COMPLEX SYSTEM MODELING AND SIMULATION
ISSN 2096-9929 04/06 pp 57−74
Volume 6, Number 1, March 2026
DOI: 10.23919/CSMS.2025.0034

MOPCGRL: Multi-Objective Procedural Content Generation via Reinforcement Learning

Yi Yuan, Qingquan Zhang, Bo Yuan*, Matthew Barthet, Ahmed Khalifa, Georgios N. Yannakakis, Huanhuan Chen, and Jialin Liu

Abstract: Online content generation enables automatic and adaptive creation of diverse and playable game content for maximizing player experience or testing Artificial Intelligence (AI) algorithms. Multiple diversity metrics have been formulated on different content facets in the literature, while some of them conflict with one another. Existing work addresses this multi-dimensional diversity nature by converting those metrics into one term that is further used to direct the training of content generators. However, each generator is trained to meet the preference specified by the weights and fails to fully interpret the relationships among these metrics or provide different trade-offs. This paper proposes a multi-objective procedural content generation via reinforcement learning to train a set of generators that create diverse game content in an online manner while balancing the trade-off between multiple diversity metrics with playability as a constraint. Our framework is compared with state-of-the-art approaches on the commonly used Mario-AI benchmark. Results show that our framework is capable of increasing the diversity of the generator distribution while accelerating the convergence during the early stages of model training. Our approach enables researchers, designers, and practitioners to gain a better understanding of the relationship among conflicting diversity metrics, allowing them to generate content more efficiently and accurately tailored to specific needs.

Key words: multi-objective evolutionary learning; procedural content generation; content diversity; video games

1 Introduction

Video games can simulate various scenarios in the real world through their complex environments and dynamic feedback mechanisms, and produce diverse datasets in the interaction process with humans or Artificial Intelligence (AI) agents[1, 2]. They also provide a certain degree of controllability for researchers to adjust scenarios according to their needs, which makes it easier to reproduce experiments. Therefore, they are ideal testbeds for AI research[3–8]. However, testing such algorithms usually requires a large amount of interactive content, which is typically labor-intensive and time consuming to manually design. By applying Procedural Content Generation (PCG)[9–13], diverse testing content can be efficiently and automatically generated from scratch or with limited human input.

• Yi Yuan, Qingquan Zhang, and Bo Yuan are with Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China. E-mail: 12332457@mail.sustech.edu.cn; 12431267@mail.sustech.edu.cn; yuanb@sustech.edu.cn.
• Matthew Barthet, Ahmed Khalifa, and Georgios N. Yannakakis are with Institute of Digital Games, University of Malta, Msida 2080, Malta. E-mail: matthew.barthet@um.edu.mt; ahmed@akhalifa.com; georgios.yannakakis@um.edu.mt.
• Huanhuan Chen is with School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China. E-mail: hchen@ustc.edu.cn.
• Jialin Liu is with School of Data Science, Lingnan University, Hong Kong 999077, China. E-mail: jialin.liu@ln.edu.hk.
∗ To whom correspondence should be addressed.
※ This article was recommended by Executive Editor-in-Chief Ling Wang.
Manuscript received: 2025-06-05; revised: 2025-08-30; accepted: 2025-09-23

There is a growing demand for diversity of generated content in the field of PCG nowadays[14], since it is an important evaluation criterion of interactive content. From the perspective of AI algorithm evaluation, Risi and Togelius[15] pointed out that Machine Learning (ML) researchers are turning to PCG methods to address challenges such as transferring trained models from simulated scenarios to the real world. Since ML algorithms are prone to overfitting, the obtained models are constrained by both specific training tasks and initial parameter configurations used for optimization. Leveraging PCG approaches for generating more diverse datasets or training scenarios can improve the generalization ability of ML-based systems[16,17]. From a gaming perspective, content generators that are capable of generating diverse content allow developers to optimize game creation based on user data, providing players with a personalized gaming experience, thus extending the life cycle of video games[1].

According to Li et al.[14], diversity of game content can be measured with various metrics on different content facets, such as rules, levels, gameplay, and game pacing. Distance-based metrics can be a straightforward method for evaluating the diversity between scenarios, thus being applied widely in 2D content generation[18–24] and even voxel-based 3D maze generation[25]. Some research calculates the similarity between content like symmetry[26] and tile-pattern distribution[22, 27, 28] when distance cannot present diversity precisely. The aforementioned studies all formulate their objective functions with either one diversity metric or a linear combination of metrics. However, designing a single objective which is able to capture the complex standard of modern testbeds is a significant challenge. It is essential to consider optimizing multiple aspects simultaneously when designing objectives within a single game facet[29, 30] as well as across different facets[31], since the metrics for evaluating the diversity of these facets can vary significantly.

Characterized by their multi-dimensional nature[14], many diversity metrics may conflict with one another[31]. Applying a multi-objective approach to content generation can illuminate the relationship among metrics in a more intuitive way, and generate scenarios that better match specific preferences[31]. However, the generation method used in the work of Ref. [31] is offline, thus, its generation process is independent of player behavior or experience. It is not able to generate subsequent content based on the current game state. Online generators are capable of creating content whilst adapting to the player's behavior and experience[21] in real time, allowing them to handle dynamically changing and interactive environments better than offline methods. Traditional ML methods are able to realize online generation, but may struggle with training-data collection and game rule design[32–36]. Deep Reinforcement Learning (RL) methods are particularly well-suited for online content generation, since once training is complete, they can generate content adaptively and even in real time. Also, it is independent of training data, and has strong interactivity with the environment[37]. Recent works show great interest in applying RL algorithms to online content generation[20–23, 38], but few consider the conflicts of their chosen diversity metrics. Moreover, identifying such conflicts requires repeatedly training generators[23].

Inspired by the recently proposed multi-objective offline content generation framework[31] and previous studies on online content generation[22, 23], we integrate a multi-objective approach with PCG via RL. In contrast to the weighted-sum approaches, we model two conflicting diversity metrics as separate objectives. By employing a multi-objective optimizer, we are able to explore the trade-offs between them within a single run and obtain a Pareto front that directly reveals the underlying conflicts. This formulation offers a more interpretable way to understand and control diversity than heuristic combinations via weighted sums, and does not reply on the assumption that user preference is known and can be explicitly modeled by weights. In addition, we consider setting playability as a constraint instead of an objective function as done in previous offline work[31]. Unplayable contents generated offline can be discarded or repaired before gameplay. In contrast, online generation occurs in real-time during gameplay, thus requires high playability to maintain content continuity. We set a hard constraint on playability to ensure the generators provide high-quality playable content.

This paper aims to explore an approach to train diversified online content generators while optimizing multiple objectives, and support AI testing with a large amount of interactive content that better captures the relationship between game diversity metrics. The contributions of this paper are as follows.

- We propose a multi-objective optimization framework with constraints, providing a set of game

content generators capable of balancing different diversity metrics and generating content in an online manner, while achieving decent playability. To the best of our knowledge, this is the first time a multi-objective RL approach to PCG has been applied.

- This framework is able to generate content that matches its objectives effectively, while achieving better performance with a lower computational budget compared to state-of-the-art online generation methods.
- We design a warm-up training strategy for initialization to enhance population diversity and reduce training costs. It demonstrates better diversity and convergence compared to other initialization strategies that require more training resources.

The structure of this paper is as follows. Section 2 briefly introduces related works of online and multi-objective generation in the field of PCG. Section 3 details our proposed framework, including the applied algorithms and diversity metrics. Section 4 compares our approach to state-of-the-art online generation methods, and verifies the effectiveness of the initialization strategy through an ablation study. Section 5 concludes our work and discusses its potential and weaknesses[§].

2 Background

This section first reviews existing research on online content generation and its methods to optimize multiple metrics. Then, we examine what diversity metrics have been applied in multi-objective PCG.

2.1 Online diverse content generation

In previous studies of online content generation[32–36], researchers tend to use search-based or traditional ML-based methods, with much of their focus centered on how the difficulty level affects player experience. Shaker et al.[32] performed an exhaustive search through level feature parameters, which are then combined with observed gameplay features and processed by Multi-Layer Perceptron (MLP) models to predict emotions and generate content accordingly. Jennings-Teats et al.[33] trained a difficulty model along with a player skill model, enabling the game to dynamically select level segments based on the current performance of the player. Stammer et al.[34] defined level templates, play styles, and difficulty adjustment rules, producing adaptive content by modifying the probabilities of tile occurrence. Shi and Chen[35] defined constructive primitives (quality yet controllable segments, short for CP) to control level geometry and properties. In their following work[36], they proposed a Dynamic Difficulty Adjustment (DDA) algorithm for online CP generation to align level difficulty with player performance. The difficulty of the next generated CP is only affected by the last played CP. They formulated this process as a Markov Decision Process (MDP)[39] and aimed to maximize the player's expected long-term performance, which is very similar to the later-developed RL-based online generation methods. However, in contrast with RL, it still relies heavily on high-quality training data and detailed handcrafted rules, just like other traditional online PCG studies above.

RL-based online generation requires minimal domain knowledge and training data while being suitable for interactive tasks. Experience-Driven PCG via Reinforcement Learning (EDRL) framework[21] is a typical example. It can generate personalized content using experience-based reward functions (modeling fun and historical deviation), rather than level features alone[21]. The real-time endless generation approach relies on the cooperation of a pre-trained Generative Adversarial Network (GAN)[40] and an RL agent. The former produces level segments, while the latter selects the most fitting segment to append to the end of the current level. Wang and Liu[38] used the current level segment and musical features as inputs to realize online generation of levels with varying difficulty that align with corresponding features. Wang et al.[22] focused on two game facets, defined as level and gameplay, to formulate the concept of fun as two metrics in order to improve intra-scenario diversity. Reference [23] proposes a negatively correlated ensemble RL approach to enhance inter-scenario diversity. Notably, a Pareto front can be observed in Ref. [23] by tuning the regularization coefficient, demonstrating the conflicts between different metrics after repeated training. However, in the aforementioned studies, the design of the reward function for the experience model is limited to a linear combination of different diversity metrics. Only a single model corresponding to a specific weighted combination of objectives can be obtained during one run. To meet specific requirements in AI testing, extensive effort in tuning the parameters may be needed.

In order to enhance content diversity and demonstrate the conflicting relationship among diversity metrics, multi-objective optimization is

[§] Code and results of this paper are available at https://github.com/SUSTechGameAI/MOPCGRL.

particularly suitable for this scenario. To the best of our knowledge, no work has applied multi-objective methods for online content generation. Our approach aims to provide a diversified range of interactive content through trade-off solutions, which enables content designers to more intuitively select the generators that best fit their needs and saves training time.

