Using Combinatorial Multi-Armed Bandits to Dynamically Update Player Models in an Experience Managed Environment

Anton Vinogradov, Brent Harrison

University of Kentucky, Lexington, KY 40506

Abstract

Designers often treat players as having static play styles, but this is shown to not necessarily be always true. This is not an issue with games that create a relatively static experience for the player but can cause problems for games that attempt to model the player and adapt themselves to better suit the player such as those with Experience Managers (ExpMs). When an ExpM makes changes to the world it necessarily biases the game environment to the better match with what it believes that the player wants. This process limits what sorts of observations that the ExpM can take and leads to problems if and when a player suddenly shifts their own preferences leading to an outdated player model that can be slow to recover. Previously it has been shown that detecting a preference shift is possible and that the Multi-Armed Bandit (MAB) framework can be used to recover the player model, but this method had limits in how much information it could gather about the player. In this paper, we offer an improvement on recovering a player model after a preference shift after one is detected by using Combinatorial MABs (CMAB). To evaluate these claims we test our method in a text-based game environment on artificial agents and find that CMABs have a significant gain in how well they can recover a model. We also validate that our artificial agents perform similarly to how humans would by testing the task on human subjects.

1. Introduction

Experience management is the study of automated systems that guide players through a more interesting and tailored experience than that which could normally be achieved. A game that features an *experience manager* (ExpM) can automatically adapt the player's experience to better serve their specific goals and play style while also balancing the wants of the author, thereby guiding the player towards an optimal gameplay path [1, 2]. An ExpM does this by observing details about the player, such as the actions that they take within the game, and extrapolating from them to make decisions about which content to serve in the future and by what means.

ExpMs often make use of player models, which are persistent models used to represent the player's internal state that the experience manager can update. These models are built over time and can be used to make more complex and long-term decisions by the ExpM, allowing for a better balance between personalizing the game to the player and ensuring the author's intent is carried out [3].

Since the player model is only an approximation of the player's internal thoughts and preferences it is not always completely accurate as it is difficult to predict how humans will behave. This can be problematic for ExpMs. When the ExpM takes actions it necessarily biases the world towards being better suited to its model of the player. At the same time, this biasing of the environment can influence the sorts of observations that the ExpM is likely to see. Take, for example, a player who has previously shown to take combat options in a game, the ExpM observes this and changes the environment to better suit this sort of play style, removing other possible types of actions and adding in more combat-focused ones. If the player suddenly shifts their preferences, which players have been shown to do [4], to prefer a more diplomatic approach, then the environment may not offer suitable affordances for the player's current preferences. While there may be some diplomacy-oriented actions available, they may be difficult to find and the majority will be focused on combat. With their choices largely limited the player may even continue to do combat actions leading the ExpM to observe the player engaging with combat and incorrectly strengthening its now outdated player model, continuing to show that the player prefers combat. For the player model to be properly updated, the player themselves would need to seek out and find appropriate content, which may be difficult and cause the player to disengage from the experience.

Because of this, many experience managers assume that player preferences remain static during gameplay. Our previous work shows that a player preference shift can be detected [5] and that by framing the problem as a Multi-Armed Bandit (MAB) it is possible to find the player's preference and quickly recover the player model, though this has only been shown to work with artificial agents [6]. These works argue that since the game environment is biased by the removal of possible methods of interaction, then one can attempt to learn a player's updated preferences by adding actions back into the environment. This is done by introducing a new form of game object called a *distraction*, an object that is used by the ExpM to gain information about the player while also minimally disrupting the game.

These distractions need to be deliberately designed to entice players that are not engaged while also being ignored by players that are engaged. This had been accomplished naturally by the limitations of adapting the problem as an MAB as MABs play rounds sequentially but only allow for a single arm pull in each round. This limits the feedback to only the single distraction added in that round but is only a limitation of the adaption, not of distractions themselves.

In this paper, we improve on this method by extending the MAB framework to a Combinatorial MAB (CMAB), which allows us to gather more information from the player and recover the player model more quickly. We make use of the CUCB algorithm [7] which allows us to use more than one distraction at a time though can potentially be more disruptive to the player due to using more distractions. We additionally create an improvement that lessens the amount of distractions needed by reusing part of the environment when it is available. Both these methods are shown to outperform previous methods in automated tests with artificial agents. We also conduct a human study to validate that

AIIDE Workshop on Intelligent Narrative Technologies, November 18, 2024, University of Kentucky Lexington, KY, USA

Anton.Vinogradov@uky.edu (A. Vinogradov); Harrison@cs.uky.edu (B. Harrison)

0009-0000-4312-4896 (A. Vinogradov); 0000-0002-1301-5928 (B. Harrison)

humans perform similarly to how these artificial agents do.

