Use All Your Skills, Not Only The Most Popular Ones

Francesco Lorenzo[1,2], Sahar Asadi[1], Alice Karnsund[1], Lele Cao[1], Tianze Wang[2], Amir H. Payberah[2]
[1]King Digital Entertainment, Sweden
[2]KTH Royal Institute of Technology, Sweden
[1]{francesco.lorenzo,sahar.asadi,alice.karnsund,lele.cao}@king.com [2]{tianzew,payberah}@kth.se

***Abstract*—Reinforcement Learning (RL) has shown promising results across various domains. However, applying it to develop gameplaying agents is challenging due to sparsity of extrinsic rewards, where agents get rewards from the environments only at the end of game levels. Previous works have shown that using *intrinsic rewards* is an effective way to deal with such cases. Intrinsic rewards allow to incorporate basic skills in agent policies to better generalize over various game levels. In a gameplay, it is common that certain actions (skills) are observed more often than others, which leads to a biased selection of actions. This problem boils down to a normalization issue in formulating the skill-based reward function. In this paper, we propose a novel solution to this problem by taking into account the frequency of all skills in the reward function. We show that our method improves the performance of agents by enabling them to select effective skills up to 2.5 times more frequently than that of the state-of-the-art in the context of the match-3 game Candy Crush Friends Saga.**

***Index Terms*—Reinforcement Learning, Deep Q-Network, Intrinsic Rewards, Skill-based Rewards, Candy Crush Friends Saga.**

I. Introduction

Reinforcement Learning (RL) is a promising approach that enables game developers to overcome many challenges in testing and releasing new contents. For example, Ariyurek et al. in [1] use RL agents to discover bugs in different games, and Borovikov et al. [2], [3] propose a learning and planning framework to develop agents for testing new games.

However, developing RL agents for *match-3 games*, such as Candy Crush Friends Saga (CCFS)[1], poses a number of challenges. Firstly, these games present more stochastic transitions compared to popular benchmarks like the Arcade Learning Environment (ALE) [4]. This makes the learning process slower and less suited for *sparse rewards*, where agents get the rewards (either positive or negative) only at the end of each level. Secondly, unlike most games in the ALE, match-3 games present multiple levels, each one with a different topology and objective to complete in order to win. Therefore, due to lack of generalization, most of the existing RL solutions, in which agents are trained and tested on the same environment, fail to replicate the same level of performance in new environments with different levels and variations [5].

Karnsund [6] and Fischer [7] tackle the problem of sparsity by defining *dense reward* functions that progressively reward agents the more they get closer to achieving the objective of a level. However, they show that such goal oriented rewards can lead to a poor performance, where, in order to win, agents should first prioritize some other sub-goals before focusing on completing the objective of a level. To address the challenge of generalization, Shin et al. [8] propose to use the concept of *strategies* to teach agents more versatile behaviours that work across levels. Strategies refer to common salient play-styles recognized from human play, and in each strategy a set of *skills* is utilized to reach the goal. Nevertheless, they achieve this by manually defining strategies through heuristics, introducing human bias and limiting the capabilities of agents.

To overcome the two aforementioned problems we propose a solution that enables agents to learn a set of skills to pursue different strategies. Although these skills should be effective, they may be independent of the objective of game levels. Therefore, instead of defining the skills through heuristics, we use RL to learn them, relying on *intrinsic rewards*, where an agent rewards itself for completing sub-goals (e.g., performing skills) that can be different from the goal of the environment.

Designing a function that rewards agents for performing multiple different skills, whilst being simple enough to avoid skill bias, is problematic. Theoretically, the total reward given for completing any strategy should be the same to ensure that agents do not exploit only the ones that are easier to pursue. This is possible if we know the maximum number of times a skill can be used in each strategy in advance. For instance, if the strategy is to destroy all elements of a specific type, then if we knew how many of those elements are available at most in a level, we could easily normalize the rewards by dividing by the total. However, for many skills like creating a specific element on a game board, this can not be reliably predicted in advance, hence, we can not distribute the reward uniformly among the skills.