2.2 Enhancing content diversity with multi-objective optimization

Multi-Objective Evolutionary Algorithms (MOEAs)[41–45] demonstrate trade-offs among conflicting objectives through a set of non-dominated solutions known as the Pareto front[46]. Due to the multi-dimensional nature of diversity metrics[14], MOEAs are highly suitable for enhancing content diversity.

Togelius et al.[30, 47, 48] first applied MOEA to PCG and later extended their work. They used various distance metrics as fitness functions to evaluate subjective map qualities of StarCraft† like aesthetics, fairness, and interestingness, revealing partial conflicts between objectives and confirming the effectiveness of the generated maps through a user study. Lara-Cabrera et al.[49] characterized interestingness through game balance and dynamism using fuzzy rules, developing a self-adaptive evolutionary algorithm to optimize these objectives. They further optimized aesthetic preferences via graph-theoretic feature sets[50]. Loiacono et al.[29, 51] designed fitness functions for racing and first-person shooting games to maximize diversity. Ma et al.[52] proposed an angle-based pruning-power indicator guided evolutionary algorithm. They validated the effectiveness of the algorithm on a series of benchmark problems and content generation tasks for a real-time strategy game based on the principles of fairness, playability, strategy, and interestingness. Some of the mentioned studies further validate the effectiveness of their multi-objective approaches by evaluating the diversity of solution sets with indicators like HyperVolume (HV)[53] and Inverted Generational Distance (IGD)[54]. Zhang et al.[55] directly employed spatial diversity as one of the objective functions to maximize Sokoban‡ level generator performance.

The studies mentioned above are all search-based PCG approaches, which typically rely heavily on domain knowledge and complex modeling processes related to game mechanics, and tend to have relatively slow generation speed. Khalifa and Togelius[56] viewed generators as content itself coded by Marahel script. Each Marahel script functions as a constructive level generator that can be created by converting an integer-based chromosome into a corresponding script using a text generation tool. In their study, objective functions are defined for three simple game level problems separately, and a multi-objective approach is employed to search for available solutions. Constructive generators are capable of producing levels much faster, but rely on manually designed rules, which are often hard to specify and are based on complex game mechanics. Zhang et al.[31] enhanced multi-dimensional diversity by applying a generation framework via deep learning to optimize a content-based metric, a player-centered metric, and a playability metric. Their work demonstrates the conflicts between diversity metrics across two different game facets, but compromises playability by including it as an optimization objective. Though diversity improves during evolution, more low-playability content is generated, resulting in limited utility for agent testing. Moreover, their generation method is in an offline manner. For offline generators, content is determined before the gameplay begins. If researchers want to change certain settings of the generated scenario like level length to extend the testing phase, they need to adjust the parameters of the generator and regenerate new content. In contrast, online generation takes agent behavior and experience into consideration, while also enabling real-time generation during gameplay. To meet the requirements of AI testing for more complex and dynamic scenarios that can adapt to the agent's actions, online content generation is a more suitable choice.

Therefore, we aim to enhance the diversity in online content generation, as existing research in online content generation is all limited to single-objective optimization approaches. MOEAs can effectively balance diversity metrics, looking beyond monotonous game elements or a single game facet. Researchers can purposefully generate content online fit for more specific preferences by leveraging the trade-offs between different optimization objectives. Furthermore, we view playability as a constraint to improve diversity without compromising the usability of generated content.

3 Multi-Objective Procedural Content Generation via RL

In this section, we propose a Multi-Objective PCG via RL (MOPCGRL) framework, extending the single-

† Blizzard Entertainment, 1998, https://starcraft.blizzard.com/zh-tw/.

‡ Thinking Rabbit, 1982, https://zh.wikipedia.org/zh-cn/sokoban.

objective optimization approach in EDRL. This section first presents an overview of our framework and gives a detailed description of the key components in its process. Next, the diversity metrics used to formulate the objective functions are introduced.

3.1 Framework overview

The framework considers optimizing a set of online content generators (e.g., RL-based generators detailed in Section 3.2) denoted as $\mathcal{M}=\{M_1, M_2, \ldots, M_\lambda\}$ using a multi-objective optimizer π, where generators form a population within the evolutionary process. Figure 1 illustrates the detailed process. The framework consists of three algorithmic modules—the overall optimization framework (Algorithm 1), the pretraining stage (Algorithm 2), and the online generation process (Algorithm 3).

First, the models act as individuals of the initial population, which are pretrained for specified timesteps, and a replay memory is initialized if needed (Line 1 in Algorithm 3). In this paper, the pretraining stage is named mixed evolution warm-up, shown in Algorithm 2, which mixes models of different pretrained steps as the initial population. Then, the models are evaluated based on the diversity metrics $\boldsymbol{F}$ and constraints G of their generated content (Line 2 in Algorithm 1). After entering the main loop, the environmental selection component of π is applied to select λ better-performing models as the parent population P (Line 4 in Algorithm 1). Subsequently, we perform crossover and mutation on the parent population P. The resulting individuals are denoted as the offspring population O (Lines 5 and 6 in Algorithm 1). Then, each offspring model is trained t_o timesteps to stabilize the newly generated networks, and updated with replay memory (Lines 7 and 8 in Algorithm 1). At the end of every iteration, the offspring models are evaluated on all the metrics to prepare for the next round of environmental selection (Line 9 in Algorithm 1), until the algorithm has met its termination condition.

The RL generator model and the components of the multi-objective optimizer used in the framework can be selected according to the desired game content type, optimization functions, and constraints. Regarding the conflicts between different game diversity metrics, this paper optimizes a content-based divergence metric and a player-centered distance metric simultaneously, and takes level playability as the constraint of the evolutionary algorithm.

3.2 Modeling RL-based generators as individuals

Each individual is an Episodic Generative Soft Actor-Critic (EGSAC) model[22]. An evolutionary algorithm

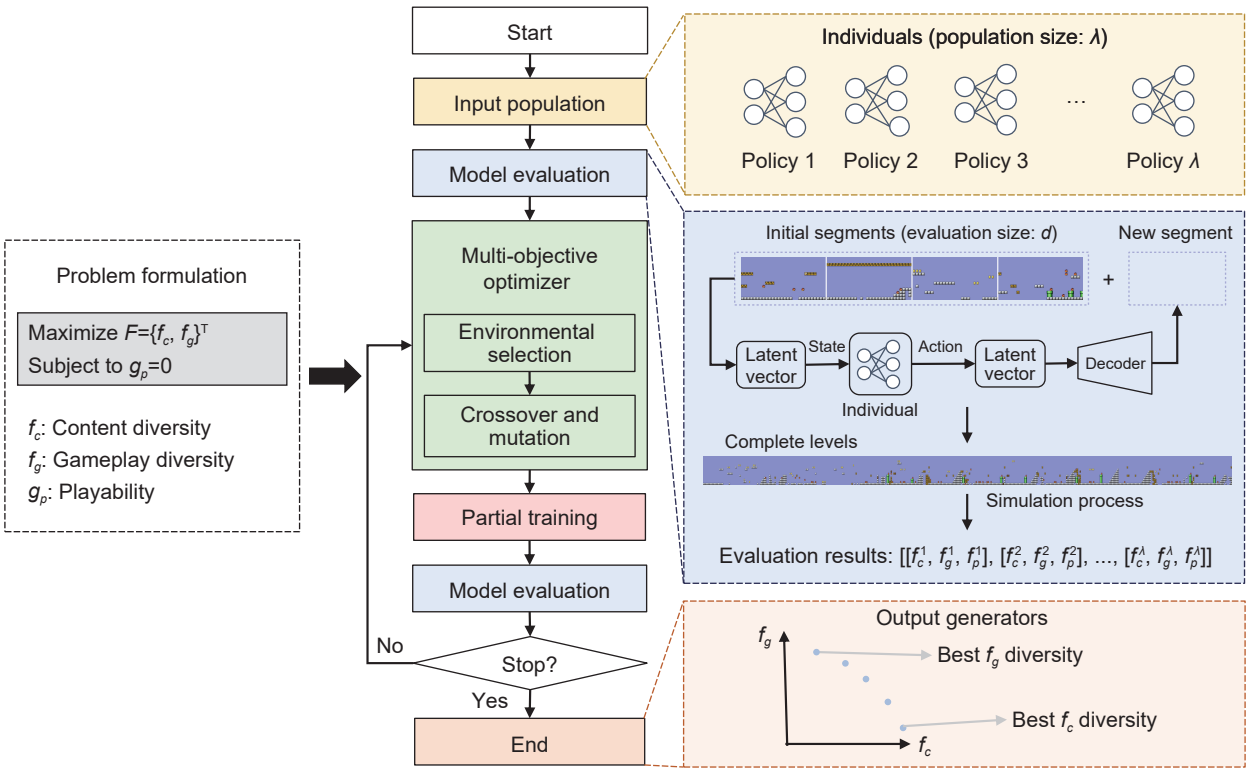


Fig. 1 Applying MOPCGRL framework to generate diverse levels of Super Mario Bros.

