

[16] K.M. Malan, A.P. Engelbrecht, Quantifying ruggedness of continuous landscapes using entropy, in: IEEE Congress on Evolutionary Computation, CEC, IEEE, 2009, pp. 1440–1447, http://dx.doi.org/10.1109/CEC.2009.4983112.

[17] P. Kerschke, M. Preuss, S. Wessing, H. Trautmann, Detecting funnel structures by means of exploratory landscape analysis, in: Genetic and Evolutionary Computation Conference, GECCO, ACM, 2015, pp. 265–272, http://dx.doi.org/10.1145/2739480.2754642.

[18] U. Škvorc, T. Eftimov, P. Korošec, Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis, Appl. Soft Comput. (ASOC) 90 (2020) 106138, http://dx.doi.org/10.1016/j.asoc.2020.106138.

[19] M.A. Muñoz Acosta, K.A. Smith-Miles, Generating new space-filling test instances for continuous black-box optimization, Evol. Comput. (ECJ) 28 (3) (2020) 379–404, http://dx.doi.org/10.1162/evco_a_00262.

[20] R.D. Lang, A.P. Engelbrecht, An exploratory landscape analysis-based benchmark suite, Algorithms 14 (3) (2021) 78, http://dx.doi.org/10.3390/a14030078.

[21] L. Schneider, L. Schäpermeier, R.P. Prager, B. Bischl, H. Trautmann, P. Kerschke, HPO × ELA: Investigating hyperparameter optimization landscapes by means of exploratory landscape analysis, in: International Conference on Parallel Problem Solving from Nature (PPSN), Springer, 2022, pp. 575–589, http://dx.doi.org/10.1007/978-3-031-14714-2_40.

[22] A.P. Engelbrecht, P. Bosman, K.M. Malan, The influence of fitness landscape characteristics on particle swarm optimisers, Nat. Comput. (2021) 1–11, http://dx.doi.org/10.1007/s11047-020-09835-x.

[23] Z. Pitra, J. Repický, M. Holeňa, Landscape analysis of Gaussian process surrogates for the covariance matrix adaptation evolution strategy, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), ACM, 2019, pp. 691–699, http://dx.doi.org/10.1145/3321707.3321861.

[24] A. Jankovic, C. Doerr, Landscape-aware fixed-budget performance regression and algorithm selection for modular CMA-ES variants, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference (GECCO), ACM, 2020, pp. 841–849, http://dx.doi.org/10.1145/3377930.3390183.

[25] R.P. Prager, M.V. Seiler, H. Trautmann, P. Kerschke, Automated algorithm selection in single-objective continuous optimization: A comparative study of deep learning and landscape analysis methods, in: International Conference on Parallel Problem Solving from Nature (PPSN), Springer, 2022, pp. 3–17, http://dx.doi.org/10.1007/978-3-031-14714-2_1.

[26] N. Belkhir, J. Dréo, P. Savéant, M. Schoenauer, Feature based algorithm configuration: A case study with differential evolution, in: J. Handl, E. Hart, P.R. Lewis, M. López-Ibáñez, G. Ochoa, B. Paechter (Eds.), Proceedings of the 14th International Conference on Parallel Problem Solving from Nature (PPSN XIV), in: Lecture Notes in Computer Science (LNCS), vol. 9921, Springer, 2016, pp. 156–166, http://dx.doi.org/10.1007/978-3-319-45823-6_15.

[27] N. Belkhir, J. Dréo, P. Savéant, M. Schoenauer, Per instance algorithm configuration of CMA-ES with limited budget, in: Genetic and Evolutionary Computation Conference, GECCO, ACM, 2017, pp. 681–688, http://dx.doi.org/10.1145/3071178.3071343.

[28] Q. Renau, J. Dréo, C. Doerr, B. Doerr, Expressiveness and robustness of landscape features, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) Companion, ACM, 2019, pp. 2048–2051, http://dx.doi.org/10.1145/3319619.3326913.

[29] B. Derbel, A. Liefooghe, S. Verel, H. Aguirre, K. Tanaka, New features for continuous exploratory landscape analysis based on the SOO tree, in: Foundations of Genetic Algorithms, FOGA, ACM, 2019, pp. 72–86, http://dx.doi.org/10.1145/3299904.3340308.

[30] J. Togelius, G.N. Yannakakis, K.O. Stanley, C. Browne, Search-based procedural content generation: A taxonomy and survery, IEEE Trans. Comput. Intell. AI Games 3 (3) (2011) 172–186, http://dx.doi.org/10.1109/TCIAIG.2011.2148116.

[31] G.N. Yannakakis, J. Togelius, A panorama of artificial and computational intelligence in games, IEEE Trans. Comput. Intell. AI Games 7 (4) (2015) 317–335, http://dx.doi.org/10.1109/TCIAIG.2014.2339221.