To resolve this challenge, Justesen et al. [9] propose to weigh skills inversely proportional to their frequency of occurrence. Therefore, the skills that are used less frequently will get an equal amount of rewards. However, in environments where skills are used in very different frequencies, this approach can generate rewards on a scale of multiple orders of magnitude, making training unstable and difficult to scale across different levels [10].

In this work, we propose a method that, unlike [9], takes into account the proportion of occurrence of each skill with respect to all the others to constrain the weight given to each of them in the reward function. This makes the rewards less affected

[1]https://king.com/game/candycrushfriends

978-1-7281-4533-4/20/$31.00 ©2020 IEEE

by the order of magnitude of the frequencies. We conduct a set of experiments on CCFS that presents unbalanced skill frequencies, and show that our method enables agents to use effective skills up to 2.5 times more frequently than the one presented in [9]. Moreover, we show that our agents, which are only trained using intrinsic rewards without being aware of the objective of a level, achieve a win rate comparable to that of agents trained on achieving the goal of a level.

II. Preliminaries

CCFS is a match-3 game characterized by a 9×9 board filled with colored *candies*. The basic action in the game, called *match*, consists of swapping the place of two candies such that three or more of the same color are aligned horizontally or vertically after the swap. These are then eliminated from the board, and replaced by other candies generated randomly. If more than three candies of the same color are matched, a *special candy* is created according to the number of involved candies in the match. There are six types of special candies[2] with different effects.

Overall there are five different objectives[3] in the game, and each level is associated with one of them. Players win a level if they reach its objective within the level specific move limits. In this paper, we consider *spread the jam* as the objective of the game that requires players to cover the entire board with jam. The jam is initially present in only a few tiles, and is spread by making matches involving tiles that already have it. Special candies allow spreading the jam more easily, since they affect multiple candies at the same time.

The *environment* in our study is episodic, where each *episode* corresponds to a full gameplay on a level. The *state space* of the environment consists of a three-dimensional representation of the board (i.e., $9 \times 9 \times 32$), where the third dimension is a one-hot encoding of any of the 32 different elements available in the levels we use here, each associated to a binary layer [11]. Inspired by [11], we define an *action* as a swapping between any two cells on the game board. If we uniquely index the edges between the tiles as the labels of the actions, then for a 9×9 board, the *action space* consists of 144 actions.

As the *baseline reward*, we consider *progressive jam* [6], which is an *extrinsic reward*, i.e., it is given by the environment to agents. The reward R for taking an action a_t in a state $\mathbf{s}_t$ at episode t is defined as:

$$R(\mathbf{s_t}, a_t) = \begin{cases} \frac{J}{B}, & \Delta j > 0 \\ 0, & \Delta j = 0 \end{cases} \tag{1}$$

where B is the size of the board (i.e., 9×9), while J and Δj denote the total number of tiles, and the number of new tiles covered by jam after the action a_t, respectively.

III. Methodology

Creating special candies is one of the most effective strategies to win a level, independently of the objective of the level. In this work, we identify the action of creating special candies as a *skill*, and define a set of six skills $X = \{x_1, x_2, \cdots, x_6\}$, each representing the creation of one of the six special candies. Our goal is to train an RL agent to use these skills more frequently. Some special candies are easier to create than the others. For instance, a special candy like a *fish*, which is created by matching four candies in a squared shape, is easier to create than a *coloring candy*, which requires at least six candies involved. This unbalance prevents us from rewarding agents with the number of special candies they create, because agents, then, would focus on creating the simpler ones, which in turn have weaker effects.

To tackle this issue, we propose a solution inspired by the *Rarity of Events (RoE)* method [9]. RoE consists of a reward function where skills, referred to as events, are weighted based on the inverse of their frequency of occurrence. Thus, the skills that are used less frequently are rewarded more. In formulas:

$$R(\mathbf{s}_t, a_t) = \sum_{x \in X} c_t^{(x)} \times \left[\frac{1}{\max\left(\tau, \mu_t^{(x)}\right)} \right], \tag{2}$$

x is the skill (i.e., creating one type of special candy), $\mu_t^{(x)}$ is the *mean episodic frequency* of skill x at episode t, which is the average of using x over the last k episodes, where k is a hyperparameter, $c_t^{(x)}$ is the number of special candies of type x created by a_t, and τ is a hyperparameter that represents the initial value of the frequency of each skill, used when no previous data is available (cold start problem), and also identifies the scale of the weights by setting a higher bound.