Algorithm 1 MOPCGRL framework

Input: Multi-objective optimizer π, populationsize λ, local search steps t_o, evaluation metric set F, constraint metric set G
Output: Optimized model set $\mathcal{M}$

Obtain pretrained models $\mathcal{M}$ as the set of pretrained models $\{M_1, M_2, \ldots, M_\lambda\}$ using mixed evolution warm-up (Algorithm 2) as the initial population;
Evaluate the pre-trained models in $\mathcal{M}$, obtain their objective values $F(\mathcal{M})$ as $\{\boldsymbol{F}_1, \boldsymbol{F}_2, \ldots, \boldsymbol{F}_\lambda\}$ and their constraint values $G(\mathcal{M})$ as $\{g_1, g_2, \ldots, g_\lambda\}$;
while termination has not been satisfied **do**
    Set P by selecting λ individuals from $\mathcal{M}$ through environmental selection component of π;
    Set O by performing crossover component of π on P;
    Update O by perform mutation component of π on O;
    Update O by training models O with t_o timesteps;
    Evaluate models in O, obtain their objective values $F(O)$ as $\{\boldsymbol{F}_1, \boldsymbol{F}_2, \ldots, \boldsymbol{F}_\lambda\}$ and constraint values $G(O)$ as $\{g_1, g_2, \ldots, g_\lambda\}$;
    Obtain $\mathcal{M}$ by performing environmental selection of π on $\mathcal{M} \cup O$ according to their objective and constraints values;
end

Algorithm 2 Mixed evolution warm-up

Input: Episode length e, batch size b, update frequency u, step counter t_c, transition counter trs, budget range T_{start} and T_{end}
Output: Pretrained model set $\mathcal{M}$ as $\{M_1, M_2, \ldots, M_\lambda\}$

Initialize an empty model set $\mathcal{M}$ to empty set $\varnothing$ and a model $\mathcal{M}$;
Initialize batch $\mathcal{B}$, a temporary buffer $\mathfrak{B}$, and replay memory $\mathcal{D}$;
Set t_c to 0 and trs to 0;
while $t_c \leqslant T_{\text{end}}$ **do**
    Perform online generation in Algorithm 3, obtain a buffer $\mathfrak{B}$;
    Update $\mathcal{D}$ by adding transitions and rewards from $\mathfrak{B}$;
    Update trs by adding $|\mathfrak{B}|$ to trs;
    if trs > u and $|\mathcal{D}| > \mathcal{B}$ **then**
        for i from 1 to $\lfloor \frac{\text{trs}}{u} \rfloor$ **do**
            Set $\mathcal{B}$ by randomly sampling a batch of size b from the replay memory $\mathcal{D}$;
            Update model M with $\mathcal{B}$;
        end
        Update trs to the result of trs % u;
    end
    if $T_{\text{start}} \leqslant t_c \leqslant T_{\text{end}}$ and t%10 000 = = 0 **then**
        Update $\mathcal{M}$ by adding the current model M to the set $\mathcal{M}$;
    end
    Increment t_c by 1;
end

searches over the weights of policy network models. Each individual consists of a policy network (actor) and a value network (critic), both of which contain a 3-layer MLP with 256 neurons per layer[22]. Crossover and mutation operations are performed on parameters of the policy network[57].

Algorithm 3 Online generation

Input: Episode length e and model M
Output: Buffer $\mathfrak{B}$

Initialize buffer $\mathfrak{B}$ to empty set $\varnothing$;
Initialize transition set U to empty set $\varnothing$;
Initialize a segment set l to contain the initial state $\{s_t\}$;
while $|l| < e$ **do**
    Sample next segment a_t using the policy in M;
    Update l by adding a_t to the set l;
    Obtain s_{t+1} by updating state based on a_t and s_t;
    Update U by adding the new transition (s_t, a_t, s_{t+1});
end
Obtain reward set $\mathcal{R}$ by computing total reward for level l;
Update $\mathfrak{B}$ by combining transition and reward set $(U, \mathcal{R})$;

In this paper, the initial state of the environment consists of n randomly generated initial segments, each segment is represented as a latent vector, denoted as s_t in the segment set in Algorithm 3 (Line 3). In the subsequent generation process, EGSAC samples an action of a new latent vector—the concatenation of the latest n latent vectors—according to the current state (Lines 5–7 in Algorithm 3). The transitions are stored in a temporary buffer (Line 8 in Algorithm 3). Each level for evaluation is composed of n initial segments and subsequent generated action segments. The accumulative reward of each segment is computed after the whole episode is completed (Line 10 in Algorithm 3). They are stored to the buffer with previous transitions, and later used to fill the replay memory (Line 11 in Algorithm 3). The initial segment set used to test the performance of the model is denoted as D_{eval}. Every model is evaluated on d levels generated from D_{eval}, where the quality of each level is simulated and assessed according to the agent's performance (shown in Fig. 1).

3.3 Multi-objective optimizer

Previous studies in multi-objective PCG consider offline generation and constraint-free cases[29, 31, 49–51], and often use NSGA-II[58] or NSGA-III[59] as the multi-objective optimizer. Despite of the diversity metrics, playability is also an essential metric for evaluating content quality, which directly affects the agent's performance. If a level is unplayable, it leads to the player's death and results in game failure. Therefore, we set playability as a constraint to limit the number of deaths of the agent within a certain range. Considering our problem formulation, the Constrained Two-Archive Evolutionary Algorithm (C-TAEA)[60] is used as the

multi-objective optimizer. The testing performance of C-TAEA on a series of artificial benchmark problems and a practical problem is significantly stronger than that of famous constrained MOEAs[60]. Thus, we choose C-TAEA as it is more suitable for high-dimensional decision space in optimizing parameters of content generators. To remain consistent with the original constrained multiobjective optimization problem setting in C-TAEA, the diversity scores are transformed by $1-f$ during algorithm execution for minimization, where f denotes the two objective functions described in Section 3.4. Since the binary crossover and the polynomial mutation used in the original paper are not suitable for evolving policy network parameters, we adopt the evolution operators from Ref. [57], which are specifically designed for this type of optimization. The archive management mechanism remains unchanged from the original algorithm.

Population initialization. The population size is denoted as λ, and set to 10 considering the simulation time required for agents to evaluate the diversity of the generated content. During the warm-up stage, the EGSAC model population is first pre-trained with a specified number of time steps. We use the designed initialization strategy mixed evolution warm-up introduced in Algorithm 2, to increase model diversity within the population by selecting pre-trained models of different step sizes with the same timestep interval in-between.

Specifically, we initialize an empty model set and a new model along with other parameters needed in training (Lines 1–3 in Algorithm 2). During the training session, after online generation (Line 5 in Algorithm 2), replay memory is updated with the transition information stored in the buffer (Line 6 in Algorithm 2). The model is updated several times under the condition that the transition counter is larger than the update frequency and the size of the replay memory exceeds the batch size (Lines 8–14 in Algorithm 2). Before the model is trained for T_{start} steps, no saving is performed. Starting from $T_{\text{start}}=200\,000$ steps, we save the model every 10 000 steps until T_{end} = 290 000, obtaining a total of 10 saved models as initial population (Line 15–17 in Algorithm 2).

Mutation. The mutation operator in this paper is to add noise directly to weight parameters in the policy network[57]. The noise is generated from a normal distribution with mean μ and standard deviation σ.

Crossover. During crossover, all parameters of the two parent networks are traversed. If the structure of the input parameter is a one-dimensional vector, one-point crossover is applied. If it is a 2-dimensional matrix, random rows are selected for exchanging information between parents[57].

3.4 Evaluation metrics

According to Liapis et al.[61], a game consists of six creative facets: visuals, audio, narrative, ludus, level architecture, and gameplay. For the Mario AI Framework used in Ref. [62], level geometry—such as platform lengths, gaps, and enemy distribution—directly influences the agent's gameplay trajectory. Conversely, its trajectory also reflects the types and distribution of tiles within the level. Therefore, we consider optimizing content architecture diversity f_{c} and gameplay diversity f_{g} to evaluate the behavioral patterns of the testing agents during gameplay[14]. The playability constraint, denoted as g_{p}, is assigned 0 if no constraint is violated. This indicates that the agent is able to complete the level within a limited number of failures. A detailed evaluation process of these three metrics is illustrated in Fig. 2.

Our aim is to maximize the objective functions while ensuring that the constraint is satisfied. In summary, the problem of this work can be modeled as follows:

$$\text{maximize } \boldsymbol{F}=\{f_c,f_g\}^{\text{T}} \tag{1}$$

$$\text{subject to } g_p=0 \tag{2}$$

Content diversity. The content diversity of generated 2D segments is assessed by tile-pattern Jensen-Shannon divergence[27] following Refs. [22, 23, 31], denoted as $\text{TPJS}(.,.)$. It evaluates the frequency of

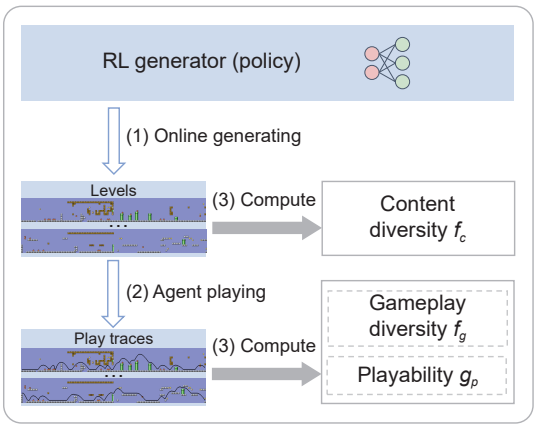


Fig. 2 Process of evaluating content diversity, gameplay diversity, and playability.

distinct tile patterns between the i-th segment S_i and the k-th previously generated segment S_{i-k}. A higher TPJS divergence value indicates greater diversity in tile distributions[22]. $R_c(S_i)$ denotes the clipped TPJS value of the i-th segement S_i,

$$R_c(S_i) = \frac{\sum_{k=1}^{n} \mathrm{AC}\,(\mathrm{TPJS}\,(S_i, S_{i-k});\, t_c,\, r_k)}{\sum_{k=1}^{n} r_k} \tag{3}$$

where $\mathrm{AC}\,(\cdot)$ is an A-clip function to adjust the amount of variation within segments; t_c and $r_k = 1 - k/(n+1)$ represent the target diversity value of the reward function and the clipping rate, respectively; n denotes the observation length of the online generator explained in Section 3.2. Let m be the metric being clipped, $\mathrm{AC}\,(\cdot)$ is formulated as

$$\mathrm{AC}\,(m) = \min\left\{r_m,\, 1 - \frac{|m - t_m|}{t_m}\right\},\ m \in \{l,\, g\} \tag{4}$$

Accordingly, the reward functions R_c and R_g for each level segment S are derived. Then, f_c is defined as the mean reward of the clipped metric on all $d \times e$ level segments, aiming to maximize the objective value for better diversity,

$$f_c = \frac{1}{d}\frac{1}{e}\sum_{j=1}^{d}\sum_{i=1}^{e} R_c^{ij}\,(S_{ij};\, t_c,\, n) \tag{5}$$

Gameplay diversity. Following previous work[22, 23, 31], the diversity of play traces produced by agents is calculated by dynamic time warping $\mathrm{DTW}\,(.,.)$, which tracks the sequence of coordinates $\tau\,(\cdot)$ of the agent while passing segments S_i and S_{i-k}. $R_g(S_i)$ denotes the clipped DTW value of the i-th segement S_i,

$$R_g(S_i) = \frac{\sum_{k=1}^{n} \mathrm{AC}\,(\mathrm{DTW}\,(\tau(S_i),\, \tau(S_{i-k}));\, t_g,\, r_k)}{\sum_{k=1}^{n} r_k} \tag{6}$$

where t_g is another target diversity value set separately from t_c. A higher DTW value indicates the agent's movement path is more complex. Similar to f_c, the second objective f_g is formulated for maximization,

$$f_g = \frac{1}{d}\frac{1}{e}\sum_{j=1}^{d}\sum_{i=1}^{e} R_g^{ij}(S_{ij};\, t_g,\, n) \tag{7}$$