2. Related Works

Experience management is the generalization of drama management [8] to not only encompass entertainment and managing narrative drama, but also more serious contexts such as education and training, and managing a player's overall experience in the game. It does this by observing the player and manipulating the virtual world with a set of experience manager actions that allow it to modify the game and its environment to coerce the player's experience according to some goal. Early work on drama management focused on balancing the intent of the author with allowing for a breadth of actions to the player, without necessarily forming an explicit persistent player model and instead modeling it as an optimization problem [9, 10, 11, 8]. These are limited in how they can represent and understand the player but do serve to allow the player a wider breadth of actions, even repairing the narrative after the player takes an unexpected action as such killing an important NPC.

Our focus is on ExpMs that make use of a player model, as having a persistent model of the player can allow for the ExpM to make longer term decisions and allow for more intelligent actions in single [1, 12, 3] and multiplayer games [13, 14]. These works assume that the player is static in their preferences and that further observation will lead to a more accurate model of the player. This assumption has been challenged more recently [4] and the need for dynamic updates to ExpM player models has been acknowledged [15]. These approaches focus on categorizing players according to some pre-existing set of play styles.

In our previous work we have shown that player preference shifts can be detected [5] and recovered [6]. These have been shown to work in automated agents that mimic human-like behavior and utilize MAB algorithms to accomplish the player model recovery process. MABs have been used in player modeling [16, 13] and CMABs in experience management [17] before but these approaches still assume that the player is static in their preferences. We extend on this work to show that humans instructed to emulate a preference shift do act similarly to these automated agents and additionally offer an improvement on player model recovery by expanding the ExpM's actions and allowing it to pull combinatorial super arms.

3. Methods

In this section, we will start with a brief overview of CMABs and the CUCB algorithm before detailing how we adapt our environment to make use of this framework.

3.1. Combinatorial Multi-Armed Bandits

Multi-armed bandits (MABs) are single state Markov Decision Processes where agents choose to take one of several actions (called arms) [18]. Each action has a reward distribution and the task of solving comes in the form of finding an optimal policy that maximizes the total reward while minimizing losses. This policy needs to be able to balance between exploring each of the arms while gaining information on their underlying reward distribution and exploiting this information to maximize the reward from all the arms.

We make use of Combinatorial Multi-Armed Bandits (CMABs) an extension on MABs [19]. Similarly to MABs, a CMAB contains a set of m base arms that are played over some number of rounds. Each arm is associated with a random variable $X_{i,t}$ for $1 \leq i \leq m$ and $t \geq 1$ which denotes the outcome for arm i at round t. The set of random variables $\{X_{i,t} | t \geq 1\}$ associated with base arm i are iid with some unknown expected mean value μ_i. $\boldsymbol{\mu} = \{\mu_1, \mu_2, ..., \mu_i\}$ is the expected mean of all arms. Instead of playing a single arm, as one would do with a MAB, a *super arm* S is played from the set of all super arms $\boldsymbol{S}$. We consider $\boldsymbol{S}$ to be the subsets of the set of all arms m of size k, for $k \in \{2, 3\}$. At the end of the round, the reward $R_t(S)$ is revealed for the super arm and is given to the contributing arm $i \in S$. We observe the reward per arm played and only reward a single arm, a type of feedback belonging to *semi-bandits* [20]. $T_t(i)$ is the number of times arm i has been played up to round t.

To learn an optimal policy, we make use of the CUCB algorithm [19]. This algorithm first takes an initial m rounds of playing a super arm that contains one of each of the arms. For our implementation of the algorithm we choose to always play a super arm that contains the lowest played arms thus far, ensuring that each arm is played k times in this time. Afterwards we calculate the adjusted means with $\bar{\mu}_i = \hat{\mu}_i + \sqrt{\frac{3 \ln t}{2 T_t(i)}}$, and select the super arm with the highest valued adjusted mean.

3.2. CMAB Adaptation

We extend on the work we introduced in [6] and thus use a similar formulation to their MAB adaptation. One of the core innovations to adapting the player model recovery system to a MAB are *distractions*. Distractions are a type of game object that are used to gain more information from the player. These are intended to entice players that are currently not engaged with the game from their current task, but ideally would not be noticed or interacted with by players that are engaged. This way they can be used as a safe means to test whether a player is currently engaged, and more importantly whether they are taking actions that align well with the ExpM's player model. Since distractions are meant to balance this thin line of engagement they have a few extra requirements. Distractions should:

1. be largely irrelevant to important parts of the game like quests as to not tread on any authorial goals
2. represent a type of action or style of play in the game (the distractions action type)
3. be easily recognizable by the player as belonging to that action type

This way distractions can serve as a means to measure the player's preference by being clear and understandable while also not being too intrusive to normal gameplay.