The idea of Equation 2 is to reward agents for exploring new parts of the environments, giving smaller rewards to skills that have already been observed. Even though RoE is designed to have an expected cumulative reward for each skill of one, if the mean episodic frequency of a skill is smaller than one, its weight, and thus immediate reward, will be higher than one. In fact, if the environment presents some skills that are seldom used in the last k episodes, their mean episodic frequency will be smaller than one, which leads to rewards with potentially different orders of magnitude, making gradient updates highly unstable.

We propose to normalize the reward function by taking into account the proportion of occurrence of a skill with respect to all the others, such that the weights will always be smaller than one. Using the same notation of Equation 2, we define the reward function as:

$$R(\mathbf{s}_t, a_t) = \sum_{x \in X} c_t^{(x)} \times \left(1 - \frac{\mu_t^{(x)}}{\sum_{x' \in X} \mu_t^{(x')}}\right). \tag{3}$$

As in RoE, our method, called *Balanced Rarity of Events (B-RoE)*, does not reward agents for winning a level or spreading jam, but it does so for using novel skills. However, unlike RoE, B-RoE does not suffer from the cold start problem, as all weights are initially set to one. Moreover, it does not require an hyperparameter like τ, as the upper bound for the weights is one. Finally, B-RoE is robust in environments with very unbalanced skills, as the frequency is taken in relative terms.

[2]https://candycrushfriends.fandom.com/wiki/Special_Candy

[3]https://candycrushfriends.fandom.com/wiki/Levels

IV. EXPERIMENTS

We compare the performance of B-RoE to two benchmarks: an agent trained (i) using the extrinsic reward (i.e., progressive jam), and (ii) using standard RoE. In the experiments, we use three levels in the training set, called A, B and C [4], and three levels in the test set, called X, Y and Z. For each agent and training level pair, we run three training trials for $80K$ episodes each, and test them for $10K$ episodes on the test levels. Then, we average the results of the trials to account for the variance in the game. Each trial uses a different random seed value in the libraries adopted. Due to the lack of space, we only show the graphs of the training performances for level A, but we report the results of each agent for all the test levels, aggregated over the three training levels and the three trials.

We define three metrics to evaluate the models: (i) *Win rate*: the total number of wins over the number of test episodes [5], (ii) *Match-3 probability*: the probability that an action does not create any special candies to summarize the overall progress in learning the skills, and (iii) *Creation probability*: the probability that an action creates a specific special candy. To measure it, we track the frequency of using each skill as the number of times it is used in an episode normalized by the number of steps in that episode. We report the aggregated value for all special candies together.

Without loss of generality, in our implementation we use vanilla Deep Q-Network (DQN) [10] as the learning algorithm for all the agents. However, our method can be easily applied to any other RL algorithms. We adopt the DQN hyperparameters from [10]. The discount factor, the exploration rate, and the number of update steps for the prediction network are taken from [6] that performs the hyperparameter search using CCFS as the environment. In our experiments, the mean episodic frequency $\mu_t^{(x)}$ is estimated using a running mean over the last 500 episodes, i.e., $k = 500$.

In the rest of this section, we first show the *skill proficiency*, by illustrating the proficiency of models in using the intrinsic skills (i.e., creating special candies), and then present the *gameplay performance*, to show the performance of models on the overall game objective (i.e., spreading the jam).

A. Skill Proficiency

We first measure the overall proficiency of each model in creating special candies. Figure 1 shows the training performances of each model on level A for the match-3 action (Figure 1a), and every type of special candy (Figures 1b-1g). The special candies in Figure 1 are ordered according to the difficulty of the creation from left (the easiest) to the right (the most difficult). The agent trained with progressive jam (portrayed in orange) does not show any learning progress, and presents flat curves for all the special candies. In particular, it has the highest match-3 probability at 85% (Figure 1a), meaning that overall it is the model that uses the skills less frequently. The agent trained with RoE (portrayed in red) shows a slow learning curve for all special candies, and a match-3 probability settling around 73%. The agent trained with B-RoE (portrayed in blue) has a clear edge in creating all special candies, only having comparable performances to RoE on the coloring and wrapped candies, which are the hardest to create. Most importantly, at the end of the training, it presents a match-3 probability of 53%, meaning that the agent creates a special candy on average every other action.