Playability. For playability constraint evaluation, if a level segment allows the agent to pass smoothly during simulation, the playability for that segment is recorded as 0, and −1 if the agent dies due to flawed generation. R_p denotes whether the current segment causes the agent's death, ϵ is the manually defined threshold of segments that the agent should be able to pass successfully in a complete level. Note that this definition is focused on the playability of game content. Agents and human players follow the same rules during the gameplay process. A level is considered playable if there is at least one feasible path to completion (i.e., the level can be solved)[22, 23], regardless of human performance skills. The formulation of playability is defined as follows:

$$g_p = \begin{cases} -\alpha, & \alpha < 0; \\ 0, & \alpha > 0 \end{cases} \tag{8}$$

where α is measured by excluding all flawed segments from total episode length e, normalizing it by dividing ϵ then subtracting by 1 to set α of all individuals that violate the constraint negative (consistent with the original design of C-TAEA[60]),

$$\alpha = \frac{(e + \frac{1}{d}\frac{1}{e}\sum_{i=1}^{d}\sum_{j=1}^{e} R_p^{ij})}{\epsilon} - 1 \tag{9}$$

4 Experiment

This section lists the details of experiments carried out in this paper. We begin with experimental settings and the benchmark used. In the following sections, we conduct two sets of experiments to evaluate the framework. To validate the effectiveness of our approach, we compare several online generation methods performed well on our two diversity metrics: (1) three SACs including the classic SAC[63], state-of-the-art EGSAC[22], and ASAC[23]; (2) four ensemble generation methods published within the past five years, including PMOE[64], DvD[65], SUNRISE[66], and NCERL[23]. Then, we verify the effectiveness of our initialization approach through an ablation study, and measure the performance of training with a varied initial population with multi-objective optimization indicators.

4.1 Experimental settings and benchmark

Our framework can be divided into two aspects: the training process of the RL model and the evolution process of MOEA. Table 1 lists the parameters of two aspects used in our experiment. Each level is generated from 4 initial segments with an episode length e of 25

Table 1 Parameter settings of MOPCGRL.

Aspect	Parameter
Model training	Episode length e: 25
	Batch size b: 384
	Update frequency u: 10
	Target divergence t_c, t_g: 0.1, 0.25
	Observation length n: 4
Multi-objective evolution process	Population size λ: 10
	Number of iterations: 100
	Playability constraint ϵ: 24
	Evaluation size d: 20
	Mutation noise μ, σ: 0, 0.02
	Local search steps t_o: 5000

(the initial segments are not included). Episode length is set longer than the longest original levels in Super Mario Bros to demonstrate that the length in online generation is independent from the original game setting. Considering pretrained steps, evolution generations, and local search steps, the total budget in a single MOPCGRL run is within 800 000 timesteps, lower than that of the compared models (1 000 000 timesteps). All experiments are conducted on an NVIDIA A800 80 GB PCIe GPU and an AMD EPYC 9654 96-Core Processor CPU. One run of MOPCGRL with current parameter settings takes approximately 4−5 days to complete. The majority of the training time is spent on agent-based gaming simulations, which is executed on CPU. To align with the number of our experimental runs for a more robust statistical analysis, we train additional models for each method using their provided code[♯]. The hyperparameters remain the same as those in their original papers for the training and evaluation of our comparative experiments.

C-TAEA maintains two different solution sets at the end of each generation (a convergence-oriented archive and a diversity-oriented archive, denoted as CA and DA, respectively), for all experiment analysis and visualization presented in this paper, we select DA for demonstration, as DA explores the regions under-exploited by CA and achieves better performance over CA in terms of both diversity and convergence in our work.

Benchmark. The experiments are conducted on the Mario AI benchmark[62], a classic tile-based 2D platform game that has been commonly used as a benchmark problem in PCG[7]. Each tile type in a Mario level is represented by a distinct integer that can be converted to a one-hot vector. The entire level can thus be represented as a latent vector with one-hot encoding. The latent vector is generated through uniform random sampling, and the decoder that converts the latent vector into a level is MarioGAN[67].

4.2 Effectiveness of MOPCGRL

MOPCGRL aims to balance and improve content diversity and gameplay diversity by providing a set of online generators. To verify the effectiveness of MOPCGRL, we compare the generation results of ten independent runs of classic and state-of-the-art online generation methods (SAC[63], EGSAC[22], ASAC[23], PMOE[64], DvD[65], SUNRISE[66], and NCERL[23] with different regularization parameters denoted as l ranging from 0 to 0.5) on 100 testing levels. The performance results are visualized in Fig. 3, where most generators obtained by MOPCGRL outperform those generated by other methods. The levels produced by some generators, such as PMOE and NCERL, with regularization parameters l larger than 0.2, have rather unstable quality on two diversity metrics. In contrast, MOPCGRL generators demonstrate greater stability while still achieving high diversity scores. They mainly aggregate in the region where gameplay diversity is greater than 0.94, and more than half of them also exhibit higher game content diversity compared to other methods, reaching above 0.97.

The evolution process of one arbitrary run is illustrated in Fig. 4. As the number of generations increases, the color of markers turns deeper. The population converges quickly towards the Pareto front marked in red at the first half of the evolution process. Testing results may fluctuate near the front, possibly due to certain generated segments exhibiting low diversity, which leads to decease in overall level quality after concatenation.

In Table 2, besides the compared models, we choose three specific models from each run of MOPCGRL and calculate their mean and standard deviation values: the model performs best on content diversity f_c, gameplay diversity f_g, and the knee point. The generators of the best f_c and f_g are able to achieve the best performance respectively, and the performance of the knee point models are in between. The model with the highest hypervolume[53] value is selected as the knee point, which represents the most balanced solution on two objectives[31, 68]. Under the condition that playability constraints are satisfied (96% playability, at least 24 out of 25 segments are playable), all selected models

♯https://github.com/PneuC/NCERL-Diverse-PCG

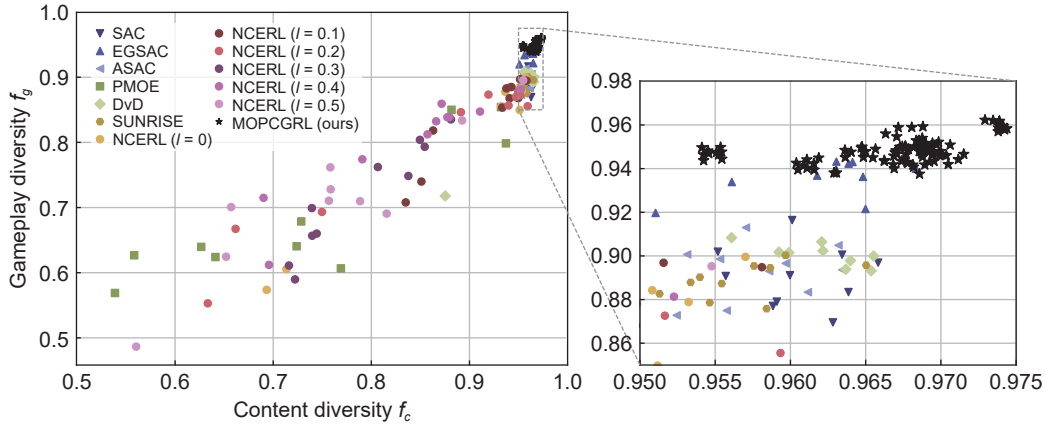


Fig. 3 All ten independent runs of MOPCGRL experiments compared with classic and state-of-the-art online PCG methods, higher metric score indicates better content and gameplay diversity.

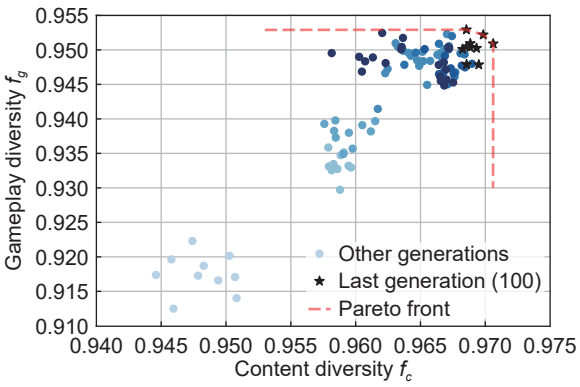


Fig. 4 MOPCGRL evolution process from one arbitrary run. The color deepens with the increase of generations, and the last generation is marked in black stars. The Pareto front formed by the last generation is illustrated in red.

from MOPCGRL outperform other online generation methods on both diversity metrics, and maintain decent playability with nearly all generated segments playable.

The dominance relationship between MOPCGRL generators and the compared algorithms is further listed in Table 3. Game content diversity and gameplay diversity of every MOPCGRL model in 10 independent runs are compared with baseline models obtained from each run. We quantify dominance relationships between them using three categorical metrics: Dominate, Incomparable, and Dominated values[69]. Specifically, if MOPCGRL generators perform better on both metrics, they dominate the compared generator. If incomparable, only one metric outperforms the compared one, and is dominated if both metrics are outperformed. The performance of MOPCGRL generators dominates all of PMOE and NCERL (l = 0.3, 0.4, 0.5) generators, only very few are incomparable with the rest NCERL generators. SAC and DvD generators perform better on the two metrics than NCERL and PMOE, but are still outperformed by MOPCGRL. The dominance over EGSAC generators is the lowest, while still reaching three-fourths of all compared pairs. These results verify MOPCGRL framework is able to optimize content diversity and gameplay diversity of the levels generated by online generators, while ensuring strong playability.

To further address the performance difference among methods, we conduct a statistical significance analysis using the Nemenyi post-hoc test with the significance level set to 0.05. The resulting critical difference diagrams (Fig. 5) present the difference in rankings on f_c and f_g. MOPCGRL is statistically better compared to PMOE and NCERL variants on both diversity objectives, and remains superior to other ensemble and non-ensemble baselines.

In conclusion, we present a constrained multi-objective approach for training online content generators that is able to balance conflicting objectives. It outperforms classic and state-of-the-art approaches when converging to the Pareto front.

4.3 Effectiveness of mixed evolution warm-up

To assess the effectiveness of our population initialization strategy, mixed evolution warm-up, we use four commonly used indicators in multi-objective optimization[70]: HV[53], Generational Distance (GD)[71], Pure Diversity (PD)[72], and spacing (SP)[73]. The overall quality of generators is evaluated using HV, which reflects convergence, spread, uniformity, and cardinality of the multi-objective optimization process[45]. The other indicators are used to evaluate the convergence, spread, and uniformity, respectively.