[32] D. Ashlock, S. Risi, J. Togelius, Representations for search-based methods, in: N. Shaker, J. Togelius, M.J. Nelson (Eds.), Procedural Content Generation in Games: A Textbook and an Overview of Current Research, Springer, 2016, pp. 159–179, http://dx.doi.org/10.1007/978-3-319-42716-4_9.

[33] P. Spronck, E. André, M. Cook, M. Preuß, Artificial and computational intelligence in games: AI-driven game design (dagstuhl seminar 17471), Dagstuhl Rep. 7 (11) (2018) 86–129, http://dx.doi.org/10.4230/DagRep.7.11.86.

[34] G.N. Yannakakis, J. Togelius, Experience-driven procedural content generation, IEEE Trans. Affect. Comput. 2 (3) (2011) 147–161, http://dx.doi.org/10.1109/T-AFFC.2011.6.

[35] V. Volz, J. Schrum, J. Liu, S.M. Lucas, A. Smith, S. Risi, Evolving Mario levels in the latent space of a deep convolutional generative adversarial network, in: Genetic and Evolutionary Computation Conference, GECCO, ACM, 2018, pp. 221–228, http://dx.doi.org/10.1145/3205455.3205517.

[36] M. González-Duque, R.B. Palm, S. ren Hauberg, S. Risi, Mario plays on a manifold: Generating functional content in latent space through differential geometry, in: 2022 IEEE Conference on Games (CoG), 2022, pp. 385–392, http://dx.doi.org/10.1109/CoG51982.2022.9893612.

[37] C.B. Browne, Automatic Generation and Evaluation of Recombination Games (Ph.D. thesis), Faculty of Information Technology, Queensland University of Technology, 2008.

[38] A. Summerville, J.R.H. Mariño, S. Snodgrass, S. Ontañón, L. Lelis, Understanding Mario: An evaluation of design metrics for platformers, in: Foundations of Digital Games, FDG, ACM, 2017, pp. 1–10, http://dx.doi.org/10.1145/3102071.3102080.

[39] D. Ashlock, C. McGuinness, Landscape automata for search based procedural content generation, in: Computational Intelligence in Games, CIG, IEEE, 2013, pp. 9–16, http://dx.doi.org/10.1109/CIG.2013.6633619.

[40] M. Preuss, Multimodal Optimization By Means of Evolutionary Algorithms, Springer, 2015, http://dx.doi.org/10.1007/978-3-319-07407-8.

[41] A. Bhatt, S. Lee, F. de Mesentier Silva, C.W. Watson, J. Togelius, A.K. Hoover, Exploring the hearthstone deck space, in: Foundations of Digital Games, FDG, ACM, 2018, pp. 18:1–18:10, http://dx.doi.org/10.1145/3235765.3235791.

[42] D. Ashlock, S. McNicholas, Fitness landscapes of evolved apoptotic cellular automata, IEEE Trans. Evol. Comput. 17 (2) (2013) 198–212, http://dx.doi.org/10.1109/TEVC.2013.2243454.

[43] E. Pitzer, M. Affenzeller, A comprehensive survey on fitness landscape analysis, in: J. Fodor, R. Klempous, C.P. Suárez Araujo (Eds.), Recent Advances in Intelligent Engineering Systems, in: Studies in Computational Intelligence, Springer, 2012, pp. 161–191, http://dx.doi.org/10.1007/978-3-642-23229-9_8.

[44] J. Shekel, Test functions for multimodal search techniques, in: Fifth Annual Princeton Conference on Information Science and Systems, 1971, pp. 354–359.

[45] S. Omidshafiei, K. Tuyls, W.M. Czarnecki, F.C. Santos, M. Rowland, J. Connor, D. Hennes, P. Muller, J. Pérolat, B.D. Vylder, A. Gruslys, R. Munos, Navigating the landscape of multiplayer games, Nature Commun. 11 (5603) (2020) http://dx.doi.org/10.1038/s41467-020-19244-4.

[46] J. Togelius, N. Shaker, S. Karakovskiy, G.N. Yannakakis, The Mario AI championship 2009–2012, AI Mag. 34 (3) (2013) 89–92, http://dx.doi.org/10.1609/aimag.v34i3.2492.

[47] B. Horn, S. Dahlskog, N. Shaker, G. Smith, J. Togelius, A comparative evaluation of procedural level generators in the Mario AI framework, in: Foundations of Digital Games, FDG, Society for the Advancement of the Science of Digital Games, 2014, p. 8.

[48] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Neural Information Processing Systems 27, NIPS, Curran Associates, 2014, pp. 2672–2680.