[4]The level names are masked for legal reasons.

[5]For legal reasons, the reported win rate is a function of the true one.

Level	Reward type	Win Rate		Match-3 prob. (%)		Creation prob. (%)	
		Avg	StDev	Avg	StDev	Avg	StDev
X	Extrinsic	9.43	0.99	79.89	0.84	2.15	0.2
	RoE	5.19	1.05	80.96	2.70	3.52	0.61
	B-RoE	**10.64**	2.22	**57.27**	6.01	**7.73**	0.83
Y	Extrinsic	0.38	0.18	84.99	1	1.79	0.2
	RoE	0.08	0.03	85.33	1.53	2.73	0.3
	B-RoE	**0.4**	0.19	**69.62**	4.9	**5.74**	0.77
Z	Extrinsic	0.01	0.001	83.79	0.42	1.79	0.01
	RoE	0	0	83.33	1.15	2.58	0.14
	B-RoE	**0.02**	0.001	**61.44**	16.17	**6.71**	2.46

TABLE I: Win rate, aggregated match-3 probability and creation probability on the test set for all models.

The results on the test levels are reported in Table I. Similarly to the training levels, B-RoE performs better than the two other models. On all levels, the creation probability of B-RoE is respectively twice as high as RoE, and three times as high as the extrinsic reward agent. This result is even more clear by looking at the match-3 probability that confirms B-RoE is more suited for this kind of environment, independently of the level where the models are trained and the one where they are tested.

B. Gameplay Performance

Here, we evaluate the performance of the models on the overall win rates. Figure 2a shows the training performance of each model on level A. The agent trained with the extrinsic reward (orange) converges within $10K$ episodes, and presents the highest win rate. The agent trained with RoE (red) has a slower learning process, converging after $50K$ episodes, and shows a much lower win rate than the extrinsic one. The agent trained with B-RoE (blue) presents a learning curve that is not converged after the training process, and reaches the same win rate of the extrinsic agent, despite having a very different playstyle. In fact, B-RoE never rewards the agent for spreading the jam on the board, but the result of creating and using more special candies is that the final amount of jam spread is the same, as depicted in Figure 2b. This shows that there is a high correlation between special candies and game objectives, which is also confirmed by the level designers at King.

The results on the test levels are reported in Table I. On level X, which is the easiest among the three, the difference in performance is similar to the one seen on level A, with B-RoE performing slightly better than the extrinsic agent this time, and RoE clearly being far off. On the harder test levels, like Y and Z, standard RoE struggles much more, while B-RoE and the extrinsic agent show comparable performances, with the former still better than the latter.

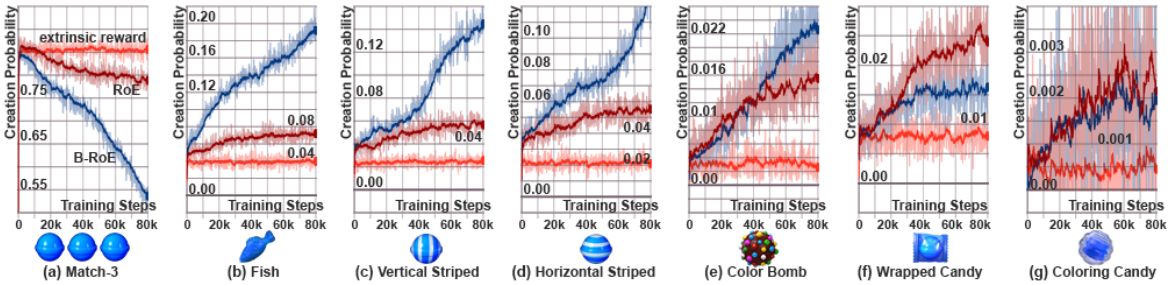
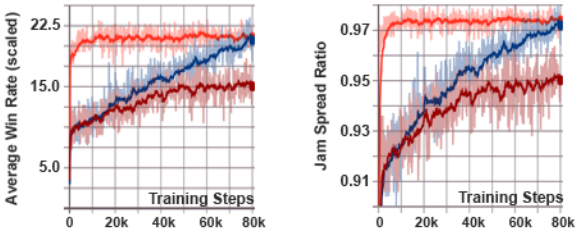


Fig. 1: Skill proficiency for extrinsic reward (orange), RoE (red), and B-RoE (blue), order by difficulty of creation.