Four combinations of initial populations are tested,

Table 2 Performance comparison on three metrics (f_c, f_g, and playability percentage converted from g_p) across different methods. Generators on best f_c, best f_g, and the knee point are selected in each run and averaged. As all of generated levels do not violate the constraint (achieve 96% of playability), we calculate the percentage of playability for more precised comparison. The best-performing results on two objective functions and the constraint are bolded in black. The standard deviations are recorded in parentheses.

Method	f_c (±std) (%)	f_g (±std) (%)	Playability (±std) (%)
SAC	0.9605 (±0.0035)	0.8907 (±0.0139)	**99.94 (±0.04)**
EGSAC	0.9621 (±0.0050)	0.9356 (±0.0085)	99.85 (±0.07)
ASAC	0.9580 (±0.0039)	0.8931 (±0.0128)	99.90 (±0.11)
PMOE	0.7337 (±0.1464)	0.6888 (±0.1051)	96.79 (±2.32)
DvD	0.9533 (±0.0276)	0.8823 (±0.0580)	99.82 (±0.30)
SUNRISE	0.9568 (±0.0039)	0.8888 (±0.0080)	99.86 (±0.08)
NCERL (l = 0.0)	0.8977 (±0.1028)	0.8178 (±0.1214)	99.23 (±1.55)
NCERL (l = 0.1)	0.9163 (±0.0470)	0.8415 (±0.0665)	99.33 (±1.21)
NCERL (l = 0.2)	0.8603 (±0.1281)	0.7966 (±0.1155)	98.50 (±2.07)
NCERL (l = 0.3)	0.7891 (±0.0630)	0.7160 (±0.0850)	97.59 (±1.48)
NCERL (l = 0.4)	0.8392 (±0.0871)	0.8011 (±0.0815)	98.71 (±1.10)
NCERL (l = 0.5)	0.7594 (±0.1162)	0.7141 (±0.1106)	97.22 (±1.73)
MOPCGRL (best average f_c)	**0.9677 (±0.0052)**	0.9485 (±0.0069)	99.84 (±0.08)
MOPCGRL (best average f_g)	0.9666 (±0.0055)	**0.9520 (±0.0061)**	99.84 (±0.10)
MOPCGRL (knee point)	0.9675 (±0.0052)	0.9510 (±0.0064)	99.84 (±0.08)

Table 3 Dominate, incomparable, and dominated percentages (mean (±standard deviation)) of MOPCGRL averaged over 10 runs compared with classic and state-of-the-art algorithms. A dominate (incomparable/dominated) value of 100% indicates that ten out of ten models in one population dominates (unable to dominate/is dominated by) one compared model. Arrows indicate preference: ↑ higher is better; ↓ lower is better. (%)

Algorithm	Dominate (↑)	Incomparable (↓)	Dominated (↓)
SAC	85.4 (±29.4)	14.6 (±29.4)	0 (±0)
EGSAC	75.2 (±31.1)	22.0 (±27.3)	2.8 (±8.4)
ASAC	89.8 (±23.5)	10.2 (±23.5)	0 (±0)
PMOE	100.0 (±0)	0 (±0)	0 (±0)
DvD	83.3 (±29.8)	16.7 (±29.8)	0 (±0)
SUNRISE	91.8 (±18.9)	8.2 (±18.9)	0 (±0)
NCERL (l=0.0)	99.0 (±3.0)	1.0 (±3.0)	0 (±0)
NCERL (l=0.1)	99.0 (±3.0)	1.0 (±3.0)	0 (±0)
NCERL (l=0.2)	99.0 (±3.0)	1.0 (±3.0)	0 (±0)
NCERL (l=0.3)	100.0 (±0)	0 (±0)	0 (±0)
NCERL (l=0.4)	100.0 (±0)	0 (±0)	0 (±0)
NCERL (l=0.5)	99.5 (±1.5)	0.5 (±1.5)	0 (±0)

each consisting of models pretrained for a different number of timesteps. Mixed evolution warm-up in one MOPCGRL run utilizes 10 pretrained models evenly sampled at 10 000-timestep intervals obtained from a single EGSAC training run, ranging from 200 000 to 290 000 timesteps. The rest three populations used in the ablation study each consist of 10 pretrained models with identical training timesteps: 200 000, 250 000, and 290 000, respectively. Each population is obtained from 10 different EGSAC training runs. We compare 10 independent runs of MOPCGRL initialized with mixed evolution warm-up (obtained from 10 EGSAC models trained independently to 290 000 timesteps) against five runs of MOPCGRL for each of the three baseline populations, where each population consists of models trained to the same timestep (obtained from 50 runs of EGSAC to 290 000 timesteps).

The calculation of HV and GD requires the nadir point derived from the true Pareto front. However, for content generators, the true Pareto front cannot be obtained directly. We aggregate the test results of generators from all independent experiments to calculate the estimation of the true Pareto front. While normalizing objective values in HV, we scale the objective space to half the original range. All quality indicator values are calculated every 10 generations in one run, with a total of 100 generations.

Average value of four indicators over four initialization strategies is illustrated in Fig. 6. The mean and standard deviation indicator values of the initial population and last generation are further listed in Table 4. Except for the population of 200 000 timesteps, the HV values of the other three initialization strategies increase with the number of

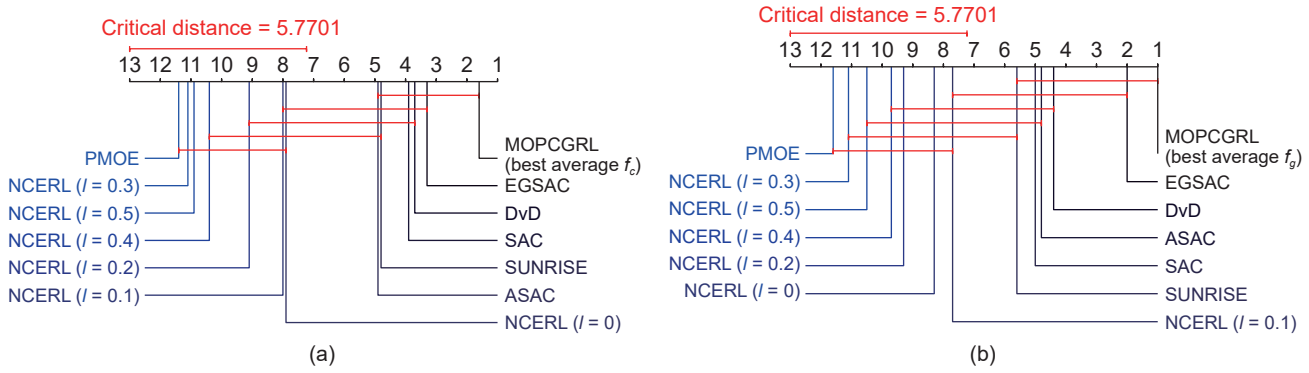


Fig. 5 Critical distance diagrams on both objectives (a) f_c and (b) f_g.

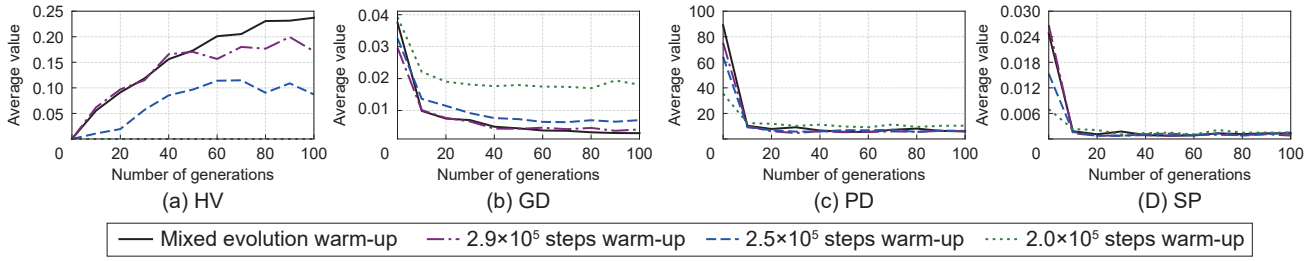


Fig. 6 Four different initialization strategies evaluated on four multi-objective optimization indicators: HV, GD, PD, and SP.

Table 4 Average (±standard deviation) on four multi-objective optimization indicators across four initialization strategies at the initial state and final generation. Mixed evolution warm-up samples models pretrained from 2.0×10^5 to 2.9×10^5 timesteps per 0.1×10^5 timesteps, the rest three populations contain models of identical training timesteps: 2.9×10^5, 2.5×10^5, and 2.0×10^5. Arrows indicate preference: ↑ denotes higher is better; ↓ denotes lower is better. The best results are highlighted in bold.

Population state	Initialization strategy	HV (↑)	GD (↓)	PD (↑)	SP (↓)
Initial	Mixed evolution warm-up	0.0000 (±0.0000)	0.0375 (±0.0091)	**89.0458 (±67.0840)**	0.0248 (±0.0391)
	2.9×10^5 steps warm-up	0.0000 (±0.0000)	**0.0296 (±0.0048)**	75.2852 (±50.0033)	0.0268 (±0.0279)
	2.5×10^5 steps warm-up	0.0000 (±0.0000)	0.0326 (±0.0027)	64.2461 (±20.9113)	0.0153 (±0.0092)
	2.0×10^5 steps warm-up	0.0000 (±0.0000)	0.0393 (±0.0033)	35.5383 (±9.5786)	**0.0070 (±0.0056)**
Final	Mixed evolution warm-up	**0.2355 (±0.0700)**	**0.0029 (±0.0015)**	4.5189 (±1.5741)	**0.0006 (±0.0002)**
	2.9×10^5 steps warm-up	0.1716 (±0.0567)	0.0041 (±0.0018)	6.1448 (±1.5964)	0.0008 (±0.0005)
	2.5×10^5 steps warm-up	0.0871 (±0.0537)	0.0070 (±0.0020)	6.3265 (±1.0826)	0.0015 (±0.0004)
	2.0×10^5 steps warm-up	0.0000 (±0.0000)	0.0181 (±0.0021)	**10.3941 (±2.0854)**	0.0015 (±0.0008)

generations, indicating that the solution sets are constantly approaching the Pareto front while maintaining a good distribution. The population using mixed evolution warm-up ranks highest of all, followed by 290 000-timestep populations. Unsurprisingly, 250 000-timestep populations rank third highest in HV since their budgets lie between the least and the most trained models. The average HV value of 200 000-timestep populations remains 0 is due to its poor optimization performance; all generators produced by these runs are unable to reach the region between the estimated Pareto front and the nadir point. The decrease of GD values on all four strategies proves the convergence of our approach. The initial state of 290 000 warm-up reaches the lowest GD value due to its highest training budget, indicating its better convergence at the beginning. However, it is outperformed by the population using mixed evolution warm-up at the final generation. Initial PD values suggest that mixed evolution warm-up has the widest spread across both diversity metrics. The decrease of average PD value suggests that the problems exhibit a small final front, and the populations gradually approach the final front from their scattered initial state. In the final generation, affected by the convergence of the population, those further from the Pareto front tend to have higher PD values. The populations with lower initial timesteps demonstrate

more instability. Compared to the mixed evolution warm-up and 290 000-timestep warm-up, SP values of 200 000 and 250 000 warm-up exhibit greater variability, reflecting less uniform distribution of the solution sets.