[49] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: International Conference on Machine Learning, ICML, in: Proceedings of Machine Learning Research, vol. 70, PMLR, 2017, pp. 214–223.
[50] A.J. Summerville, S. Snodgrass, M. Mateas, S.O. Villar, The VGLC: The video game level corpus, 2016, arXiv preprint arXiv:1606.07487.
[51] A. Canossa, G. Smith, Towards a procedural evaluation technique: Metrics for level design, in: Foundations of Digital Games, FDG, Society for the Advancement of the Science of Digital Games, 2015, p. 7.
[52] N. Shaker, M. Nicolau, G.N. Yannakakis, J. Togelius, M. O'Neill, Evolving levels for Super Mario Bros using grammatical evolution, in: Computational Intelligence and Games, CIG, IEEE, 2012, pp. 304–311, http://dx.doi.org/10.1109/CIG.2012.6374170.
[53] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, D. Brockhoff, COCO: a platform for comparing continuous optimizers in a black-box setting, Optim. Methods Softw. 36 (1) (2021) 114–144, http://dx.doi.org/10.1080/10556788.2020.1808977.
[54] V. Volz, Uncertainty handling in surrogate assisted optimisation of games, KI – Künstliche Intell. 34 (1) (2020) 95–99, http://dx.doi.org/10.1007/s13218-019-00613-1.
[55] F. Rothlauf, Representations for genetic and evolutionary algorithms, in: Representations for Genetic and Evolutionary Algorithms, Vol. 104, Springer, 2006, pp. 73–96.
[56] V. Volz, B. Naujoks, On the effects of simulating human decisions in game analysis, in: IEEE Conference on Games (CoG), IEEE, 2019, pp. 1–8, http://dx.doi.org/10.1109/CIG.2019.8848071.
[57] A. Liapis, C. Holmgård, G.N. Yannakakis, J. Togelius, Procedural personas as critics for dungeon generation, in: A.M. Mora, G. Squillero (Eds.), Applications of Evolutionary Computation, Springer, 2015, pp. 331–343, http://dx.doi.org/10.1007/978-3-319-16549-3_27.
[58] P. Kerschke, M. Preuss, S. Wessing, H. Trautmann, Low-budget exploratory landscape analysis on multiple peaks models, in: Proceedings of the 18th Annual Conference on Genetic and Evolutionary Computation (GECCO), ACM, 2016, pp. 229–236, http://dx.doi.org/10.1145/2908812.2908845.
[59] N. Hansen, S. Finck, R. Ros, A. Auger, Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions, Research Report RR-6829, INRIA, Paris, France, 2009.
[60] M. Lunacek, L.D. Whitley, The dispersion metric and the CMA evolution strategy, in: Genetic and Evolutionary Computation Conference, GECCO, ACM, 2006, pp. 477–484, http://dx.doi.org/10.1145/1143997.1144085.
[61] P. Kerschke, M. Preuss, C.I. Hernández Castellanos, O. Schütze, J.-Q. Sun, C. Grimme, G. Rudolph, B. Bischl, H. Trautmann, Cell mapping techniques for exploratory landscape analysis, in: EVOLVE – a Bridge Between Probability, Set Oriented Numerics, and Evolutionary Computation V, Springer, 2014, pp. 115–131, http://dx.doi.org/10.1007/978-3-319-07494-8_9.
[62] C. Hanster, P. Kerschke, Flaccogui: Exploratory landscape analysis for everyone, in: Genetic and Evolutionary Computation Conference, GECCO, ACM, 2017, pp. 1215–1222, http://dx.doi.org/10.1145/3067695.3082477.
[63] T. Therneau, B. Atkinson, B. Ripley, `Rpart`: Recursive partitioning and regression trees, 2017, R-package version 4.1-11.
[64] A. Liaw, M. Wiener, Classification and regression by RandomForest, R News 2 (3) (2002) 18–22.
[65] A. Karatzoglou, A. Smola, K. Hornik, A. Zeileis, Kernlab – An S4 package for kernel methods in R, J. Statist. Softw. (JSS) 11 (9) (2004) 1–20, URL http://www.jstatsoft.org/v11/i09/.
[66] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, `Xgboost`: Extreme gradient boosting, 2017, R-package version 0.6-4.
[67] P. Kerschke, L. Kotthoff, J. Bossek, H.H. Hoos, H. Trautmann, Leveraging TSP solver complementarity through machine learning, Evol. Comput. J. (ECJ) 26 (2018) 597–620, http://dx.doi.org/10.1162/evco_a_00215.
[68] N. Hansen, The CMA evolution strategy: A comparing review, Stud. Fuzz. Soft Comput. 192 (2006) 75–102, http://dx.doi.org/10.1007/3-540-32494-1_4.
[69] L.v.d. Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (2008) 2579–2605.