These distractions are utilized with a set of ExpM actions, which we call *distraction actions*, which put into play a new distraction for the player to interact with. In terms of MABs, these distraction actions form the arm pulls, where adding a distraction with a specific type of action to the environment is a pull of an arm for that type of action. For example: if a valid action type is crafting, a distraction action for crafting would be spawning a low level crafting ingredient like a torn sack for cloth as a distraction. A player that is well engaged with the game would ignore this and continue on

with their current task, but a player that is not engaged may investigate further.

In adapting the player model recovery system to a MAB framework, previously we were were restricted to pulling a single arm at a time[6], which aligned well with the goal of reducing the number of distractions used. Since the amount of distractions introduced per round is flexible, we found that further adapting to CMAB framework is natural and allows us to gain more information per round since super arms can be played. Pulling a super arm is a distraction action that adds more than one distraction per turn and each individual arm is an individual distraction. The super arm formed from the touch and read arms would add 2 distractions, one of touch and the other of read action types. We also follow our previous method of giving a reward of 1 to the arm for the distraction that the agent has interacted with and a reward of 0 for all other arms. This means that super arms are not rewarded together, only a single arm within the super arm gets a reward. If the agent has not interacted with any distractions in that turn then no reward is given to any arms that round. This allows for more specific rewards to the action types that are interacted with compared to rewarding the super arm as a whole.

We have also previously found mixed success in purely reducing the bias of the environment by only adding a single distraction per round, selecting the distraction from those that have action types that are not in the current area. Taking that as inspiration we have developed an additional improvement: *replace-with-environment-action*. With this improvement active, a super arm that contains a distraction with an action type that is already present within the current area will have the distraction replaced with the matching environment action. Instead of adding a distraction of that overlapping type, the algorithm considers any action taken of that overlapping type in that round to count towards that distraction. This has the effect of adding some of the player's interaction with the environment to the CMAB reward model. This has the effect of reducing the amount of distractions added early on to around 1.7 per round in the first 10 turns, though as the algorithm gains more information it starts to rarely need to give an environment action and thus this decreases to an average of 1.95 after the first 10 turns.

4. Experiments and Results

In this section, we will go over both our tests with automated agents and human subjects as well as their results. While these two experiments share many features, humans require much more than artificial agents do and as such we will note where there are differences.

4.1. Automated Experiments

The goal of these experiments is to evaluate our CUCB method in a controlled situation. Thus, we follow our previous experimental outline in which we use autonomous agents rather than humans. In these experiments, we compare our CUCB methods against their best performing method ϵ-greedy.

4.1.1. Environment

Our automated agents use a smaller environment with simpler action types as the artificial agents are free to perform the same actions and explore the same rooms without getting bored or disengaging with the experience, as a human in a similar situation would. This environment is written in Inform7, a language and integrated engine used to create interactive fiction that is played with natural language syntax. Inform7 is well suited to be used with artificial agents and is easily understandable to humans.

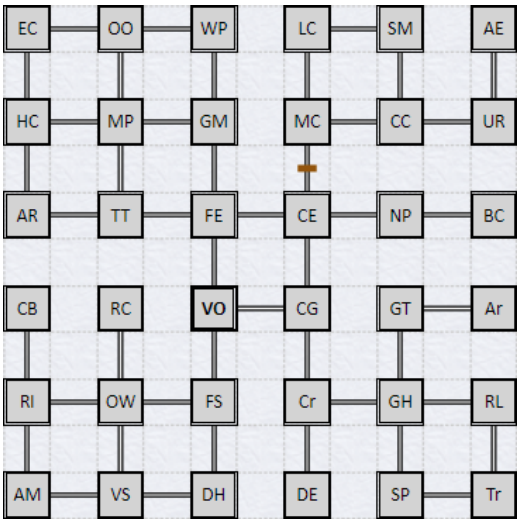

Figure 1: The expanded map for the human environment. The participant starts in the Village Outskirts (VO) and is tasked with retrieving a gem from the Secret Mining Shaft (SM) in the upper right, finding a way to bypass a group of miners blocking the path.

In this environment, we use seven areas, called rooms, which are all traversable using cardinal directions. Within these rooms are several objects that the player agents can interact with, with each object having a corresponding action type. We make use of five different action types: *look*, *talk*, *touch*, *read*, and *eat*. There is a sixth type of action, *move*, but since it is so prevalent and necessary for gameplay we do not consider this something that a player can prefer. Of these, we consider *talk* to be the Primary action type since it is the one most prevalent in the environment and is the focus of the quest that the player agent is tasked with. Other types of actions are also present within the environment but are not the focus, namely *look* and *touch*. It is often not possible to create an environment or quest in a game that only uses a single type of action so we include these to mimic that. These, along with the Primary action type, are considered Environmental action types since they are present in the environment. The last two, *read* and *eat*, are missing from the environment and thus are considered Missing action types. Our inclusion of these two types of actions that are completely missing from the environment is to simulate the effects of the ExpM previously biasing environment due to the player not preferring read or eat actions.