(a) Average win rate. (b) Jam spread ratio.

Fig. 2: Gameplay performance comparison: (a) the win rate averaged over a window of 100 game episodes, and (b) the game objective – jam spread ratio.

V. Discussion and Perspectives

The main idea presented in this paper is that, in order for RL agents to generalize over a variety of levels with different objectives and game features, they should first master basic skills that can be used in different higher level tasks. To this end, we propose B-RoE, as a solution to normalize intrinsic reward functions by taking into account the frequency of all the skills of interest. We examined this on a match-3 game CCFS, and empirically show that B-RoE learns the defined skills better than agents trained with either RoE, or extrinsic rewards. Moreover, our results indicate a higher win rate when agents are trained using B-RoE.

The research presented in this paper is a work in progress and it has some limitations. Starting from the experimental setup, although we observed a consistent trend in our results, our model clearly requires more than $80K$ episodes to converge. Here, we averaged the results of each model over only three trials. We will extend this to more trials to better account for the high variance in the training process and game dynamics. Moreover, in our early analysis, we used a training set of only three levels, in which we trained the models separately. As a next step we will train the models on data from more levels, following the approaches presented in [5]. In addition, currently, CCFS consists of more than $3K$ levels, including several different game elements and objectives, and our results do not generalize to all of them.

In this work, we mainly focused on only one objective (i.e., spreading the jam), but the correlation between special candies and the win rate may vary when using levels with the other objectives available in the game. Moreover, we presented our early results for only a subset of basic skills, related to creating special candies. However, we need to conduct experiments with other relevant skills, such as killing blockers, which are another important game feature in CCFS. We will also need to explore hybrid RL architectures that allow agents to choose the best skill given the gameplay state, similar to human play.

We also need to investigate the impact of different hyper-parameters on the performance of the models. The discount factor is especially important, as B-RoE is trying to learn relations between the moves and the candies created, so lower values might turn out to be better in this case. The number of episodes considered for calculating the mean episodic frequency is also important, and we need to study its impact when we calculate it over all the episodes from the start to one where we use a finite number of newer episodes. Moreover, the models may show different performances with different learning algorithms. For example, the original RoE approach [9] uses A2C instead of DQN, so both policy-based methods and on-policy ones are interesting paths to explore.

References

[1] S. Ariyurek et al., "Automated video game testing using synthetic and human-like agents," *IEEE Transactions on Games*, 2019.

[2] I. Borovikov at al., "Towards interactive training of non-player characters in video games," *arXiv preprint arXiv:1906.00535*, 2019.

[3] I. Borovikov et al., "Winning isn't everything: Training agents to playtest modern games," in *AAAI Workshop on Reinforcement Learning in Games*, 2019.

[4] M. Bellemare et al., "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[5] K. Cobbe et al., "Quantifying generalization in reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2019, pp. 1282–1289.

[6] A. Karnsund, "DQN tackling the game of candy crush friends saga: A reinforcement learning approach," Master's thesis, KTH Royal Institute of Technology, 2019.

[7] M. Fischer, "Using reinforcement learning for games with nondeterministic state transitions," Master's thesis, KTH Royal Institute of Technology, 2019.

[8] Y. Shin et al., "Playtesting in match 3 game using strategic plays via reinforcement learning," *IEEE Access*, vol. 8, pp. 51 593–51 600, 2020.

[9] N. Justesen et al., "Automated curriculum learning by rewarding temporally rare events," in *Conference on Computational Intelligence and Games (CIG)*. IEEE, 2018, pp. 1–8.

[10] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[11] S. Gudmundsson et al., "Human-like playtesting with deep learning," in *Conference on Computational Intelligence and Games (CIG)*. IEEE, 2018, pp. 1–8.