We select one arbitrary run from four initialization strategies respectively and illustrates their initial states and the solution sets of the final generation in Fig. 7. The initial state of mixed evolution warm-up exhibits the widest range on two diversity metrics, suggesting a more diverse and well-distributed starting population. Applying the mixed evolution warm-up strategy, one EGSAC training can fully support the initialization of one MOPCGRL experiment and be able to outperform the initial population composed entirely of models trained with the highest timestep. It ensures the diversity of initialization at the beginning of the evolution process while keeping training costs minimal.

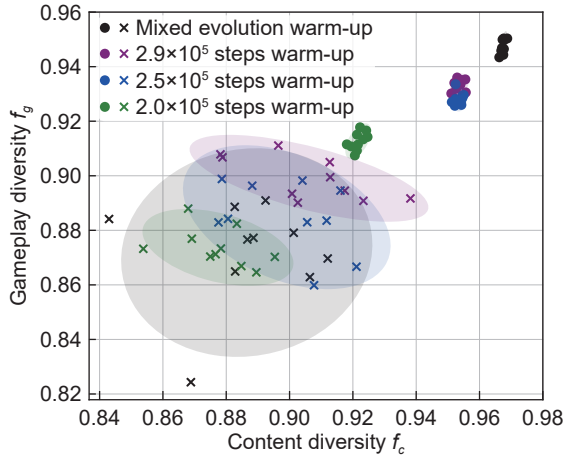


Fig. 7 Four arbitrary runs from four initialization strategies are selected separately. Their initial states are marked as "×" and their final states are marked as dots.

4.4 Sensitivity analysis

To examine the key hyperparameters that may affect model performance, we conduct a sensitivity study on the population size $\lambda \in \{5, 10\}$ and the playability constraint $\epsilon \in \{20, 22, 24\}$. For $\lambda = 5$, the number of iterations is set to 200 to match the total budget of the original setting ($\lambda = 10$ with 100 iterations). The other parameters are kept the same as those listed in Table 1. All comparative experiments are repeated for five runs.

The sensitivity analysis for both parameter sets is also evaluated using HV, GD, PD, and SP. Their comparison is illustrated in Fig. 8. Figure 8a demonstrates results of parameter set $\lambda \in \{5, 10\}$. In the early stages of the evolution process, the experiments with a population size $\lambda = 5$ obtain higher HV values, but they stabilize around 0.15 without further improvement. In contrast, the HV values for the $\lambda = 10$ population continue to increase steadily, ultimately outperforming the population of $\lambda = 5$ once they reach the same budget. This indicates that a smaller population may exhibit quick progress in finding quality solutions due to its size, yet its limited exploration capacity prevents further improvement. A larger population size allows more model to explore more policies in parallel, enabling MOPCGRL to discover a broader range of solutions. The other three indicators, despite the randomness of initial states,

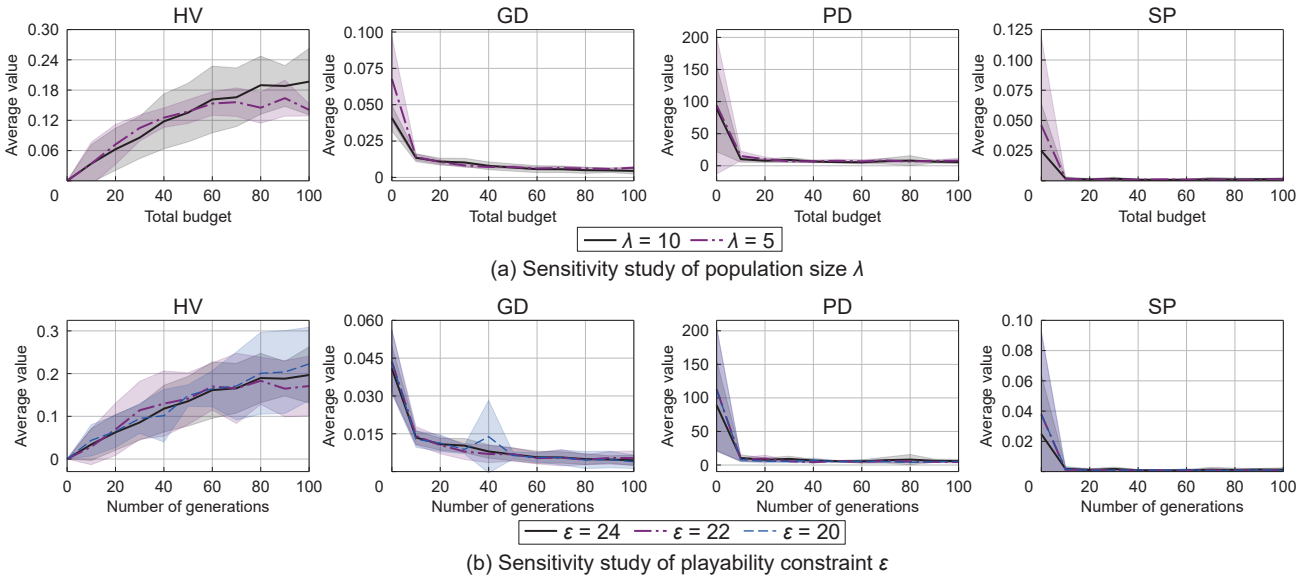


Fig. 8 Sensitivity study of population size λ and playability constraint ϵ.

fail to distinguish more between the two types of populations.

As can be seen in Fig. 8b, changing the playability constraint ϵ to smaller values does not significantly affect the testing results of MOPCGRL. The final HV values fluctuate slightly around 0.2, but they are not clearly correlated with ϵ. While there exist trade-offs between playability and diversity metrics[31], our exploration on varying ϵ does not seem to capture this relationship. A possible explanation is that since playability is a prior in online generation, models in our initial population can already generate diverse contents with relatively high playability. Lowering the value ϵ will not affect as much under this circumstance.

5 Conclusion

In our paper, we propose an MOPCGRL framework for online content generation. The framework combines multi-objective evolutionary reinforcement learning with procedural content generation techniques, providing a set of online game generators capable of balancing multi-dimensional diversity metrics. Experiment results demonstrate the effectiveness of our approach; the models obtained by MOPCGRL outperform seven generation methods, including both classic and recent state-of-the-art approaches, by achieving the best diversity scores. To enhance the evolution performance, we design a population initialization strategy and test it with three alternative populations through an ablation study. We analyze the generation results by comparing HV, PD, GD, and SP. Our strategy with mixed pretrained models as initial population, namely mixed evolution warm-up, shows the best diversity and convergence in the evolution process.

For future study, approaches can be explored to reduce the training time of MOPCGRL models, as the simulation process at the end of each generation is time-consuming. Implementing agent trace prediction or surrogate models may improve time efficiency. Second, additional diversity metrics can be taken into consideration. As demonstrated in Ref. [23], hamming distance contradicts the linear sum of two different diversity metrics. We are interested in expanding the number of simultaneously optimized metrics with different combinations. The integration of explainability or interpretability tools[55, 74] is also helpful for analyzing meaningful correlations between objective values and specific content design. Furthermore, applying multi-objective online generation to other game genres is also a direction worth exploring. Since our proposed framework is not tied to specific game rules or mechanics, it is transferable to other genres as long as the generated content can be represented and evaluated under appropriate objectives. This also introduces new challenges like determining which genres benefit most from online generation. Games that require real-time interaction and are sensitive to environmental uncertainty should be considered—such as track generation in racing games or task generation in multiplayer cooperative games. For larger content spaces such as 3D worlds, extending MOPCGRL would require more expressive content representations and effective simulation strategies. For example, mapping 3D levels to 2D sketches and computing the top-view tile distribution may reduce the computational cost of diversity evaluation.

In terms of multi-objective approaches, the MOEAs currently used in the field of PCG are primarily Pareto-dominance-based. Recently proposed decomposition-based many-objective optimization methods[75–77] and multi-objective genetic programming[78] could be explored for complex game scenario generation and may serve as comparative methods with our approach. Quality-diversity methods[79], such as MAP-Elites[80] and its variants, have been used in PCG to enhance content diversity as well. With their focus more on solution set coverage, they can yield different outcomes compared to MOEAs when applied to online generation. Multi-agent RL methods[81, 82] could also be employed to collaboratively generate game scenarios.

Acknowledgment

This work was supported by the National Key R&D Program of China (No. 2023YFE0106300), the National Natural Science Foundation of China (Nos. 62250710682 and 62476119), and internal grant of the Lingnan University, Hong Kong, China.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] G. N. Yannakakis and J. Togelius, *Artificial Intelligence and Games*. Cham, Switzerland: Springer, 2018.
[2] R. Gallotta, G. Todd, M. Zammit, S. Earle, A. Liapis, J.

Togelius, and G. N. Yannakakis, Large language models and games: A survey and roadmap, *IEEE Trans. Games*, doi: 10.1109/TG.2024.3461510.

[3] P. Rohlfshagen, J. Liu, D. Perez-Liebana, and S. M. Lucas, Pac-Man conquers academia: Two decades of research using a classic arcade game, *IEEE Trans. Games*, vol. 10, no. 3, pp. 233–256, 2018.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of Go with deep neural networks and tree search, *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[5] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al., Grandmaster level in StarCraft II using multi-agent reinforcement learning, *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[6] D. Perez-Liebana, J. Liu, A. Khalifa, R. D. Gaina, J. Togelius, and S. M. Lucas, General video game AI: A multitrack framework for evaluating agents, games, and content generation algorithms, *IEEE Trans. Games*, vol. 11, no. 3, pp. 195–214, 2019.

[7] C. Hu, Y. Zhao, Z. Wang, H. Du, and J. Liu, Games for artificial intelligence research: A review and perspectives, *IEEE Trans. Artif. Intell.*, vol. 5, no. 12, pp. 5949–5968, 2024.