These five action types are also the basis for our player model, with the player agents preferring to primarily interact with one of these action types. We expect that players will have a strong preference towards one of these, but also that it is often not possible to complete quests without engaging with others. Because of this, our player agent preferences are set to primarily interact with one type of action (11/15ths of the time), but have a low chance of interacting

with the other 4 ($1/15$th of the time each).

4.1.2. Agents

To test our method in our environment we make use of the same 3 automated agents used in our previous work. These agents are known as the: *Exploration Focused Agent*, *Goal Focused Agent*, and *Novelty Focused Agent*. These agents work in different ways and are intended to attempt to resemble different aspects of how humans would play the game. Two of these, the Exploration and Goal focused agents, are inspired by the Bartle taxonomy of player types [21], representing the explorers and achievers quadrants respectively. Since there is no social aspect to our game we do not consider the other two quadrants, instead for our last agent we take inspiration from literature on user engagement that states that more novel objects are likely to increase engagement [22].

Each of these three player agents have an internal preference distribution that they use to decide which game objects to interact with, but each agent uses it slightly differently. The *Exploration Focused Agent* has a 90% chance to interact with an object that it sees in the room, with a 10% chance to wander to a different room. For all the objects that are available it randomly chooses one, drawing the probability of interaction from its preference distribution to choose which one, and if there are no objects it moves to a different room. The *Goal Focused Agent* on the other hand first chooses a type of action to interact with according to its preference distribution. If it fails to find a suitable object to interact with that is compatible with that type of action it will instead take a step towards completing the quest goal. Most actions needed to complete the quest goal are either *moving* to a different room or *talking*, the primary action type. Lastly, the *Novelty Focused Agent* puts equal importance on both the novelty of an object and its own preference distribution, and likewise first chooses a type of action to interact with before finding the object in the environment matching that action type. If there are multiple objects of that type it will prefer the one it has interacted with the least, and if there are no objects of that type it falls back on taking a step towards completing the quest goal.

4.1.3. Managers

Alongside our three agents, we tested each against five different managers. As these do not implement full experience managers we use the term manager to distinguish them. We compare against a lower baseline which takes advantage of how our player agents work by always providing 5 different types of distractions, which we call the One-of-each manager. We also compare against our previous best, ϵ-greedy. The last three of these are dedicated to testing variants on CUCB, one where a super arm consists of 2 distractions ($k = 2$), another with a super arm of 3 distractions ($k = 3$), and the last where a super arm only has 2 or fewer distractions using our replace-with-environment-action strategy ($k = 2$, rwea). All of these managers calculate the player model the same way, by measuring the frequency of types of actions that the player takes starting when the managers thinks that the player has shifted their preference.

4.1.4. Preference Scenarios

We also tested a number of preference switch scenarios for a full understanding of how our methods perform. We include 20 different scenarios, switching from primarily preferring one action type to a different action type. These are grouped together into 4 groups that represent whether the preferred action type is considered to be Environmental or Missing: Environment to Environment, Missing to Environment, Environment to Missing, and Missing to Missing. We previously identified that the most relevant scenario group for analysis is Environment to Missing so we will be focusing on it, but for completeness we include the results for all scenario groups in the appendix.

4.1.5. Results

For each combination of the 5 managers, 3 agents, and the preference switch scenarios we run 100 trials. Each trial consists of 100 turns of history that is shared between all the agents but differ for each preference switch scenario. Since this history is shared between the agents it is run with the Goal focused agent. This history consists of 90 turns in which the agent follows its initial preference, followed by 10 turns of its switched preference in which the managers is recording but not actively giving distractions. These 10 turns are used to simulate the time it would take to detect that a player preference shift has happened. At turn 100 the state of the quest is reset to allow agents to continue exhibiting their quest completing behavior and the managers starts taking distraction actions and it and the agent continue until turn 199. We compare the agent's internal preference to the managers calculated player model (which it starts to measure from 90 onwards) with the Jensen-Shannon (JS) distance. A lower value corresponds to a closer match between the two models and indicates a better result.

The results of our tests can be seen in Figure 2 focusing on what has been identified as the most realistic scenario, when the player agent switches from preferring Environmental action types to Missing ones.

4.2. Human Study

We found that our method performs better than previous methods with automated agents but has not been tested against human subjects. In this section, we will go over the modifications that we made to the environment and the distraction to make this compatible with human players. We will also detail the specifics of the human task. This study was reviewed by our University's Institutional Review Board (IRB).

4.2.1. Environment

To better accommodate the complexities of human behavior we have heavily modified the Inform7 environment for human use. We expect that, unlike agents, humans will grow bored and disengaged when attempting to traverse the same 7 rooms and the single quest for 75 turns. Thus, we have expanded the number of rooms available to the player from 7 to 36 rooms in a 6 by 6 grid. These rooms are all traversable with only cardinal directions for ease of navigation, but form a more complex map which can be seen in Figure 1. Along with the expanded amount of rooms, we have added 2 tasks for players to engage in, each more complex than the task in our automated tests, to better simulate a normal interactive fiction game.

We have also used different action types to better represent the types of actions that humans would normally

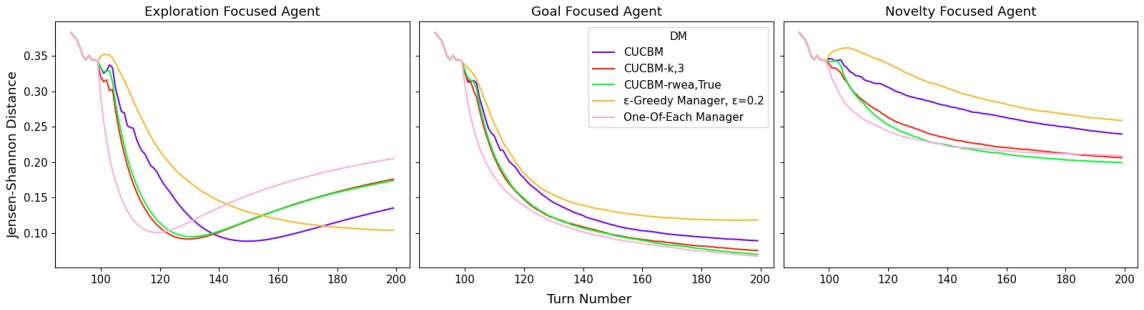

Figure 2: Mean JS Distance between Player Model and Agent Preferences vs. Turn in the Environment to Missing Scenario

engage with in games: *Diplomacy*, *Crafting*, *Combat*, *Stealth*, and *Magic*. Since we have fewer human participants than we can with automated agents we have also modified the distributions of the action types in the environment. For these tests we consider Diplomacy the Primary action type, Combat to be an additional Environmental action type, and the rest to be missing since they are not normally available in the environment.

4.2.2. Distractions

This environment also includes distractions, which are normally not present in the world but can be added to the room the player is currently in with a manager action. Since we suspected that players would be either need to be alerted to objects added to a room or otherwise they may not notice that they are added, we have opted to only add distractions when the player moves to a different room as this will align with when the room description is printed out, listing the items. These distractions are removed from the room when the player leaves the room.

This change does affect how our manager handles observing to player actions. Now the observations happen on room change and all distractions that the player has interacted with before moving onto a new room are counted as being interacted with. Distractions resolve themselves in a single turn and are subsequently removed as to only allow for the player to interact with each once. A sample of the distractions for each type can be seen in Table 1.

Distractions also were originally designed to end with a failure outcome, a monster turns out to be a bunch of branches shaped like a monster, or a deer runs away before it is hit. This is due to our focus on the distraction being irrelevant to the larger narrative. We found that preliminary participants found these discouraging and recognized that such objects would end in failure, which led to many to simply not interact with any distractions. This led us to change most distractions to resolve themselves with a small success instead, giving the player a small reward such as XP, rations, or crafting materials, though these values were never actually tracked. These changes contributed to participants continuing to interact with distractions though may have made them too enticing. For future work, we consider that these sorts of rewards need to be finely tuned and contextualized so as to not make the act of interacting with distractions too enticing to the player.

4.2.3. Task

Our goal in this task is to show that the same trends that are visible in the automated agents reflect the sorts of behaviors found in human players. To do this we set up the task to mimic the Environment to Missing scenario group, focusing only on a single type of preference shift for simplicity. To this end we recruited 30 anonymous human participants on Prolific to take part in this study, only restricting the location to English speakers in the United States. Of these, we had to remove a single user. This user was able to finish the task but the majority of actions they tried to take caused errors as they did not use the syntax necessary for Inform7 games, leaving only 22 valid actions, all but one of which was just moving around the map.

We task the human player to start with a preference for Diplomacy type actions and after playing for 20 turns to switch their preference to Crafting. We simulate that it will take some time for the manager to detect that a player is distracted so have added a 5 turn offset from when the participant is asked to switch their preference to when the manager can start taking actions. This 5 turn offset is optimistic and previous results have indicated that it should take longer [5]. Nonetheless, this is left at 5 turns so as to not frustrate the participant and to minimize the amount of time they need to spend on the task.

Afterward, the manager continues to observe the participant's actions and add new distractions for 50 more turns. Due to the way that the CUCB algorithm works the first 5 of these turns contain a super arm with one of the five arms, which we have chosen to always play the two distractions that have been least played so far, which results in each type of distraction appearing twice. After those 50 turns are over, on turn 75, we automatically end the play session.