[8] Y. Zhao, C. Hu, and J. Liu, Playing with Monte-Carlo tree search [AI-eXplained], *IEEE Computat. Intell. Mag.*, vol. 19, no. 1, pp. 85–86, 2024.

[9] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, Search-based procedural content generation: A taxonomy and survey, *IEEE Trans. Computat. Intell. AI Games*, vol. 3, no. 3, pp. 172–186, 2011.

[10] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation in Games*. Cham, Switzerland: Springer, 2016.

[11] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgard, A. K. Hoover, A. Isaksen, A. Nealen, and J. Togelius, Procedural content generation via machine learning (PCGML), *IEEE Trans. Games*, vol. 10, no. 3, pp. 257–270, 2018.

[12] J. Liu, S. Snodgrass, A. Khalifa, S. Risi, G. N. Yannakakis, and J. Togelius, Deep learning for procedural content generation, *Neural Comput. Appl.*, vol. 33, no. 1, pp. 19–37, 2021.

[13] C. Hu, Y. Zhao, and J. Liu, Game generation via large language models, in *Proc. 2024 IEEE Conf. Games*, Milan, Italy, 2024, pp. 1−4.

[14] Y. Li, Z. Wang, Q. Zhang, B. Yuan, and J. Liu, Measuring diversity of game scenarios, *IEEE Trans. Games*, vol. 17, no. 3, pp. 558–581, 2025.

[15] S. Risi and J. Togelius, Increasing generality in machine learning through procedural content generation, *Nat. Mach. Intell.*, vol. 2, no. 8, pp. 428–436, 2020.

[16] N. Justesen, R. R. Torrado, P. Bontrager, A. Khalifa, J. Togelius, and S. Risi, Illuminating generalization in deep reinforcement learning through procedural level generation, arXiv preprint arXiv: 1806.10729, 2018.

[17] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, Leveraging procedural generation to benchmark reinforcement learning, in *Proc. 37th Int. Conf. Machine Learning*, Virtual Event, 2020, pp. 2048−2056.

[18] M. Preuss, A. Liapis, and J. Togelius, Searching for good and diverse game levels, in *Proc. 2014 IEEE Conf. Computational Intelligence and Games*, Dortmund, Germany, 2014, pp. 1−8.

[19] M. Beukman, C. W. Cleghorn, and S. James, Procedural content generation using neuroevolution and novelty search for diverse video game levels, in *Proc. Genetic and Evolutionary Computation Conf.*, Boston, MA, USA, 2022, pp. 1028−1037.

[20] Z. Wang, T. Shu, and J. Liu, State-space closure: Revisiting endless online level generation via reinforcement learning, *IEEE Trans. Games*, vol. 16, no. 2, pp. 489–492, 2024.

[21] T. Shu, J. Liu, and G. N. Yannakakis, Experience-driven PCG via reinforcement learning: A Super Mario Bros study, in *Proc. 2021 IEEE Conf. Games*, Copenhagen, Denmark, 2021, pp. 1−9.

[22] Z. Wang, J. Liu, and G. N. Yannakakis, The fun facets of Mario: Multifaceted experience-driven PCG via reinforcement learning, in *Proc. 17th Int. Conf. Foundations of Digital Games*, Athens, Greece, 2022, p. 44.

[23] Z. Wang, C. Hu, J. Liu, and X. Yao, Negatively correlated ensemble reinforcement learning for online diverse game level generation, in *Proc. 12th Int. Conf. Learning Representations*, Vienna, Austria, https://iclr.cc/virtual/2024/poster/18072, 2024.

[24] S. Earle, M. Edwards, A. Khalifa, P. Bontrager, and J. Togelius, Learning controllable content generators, in *Proc. 2021 IEEE Conf. Games*, Copenhagen, Denmark, 2021, pp. 1−9.

[25] Z. Jiang, S. Earle, M. Green, and J. Togelius, Learning controllable 3D level generators, in *Proc. 17th Int. Conf. Foundations of Digital Games*, Athens, Greece, 2022, p. 71.

[26] A. Alvarez, S. Dahlskog, J. Font, J. Holmberg, and S. Johansson, Assessing aesthetic criteria in the evolutionary dungeon designer, in *Proc. 13th Int. Conf. Foundations of Digital Games*, Malmö, Sweden, 2018, p. 44.

[27] S. M. Lucas and V. Volz, Tile pattern KL-divergence for analysing and evolving game levels, in *Proc. Genetic and Evolutionary Computation Conf.*, Prague, Czech Republic, 2019, pp. 170−178.

[28] S. Dai, X. Zhu, N. Li, T. Dai, and Z. Wang, Procedural level generation with diffusion models from a single example, in *Proc. 38th AAAI Conf. Artificial Intelligence*, Vancouver, Canada, 2024, pp. 10021−10029.

[29] D. Loiacono and L. Arnaboldi, Multiobjective evolutionary map design for Cube 2: Sauerbraten, *IEEE Trans. Games*, vol. 11, no. 1, pp. 36–47, 2019.

[30] J. Togelius, M. Preuss, and G. N. Yannakakis, Towards multiobjective procedural map generation, in *Proc. 2010 Workshop on Procedural Content Generation in Games*, Monterey, CA, USA, 2010, p. 3.

[31] Q. Zhang, Z. Wang, Y. Li, K. Zhang, B. Yuan and J. Liu, Expanding horizons of level diversity via multi-objective evolutionary learning, *IEEE Trans. Artif. Intell.*, doi: 10.1109/TAI.2024.3489534.

[32] N. Shaker, G. Yannakakis, and J. Togelius, Towards automatic personalized content generation for platform games, in *Proc. 6th AAAI Conf. Artificial Intelligence and Interactive Digital Entertainment*, Stanford, CA, USA, 2010, pp. 63−68.

[33] M. Jennings-Teats, G. Smith, and N. Wardrip-Fruin, Polymorph: A model for dynamic level generation, in *Proc. 6th AAAI Conf. Artificial Intelligence and Interactive Digital Entertainment*, Stanford, CA, USA, 2010, pp. 138−143.

[34] D. Stammer, T. Günther, and M. Preuss, Player-adaptive Spelunky level generation, in *Proc. 2015 IEEE Conf. Computational Intelligence and Games*, Tainan, China, 2015, pp. 130−137.

[35] P. Shi and K. Chen, Online level generation in Super Mario Bros via learning constructive primitives, in *Proc. 2016 IEEE Conf. Computational Intelligence and Games*, Santorini, Greece, 2016, pp. 1−8.

[36] P. Shi and K. Chen, Learning constructive primitives for real-time dynamic difficulty adjustment in Super Mario Bros, *IEEE Trans. Games*, vol. 10, no. 2, pp. 155–169, 2018.

[37] A. Khalifa, P. Bontrager, S. Earle, and J. Togelius, PCGRL: Procedural content generation via reinforcement learning, in *Proc. 16th AAAI Conf. Artificial Intelligence and Interactive Digital Entertainment*, Virtual Event, 2020, pp. 95−101.

[38] Z. Wang and J. Liu, Online game level generation from music, in *Proc. 2022 IEEE Conf. Games*, Beijing, China, 2022, pp. 119−126.

[39] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[40] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial networks, *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[41] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II, in *Proc. 6th Int. Conf. Parallel Problem Solving from Nature-PPSN VI*, Paris, France, 2000, pp. 849−858.

[42] Q. Zhang and H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, *IEEE Trans. Evol. Computat.*, vol. 11, no. 6, pp. 712–731, 2007.

[43] Z. Zhan, J. Li, J. Cao, J. Zhang, H. S. Chung, and Y. Shi, Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems, *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 445–463, 2013.

[44] H. Wang, L. Jiao, and X. Yao, Two_Arch2: An improved two-archive algorithm for many-objective optimization, *IEEE Trans. Evol. Computat.*, vol. 19, no. 4, pp. 524–541, 2015.

[45] Q. Zhang, J. Liu, and X. Yao, An efficient many objective optimization algorithm with few parameters, *Swarm Evol. Computat.*, vol. 83, p. 101405, 2023.

[46] A. Zhou, B. Qu, H. Li, S. Zhao, P. N. Suganthan, and Q. Zhang, Multiobjective evolutionary algorithms: A survey of the state of the art, *Swarm Evol. Computat.*, vol. 1, no. 1, pp. 32–49, 2011.

[47] J. Togelius, M. Preuss, N. Beume, S. Wessing, J. Hagelbäck, and G. N. Yannakakis, Multiobjective exploration of the StarCraft map space, in *Proc. 2010 IEEE Conf. Computational Intelligence and Games*, Copenhagen, Denmark, 2010, pp. 265−272.

[48] J. Togelius, M. Preuss, N. Beume, S. Wessing, J. Hagelbäck, G. N. Yannakakis, and C. Grappiolo, Controllable procedural map generation via multiobjective evolution, *Genet. Program. Evolvable Mach.*, vol. 14, no. 2, pp. 245–277, 2013.

[49] R. Lara-Cabrera, C. Cotta, and A. J. Fernández-Leiva, A self-adaptive evolutionary approach to the evolution of aesthetic maps for a RTS game, in *Proc. 2014 IEEE Congress on Evolutionary Computation*, Beijing, China, 2014, pp. 298−304.

[50] R. Lara-Cabrera, C. Cotta, and A. J. Fernández-Leiva, On balance and dynamism in procedural content generation with self-adaptive evolutionary algorithms, *Nat. Comput.*, vol. 13, no. 2, pp. 157–168, 2014.

[51] D. Loiacono, L. Cardamone, and P. L. Lanzi, Automatic track generation for high-end racing games using evolutionary computation, *IEEE Trans. Computat. Intell. AI Games*, vol. 3, no. 3, pp. 245–259, 2011.

[52] L. Ma, S. Cheng, M. Shi, and Y. Guo, Angle-based multi-objective evolutionary algorithm based on pruning-power indicator for game map generation, *IEEE Trans. Emerg. Top. Computat. Intell.*, vol. 6, no. 2, pp. 341–354, 2022.

[53] E. Zitzler and L. Thiele, Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach, *IEEE Trans. Evol. Computat.*, vol. 3, no. 4, pp. 257–271, 1999.

[54] C. A. C. Coello and M. R. Sierra, A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm, in *Proc. MICAI 2004: Advances in Artificial Intelligence: Third Mexican Int. Conf. on Artificial Intelligence*, Mexico City, Mexico, 2004, pp. 688−697.