4.2.4. Processing

In serving the Inform7 environment to humans we were presented with a number of limitations, both from the game due to using a web interpreter (Quixe, bundled with Inform7) and due to the changes that were needed to make this playable by humans. Unlike the automated tests, we no longer had access to the internal state information that reports what kind of action was made and had to classify it manually. This was done based on keyword matching, generally based on the verb used, and iteratively continuing to add keywords until all user commands were classified. We classified these into several categories, one for each of the five action types and additionally *move* for movement actions, and *none* for all the rest. The five action type categories mostly used the verbs related to the proper way of interacting with the distraction, but we also added extra keywords when the intent was clear (e.g. the invalid verbs "repair" and "craft" which are clearly an attempt to

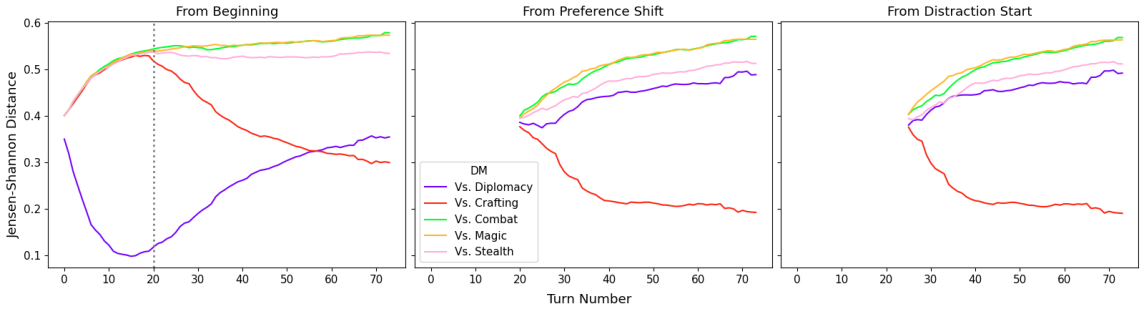

Figure 3: Mean JS Distance between Player Model and 5 Primary Preference Models.

craft). The largest of these categories is *none* and accounts for all invalid verbs due to spelling mistakes, commands without verbs, and valid verbs like "look" and "examine" which simply do not correspond to an action type. For our player model calculation, we only used the 5 action type categories.

4.2.5. Results

The results of our human experiments can be seen in Figure 3. Since we do not expect to know what the final preference is in normal gameplay, we have opted to show the JS distance of the measured player model compared to five different preference distributions. These five distributions correspond to preferring one of the five different action types and are distributed similarly to our agent's internal preferences, $11/15$ for the preferred action type and $1/15$ for the others. The results also depend on what turn we start measuring the player model. Ideally detecting a preference shift would be able to find which turn a player shifted their preference, but to cover all cases we show what the JS distance looks when starting from the beginning (20 turns before a shift), when the preference shift occurs, and when the manager starts to give distractions (5 turns after the shift).

5. Discussion

We found that CUCB outperforms the previous best of ϵ-greedy with the ability to get surprisingly close to One-Of-Each. Additionally we found that human results show similarities to the artificial agents, but that creating distractions that are well suited for human players requires careful balancing. We will discuss further implications of these experiments below.

5.1. Automated Agents

The best performing manager is the One-Of-Each manager which serves as our lower baseline, specifically taking advantage of our agent's behavior. This manager provides one of each of the five distractions on every turn, which means that all possible action types are always represented. Since our agents do not take into account how many distractions there are like humans would, this serves as an approximation to the lower limit to how quickly a player model can possibly be recovered. For this reason, we do not consider this to be a valid strategy to test on humans since it would quickly overwhelm them with options, and would require a large amount of varying distractions as to not be repetitive.

For our CUCB based managers we find that giving only two distractions at a time already provides significant benefits over our previous best, ϵ-greedy ($\epsilon = 0.2$). This is expected as giving more distractions allows us to gain more information, taking advantage of not just if the player agent interacts with a distraction, but also which of the distractions it chooses to interact with. Previously an effect was found where the distance between the agent's internal preference distribution and the measured preference distribution hits a minimum value and then starts to increase. This is attributed to the MAB gaining enough information on the preferences that it started to give the highest valued action type almost exclusively, though was often only seen in less realistic scenarios like when the agent shifts to an action type that is already well represented by the environment.

In our tests we see that this effect is also present even in the Environment to Missing scenario in the Exploration focused agent for everything but the ϵ-greedy manager. This suggests that for this agent the strategies that exhibit this effect are capable of identifying the new preference within 20 turns of the manager activating, and is likewise because one of the distractions given is almost always the preferred distraction. Other agents do not exhibit this effect, which we take to mean that they are significantly more difficult to recover the player's preferences for. For both the Goal and Novelty focused agents this is likely because they will default to environment actions when they are not given an action that they wish to interact with, thus skewing the data in favor of environment action types.

For CUCB both $k = 3$ and $k = 2$ when replacing a distraction with an environment action are capable of getting surprisingly close to One-Of-Each in all agents. Replacing a distraction with an environment action was developed to reduce the number of distractions that were shown each turn, which reduced the number of distractions from 2 to an average of 1.93, though this value is around 1.3 for the first couple turns. Later on the manager has enough information about actions in the environment that it does not need to play them as often so replacement only occurs rarely. We expected that this would result in a hit to this strategy's ability to recover the new preferences but found that instead it increased its ability. Without replacement when the agent interacts with the environment no reward is given to any of the actions in the environment. In replacing a distraction with an environment action we allow any object that the agent interacts with to count as a reward for the CMAB algorithm, which allows it to watch for more actions than before and naturally fill in action types that are underrepresented due to the biasing of the environment.

Distraction type	Distraction Action	Resolution
Combat	Attack Strange Goblin	As you approach the figure it turns out to be several branches vaguely in the shape of a goblin.
	Attack Deer	You quickly take down the deer. +**10xp**, +**3 rations**.
Crafting	Take Meteoric Iron	You have acquired +**1 iron**.
	Take cotton cloth	You have acquired +**1 cloth**.
Diplomacy	Help Hungry Beggar	You give some food to the beggar, who eats it gratefully and blesses you for your kindness. +**1xp**.
	Help Trapped Frog	You carefully free the frog from the crevice, and it hops away with a thankful croak. +**2xp**.
Magic	Pray at idol statue	Your heart is warmed by the god's blessing and you are filled with peace and courage. +**2mp**.
	Touch mystical vine	The vine's light pulses stronger, and you feel a rush of vitality. +**1hp**.
Stealth	Steal coin purse	You decide to take the whole thing +**10gp**, though you discard the coin purse itself after you extract the contents.
	Pickpocket snoozing man	The man reeks of alcohol and is absolutely conked out, but unfortunately you do not find anything of use on him.

Table 1
An example of distractions used in the human study. Distractions are designed to resolve themselves after a single interaction and to sometimes give the player a small reward. These rewards are bolded in the resolution and were displayed as bolded for the player though their values are not tracked for this experiment.

5.2. Human Study

In our results, we find that we see a pattern that is similar to our agents. Our agents (Figure 2) measure the JS distance against their own internal preference distribution, and the analogous curve in our human data (Figure 3) would correspond to *Vs. Crafting*, as that is what we instructed the participants to prefer after the switch. We expect that as soon as distractions are available (turn 25) we should see a sharp drop in the distance between the measured player model from a primarily Crafting model, while the distance to other models steadily rises starting from when the preference shift occurred (turn 20). This trend exists when measuring from the beginning, but we additionally see a small immediate drop. This is due to some players attempting to take crafting actions immediately even though the environment does not allow for it. We also expect the trend of the measured distance to eventually flatten out, which is observed in all three plots.

The behavior of users attempting to do actions that are not possible is due to how we have to manually classify what type of action a human commands. Instead of directly looking at the type of action as reported by Inform like we can for our automated tests, we instead classify actions based on the keyword that was used in the command. This allows us to classify actions that are not actually possible in the game, so it still counts as a crafting action when a participant attempts to "craft" something. We found in preliminary tests that many participants struggled to navigate the interface early in the experiment though we still wanted to capture the intent of the commands entered.

This confusion on how to navigate did not affect all users, but in response, we clarified the instructions and guided users to use the "help" command if they needed it, but we also found that the issue may have been more fundamental to the distractions. An early version of the distractions had a larger variety of interactions with each type of distraction, and we had not yet considered the importance of the distraction's action type being easily recognizable by the player. Take the Combat distractions as an example. Early formulations of these had either equipment you could pick up or monsters you could attack. These could be confusing to players as picking up items to disassemble them is recognizably a Crafting action, the only distinguishable difference between the two would be the outcome after the player has already interacted with it. The current version of Combat distractions simplifies this, always presenting animals to be hunted or threatening monsters, and always interacting by attacking. An ideal implementation of the system would allow for distractions to be independent of the action, allowing for a generic distraction with multiple types of interaction. In the future we plan on investigating how these sorts of combo-distractions can be used, especially if they can themselves be formed as a super arm, thereby allowing for using fewer distractions at a time while still gaining the same amount of information on the player's preferences.

We find that the distance measurement is sensitive to when we start measuring. When we start measuring at the beginning, 20 turns before the preference shift, we find that the extra Diplomacy moves make it more difficult to find what the preference is immediately after. Starting at later points makes the trend is easier to see, but may be throwing out relevant data. Only starting from when the manager detects a preference shift may not perform as well as having a good estimate for when the preference shift occurred.