[55] Q. Zhang, Y. Li, Y. Lin, H. Wang, and J. Liu, Interpreting multi-objective evolutionary algorithms via sokoban level generation, in *Proc. 2024 IEEE Conf. Games*, Milan, Italy, 2024, pp. 1−2.

[56] A. Khalifa and J. Togelius, Multi-objective level generator generation with Marahel, in *Proc. 15th Int. Conf. the Foundations of Digital Games*, Bugibba, Malta, 2020, p. 104.

[57] K. Suri, Off-policy evolutionary reinforcement learning with maximum mutations, in *Proc. 21st Int. Conf. Autonomous Agents and Multiagent Systems*, Virtual Event, 2022, pp. 1237−1245.

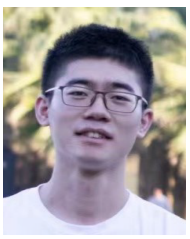
[58] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. Evol. Computat.*, vol. 6, no. 2, pp. 182–197, 2002.

[59] K. Deb and H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints, *IEEE Trans. Evol. Computat.*, vol. 18, no. 4, pp. 577–601, 2014.

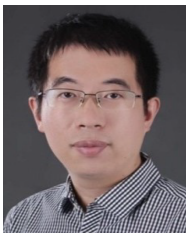
[60] K. Li, R. Chen, G. Fu, and X. Yao, Two-archive evolutionary algorithm for constrained multiobjective

optimization, *IEEE Trans. Evol. Computat.*, vol. 23, no. 2, pp. 303–315, 2019.
[61] A. Liapis, G. N. Yannakakis, and J. Togelius, Computational game creativity, in *Proc. 5th Int. Conf. Computational Creativity*, Ljubljana, Slovenia, https://www.um.edu.mt/library/oar/handle/123456789/29473, 2014.
[62] S. Karakovskiy and J. Togelius, The Mario AI benchmark and competitions, *IEEE Trans. Computat. Intell. AI Games*, vol. 4, no. 1, pp. 55–67, 2012.
[63] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in *Proc. 35th Int. Conf. Machine Learning*, Stockholm, Sweden, 2018, pp. 1861−1870.
[64] J. Ren, Y. Li, Z. Ding, W. Pan, and H. Dong, Probabilistic mixture-of-experts for efficient deep reinforcement learning, arXiv preprint arXiv: 2104.09122, 2021.
[65] J. Parker-Holder, A. Pacchiano, K. M. Choromanski, and S. J. Roberts, Effective diversity in population based reinforcement learning, in *Proc. 34th Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2020, pp. 18050−18062.
[66] K. Lee, M. Laskin, A. Srinivas, and P. Abbeel, SUNRISE: A simple unified framework for ensemble learning in deep reinforcement learning, in *Proc. 38th Int. Conf. Machine Learning*, Virtual Event, 2021, pp. 6131−6141.
[67] V. Volz, J. Schrum, J. Liu, S. M. Lucas, A. Smith, and S. Risi, Evolving Mario levels in the latent space of a deep convolutional generative adversarial network, in *Proc. Genetic and Evolutionary Computation Conf.*, Kyoto, Japan, 2018, pp. 221−228.
[68] I. Das, On characterizing the "knee" of the Pareto curve based on normal-boundary intersection, *Structural Optimization*, vol. 18, no. 2, pp. 107–115, 1999.
[69] Q. Zhang, J. Liu, Z. Zhang, J. Wen, B. Mao, and X. Yao, Mitigating unfairness via evolutionary multiobjective ensemble learning, *IEEE Trans. Evol. Computat.*, vol. 27, no. 4, pp. 848–862, 2023.
[70] M. Li and X. Yao, Quality evaluation of solution sets in multiobjective optimisation: A survey, *ACM Comput. Surveys*, vol. 52, no. 2, p. 26, 2020.
[71] D. A. Van Veldhuizen, *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. Wright-Patterson AFO, OH, USA: Air Force Institute of Technology, 1999.
[72] H. Wang, Y. Jin, and X. Yao, Diversity assessment in many-objective optimization, *IEEE Trans. Cybern.*, vol. 47, no. 6, pp. 1510–1522, 2017.
[73] J. R. Schott, Fault tolerant design using single and multicriteria genetic algorithm optimization, Master dissertation, Massachusetts Institute of Technology, Boston, MA, USA, 1995.
[74] Y. Ma, Z. Zhang, R. Cheng, Y. Jin, and K. C. Tan, ParetoLens: A visual analytics framework for exploring solution sets of multi-objective evolutionary algorithms [Application Notes], *IEEE Computat. Intell. Mag.*, vol. 20, no. 1, pp. 78–94, 2025.
[75] L. Ma, N. Li, Y. Guo, X. Wang, S. Yang, M. Huang, and H. Zhang, Learning to optimize: Reference vector reinforcement learning adaption to constrained many-objective optimization of industrial copper burdening system, *IEEE Trans. Cybern.*, vol. 52, no. 12, pp. 12698–12711, 2022.
[76] L. Ma, M. Huang, S. Yang, R. Wang, and X. Wang, An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization, *IEEE Trans. Cybern.*, vol. 52, no. 7, pp. 6684–6696, 2022.
[77] W. Li, Y. Chen, Y. Dong, and Y. Huang, A solution potential-based adaptation reference vector evolutionary algorithm for many-objective optimization, *Swarm Evol. Computat.*, vol. 84, p. 101451, 2024.
[78] X. Xue, G. Mao, S. Kumari, and Z. Liu, Multi-objective genetic programming assisted stochastic deep reinforcement learning for dynamic knowledge integration in transportation networks, *IEEE Trans. Intell. Transp. Syst.*, doi: 10.1109/TITS.2025.3545572.
[79] D. Gravina, A. Khalifa, A. Liapis, J. Togelius, and G. N. Yannakakis, Procedural content generation through quality diversity, in *Proc. 2019 IEEE Conf. Games*, London, UK, 2019, pp. 1−8.
[80] J. B. Mouret and J. Clune, Illuminating search spaces by mapping elites, arXiv preprint arXiv: 1504.04909, 2015.
[81] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, A survey and critique of multiagent deep reinforcement learning, *Auton. Agent. Multi Agent Syst.*, vol. 33, no. 6, pp. 750–797, 2019.
[82] L. Zhao, L. Li, Z. Tan, A. Hawbani, Q. He, and Z. Liu, Multiagent deep-reinforcement-learning-based cooperative perception and computation in VEC, *IEEE Internet Things J.*, vol. 12, no. 12, pp. 21350–21363, 2025.

Yi Yuan received the BS degree in physical geography & resources and environment and the BEng degree in data science and big data technology from the Beijing Forestry University, Beijing, China, in 2021 and 2023, respectively. She is currently a master student in electronic science and technology at Southern University of Science and Technology (SUSTech), Shenzhen, China. Her research interests include procedural content generation, reinforcement learning, and multi-objective optimization.

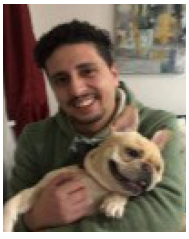


Qingquan Zhang received the BEng degree in intelligent science and technology from the Xidian University, Xi'an, China, in 2019, and the MEng degree in electronics science and technology from SUSTech, China, in 2022. He is currently a PhD candidate at SUSTech, Shenzhen, China. His research interests include fair machine learning, procedural content generation, and multi-objective optimisation.



Bo Yuan received the BEng and PhD degrees in electronic science and technology from University of Science and Technology of China, Hefei, China, in 2009 and 2014 respectively. He is currently an assistant professor at Department of Computer Science and Engineering, SUSTech, Shenzhen, China. His current research interests include computational intelligence, reinforcement learning, and electronic design automation.

Matthew Barthet received the BEng degree in computer science and the MEng degree in digital games from University of Malta, Malta, in 2019 and 2021, respectively. He is currently a PhD candidate at University of Malta, researching training reinforcement learning agents in affective computing applications. He has published 13 papers in international conferences, workshops, and journals, centered around affective computing, procedural content generation, game artificial intelligence, and computational creativity.



Ahmed Khalifa received the PhD degree from New York University, NY, USA, in 2020. He is currently an AI researcher at Institute of Digital Games, University of Malta, Malta. He has published more than 60 papers in workshops, conferences, and journals. His work focuses on exploring different methods and techniques for generating game content. He is known for his research on Procedural Content Generation via Reinforcement Learning (PCGRL), Procedural Content Generation (PCG) with quality diversity, and deep tingle. He is also an independent game designer/developer with more than 40 released games/prototypes. Some of his games were nominated for multiple awards in different conferences, such as IndiePrize, Melbourne Queer Games Festival, Queer Games Conference, and International Mobile Game Awards.

Georgios N. Yannakakis received the PhD degree in informatics from University of Edinburgh, UK, in 2006. He is currently a professor at Institute of Digital Games, University of Malta, Malta, and a co-founder of modl.ai and humanfeedback.ai. He does research at the crossroads of artificial intelligence, games, affective computing, and human-computer interaction. He has published more than 400 papers in the aforementioned fields and his work has been cited broadly. He is the editor-in-chief of the *IEEE Transactions on Games*. He is an IEEE fellow.

Huanhuan Chen received the BS degree from University of Science and Technology of China (USTC), China, in 2004, and PhD degree, sponsored by Dorothy Hodgkin Postgraduate Award, in computer science from University of Birmingham, Birmingham, UK, in 2008. He is currently a professor at School of Computer Science and Technology, USTC, China. His PhD thesis received the 2011 IEEE Computational Intelligence Society Outstanding PhD Dissertation Award (the only winner) and 2009 CPHC/British Computer Society Distinguished Dissertations Award (the runner up). He received the *IEEE Transactions on Neural Networks* Outstanding 2009 Paper Award (bestowed in 2012, and only one paper in 2009 received this award). He received the International Neural Network Society Young Investigator Award in 2015. His research interests include computational intelligence, statistical machine learning, data fusion, neural networks, and Bayesian inference. He is an IEEE fellow.

Jialin Liu received the PhD degree in computer science from Université Paris-Saclay, France, in 2016, the MEng degree in bioinformatics and biostatistics from the École Polytechnique and the Université Paris-Sud, France, in 2013, and the BEng degree in optoelectronic information science and engineering from the Huazhong University of Science and Technology, China, in 2010. She is currently an associate professor at Lingnan University, Hong Kong, China. Her research interests include artificial intelligence in games, evolutionary computation, and fair machine learning. She is an associate editor of *IEEE Transactions on Evolutionary Computation*, *IEEE Transactions on Artificial Intelligence*, and *IEEE Transactions on Games and Knowledge-based Systems*.