6. Conclusion

Experience managed games help guide players through a more interesting and tailored experience, but this process of customizing the experience does not take into account the unpredictable nature of people. Player experience may degrade if this unpredictability is not taken into account and accommodated. Previously it has been shown that recovering a player model after a preference shift is possible, but only in automated agents. In this paper we further improve on this process by modeling it as a combinatorial multi-armed bandit and making use of the existing environment to supplement distractions. In addition we demonstrate that humans behave similarly to how the artificial agents behave. In the future we plan on expanding this system to combine detection of preference shifts and player model recovery after a shift has been detected.

References

[1] M. Sharma, S. Ontanón, C. R. Strong, M. Mehta, A. Ram, Towards player preference modeling for drama management in interactive stories., in: FLAIRS, 2007, pp. 571–576.

[2] D. J. Thue, Generalized experience management (2015).

[3] H. Yu, M. Riedl, Data-driven personalized drama management, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 9, 2013, pp. 191–197.

[4] J. Valls-Vargas, S. Ontanón, J. Zhu, Exploring player trace segmentation for dynamic play style prediction, in: Proceedings of the AAAI conference on artificial intelligence and interactive digital entertainment, volume 11, 2015, pp. 93–99.

[5] A. Vinogradov, B. Harrison, Detecting player preference shifts in an experience managed environment, in: International Conference on Interactive Digital Storytelling, Springer, 2023, pp. 517–531.

[6] A. Vinogradov, B. Harrison, Using multi-armed bandits to dynamically update player models in an experience managed environment, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 18, 2022, pp. 207–214.

[7] W. Chen, W. Hu, F. Li, J. Li, Y. Liu, P. Lu, Combinatorial multi-armed bandit with general reward functions, Advances in Neural Information Processing Systems 29 (2016).

[8] M. O. Riedl, A. Stern, D. Dini, J. Alderman, Dynamic experience management in virtual worlds for entertainment, education, and training, International Transactions on Systems Science and Applications, Special Issue on Agent Based Systems for Human Learning 4 (2008) 23–42.

[9] M. Mateas, A. Stern, Integrating plot, character and natural language processing in the interactive drama façade, in: Proceedings of the 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment (TIDSE-03), volume 2, 2003.

[10] A. Lamstein, M. Mateas, Search-based drama management, in: Proceedings of the AAAI-04 Workshop on Challenges in Game AI, 2004, pp. 103–107.

[11] M. J. Nelson, M. Mateas, Another look at search-based drama management., in: AAMAS (3), 2008, pp. 1293–1298.

[12] M. Sharma, S. Ontañón, M. Mehta, A. Ram, Drama management and player modeling for interactive fiction games, Computational Intelligence 26 (2010) 183–211.

[13] R. C. Gray, J. Zhu, S. Ontañón, Multiplayer modeling via multi-armed bandits, in: 2021 IEEE Conference on Games (CoG), IEEE, 2021, pp. 01–08.

[14] J. Zhu, S. Ontañón, Experience management in multiplayer games, in: 2019 IEEE Conference on Games (CoG), IEEE, 2019, pp. 1–6.

[15] R. Khoshkangini, S. Ontanón, A. Marconi, J. Zhu, Dynamically extracting play style in educational games, EUROSIS proceedings, GameOn (2018).

[16] R. C. Gray, J. Zhu, D. Arigo, E. Forman, S. Ontañón, Player modeling via multi-armed bandits, in: Proceedings of the 15th international conference on the foundations of digital games, 2020, pp. 1–8.

[17] K. Y. Kristen, M. Guzdial, N. R. Sturtevant, M. Cselinacz, C. Corfe, I. H. Lyall, C. Smith, Adventures of ai directors early in the development of nightingale, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 18, 2022, pp. 70–77.

[18] A. Slivkins, et al., Introduction to multi-armed bandits, Foundations and Trends® in Machine Learning 12 (2019) 1–286.

[19] W. Chen, Y. Wang, Y. Yuan, Combinatorial multi-armed bandit: General framework and applications, in: International conference on machine learning, PMLR, 2013, pp. 151–159.

[20] J.-Y. Audibert, S. Bubeck, G. Lugosi, Minimax policies for combinatorial prediction games, in: Proceedings of the 24th Annual Conference on Learning Theory, JMLR Workshop and Conference Proceedings, 2011, pp. 107–132.

[21] R. Bartle, Hearts, clubs, diamonds, spades: Players who suit muds, Journal of MUD research 1 (1996) 19.

[22] H. L. O'Brien, E. G. Toms, The development and evaluation of a survey to measure user engagement, Journal of the American Society for Information Science and Technology 61 (2010) 50–69.