Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-20)

Contrast Motif Discovery in Minecraft

Samaneh Saadat, Gita Sukthankar
University of Central Florida, Orlando, FL
{ssaadat, gitars}@cs.ucf.edu

Abstract

Understanding event sequences is an important aspect of game analytics, since it is relevant to many player modeling questions. This paper introduces a method for analyzing event sequences by detecting contrasting motifs; the aim is to discover subsequences that are significantly more similar to one set of sequences vs. other sets. Compared to existing methods, our technique is scalable and capable of handling long event sequences. We applied our proposed sequence mining approach to analyze player behavior in Minecraft, a multiplayer online game that supports many forms of player collaboration. As a sandbox game, it provides players with a large amount of flexibility in deciding how to complete tasks; this lack of goal-orientation makes the problem of analyzing Minecraft event sequences more challenging than event sequences from more structured games. Using our approach, we were able to discover contrast motifs for many player actions, despite variability in how different players accomplished the same tasks. Furthermore, we explored how the level of player collaboration affects the contrast motifs. Although this paper focuses on applications within Minecraft, our tool, which we have made publicly available along with our dataset, can be used on any set of game event sequences.

Introduction

Analyzing player behaviors can be beneficial for myriad purposes including improving user experience, supporting administrative tasks, and assisting social science studies (Müller et al. 2015b). Event sequences are a valuable data type for game analytics as they provide not only the frequency of events but also the temporal order in which those events occurred. The most popular unsupervised approach for analyzing player action sequences is Sequential Pattern Mining (SPM) (Makarovych et al. 2018; Kastbjerg 2011; Leece and Jhala 2014). However, these techniques are memory- and time-consuming for large datasets or long sequences (Saraf 2015). In this paper, we present a more efficient approach for finding common subsequences of a sequence group that can handle long sequences with minimal memory consumption and reasonable execution time. This paper demonstrates the usage of our technique to analyze Minecraft player action sequences.

Minecraft is a multiplayer online game, where players can explore a 3D world, mine materials, and craft tools and structures. It is a sandbox environment where players are afforded a great deal of freedom in how they interact with the game world. Most Minecraft servers are maintained by players rather than by private companies, making it an ideal laboratory for studying player behaviors and social interactions. Minecraft offers many opportunities for collaboration, including joint crafting and combat. Prior research indicates that players prefer to cooperate with players who have similar action preferences in terms of building, mining, fighting and exploring (Müller et al. 2015a).

Game events form sequences that provide valuable information about the play style and high level goals of the players. The observable events are low-level: *move*, *place block*, *consume item*, etc. High-level actions in the game world, such as exploring, mining, fighting, or building, are accomplished by performing chains of low-level actions. Since events are logged multiple times per second, the sequence of low-level game events may be long and filled with superfluous detail. Our aim is to develop an unsupervised method for detecting common subsequences across different instances of event sequences related to a group, allowing long sequences to be generalized into a few short subsequences. To that end, we developed a contrast mining approach that discovers subsequences that differentiate groups of sequences. We aim to not only discover motifs of a group of sequences but also to refine the motifs to ensure that they represent the differences between classes of sequences.

In classification, the goal is to guess the category of an object or data point based on its attribute, as opposed to contrast mining which takes the category of data points and reverse engineers the attributes that mark the data point as a member of a category. Contrast mining attempts to detect meaningful differences between groups of objects. For example, given the attributes of categories of online banking customers, contrast mining would identify dissimilar features between fraudulent and normal users. Discovering contrasts between specific groups of interest is particularly valuable in social science research (Bay and Pazzani 2001).

Existing contrast mining approaches discover contrasting sequential patterns which are costly to find for large datasets or long sequences. To the best of our knowledge, we are the first to propose a method for discovering *contrast motifs*. Motifs are less general than the patterns extracted from SPM and hence computationally faster to find. We exploit conceptual ideas from motif finding techniques that were designed for time series data and apply them to sequence data. The contributions of our research include: 1) a novel contrast motif discovery approach for discrete sequences 2) an analysis of action motifs in Minecraft 3) a study of the impact of collaboration differences on player motifs in Minecraft. To promote shared progress, we have made our code and data publicly available [1].

Given multiple groups of sequences, our algorithm, first, finds a set of candidate motifs for each group. Then, for each set of candidate motifs, the algorithm selects the subset of motifs that are significantly closer to their own group compared to other groups. These refined motifs are designated as the contrast motifs of the group.

This paper describes the usage of our contrast motif discovery algorithm to analyze Minecraft event sequences. We created event sequences for different Minecraft actions (fight, mine, explore, and build) and extracted motifs that differentiate actions from each other. Certain actions, such as fighting, yielded more contrast motifs, representing variations in player style. On the other hand, the explore action did not have any motifs that were distinctive to that class.

Moreover, players can be categorized into different groups based on style and characteristics such as collaborative vs. non-collaborative, expert vs. novice, and effective vs. ineffective. Our contrast mining approach can help us achieve a better understanding of the differences between various groups of players. We compared the behavior of *highly* collaborative and *hardly* collaborative players by examining their event sequences. Our investigation revealed that fighting is a common behavior amongst highly collaborative players while there is no behavior shared between less collaborative players.

Related Work

Sequence Mining

A sequence is an ordered list of events, where events can be represented by symbols from a specific alphabet. Pattern mining in sequences has countless applications in academia and industry including biological, purchasing, and weblog pattern mining (Liao and Chen 2013). Because of this, numerous methods have been designed to extract patterns in sequential data, including traditional sequential pattern mining (Agrawal and Srikant 1995; Pei et al. 2004), maximal sequential patterns (Luo and Chung 2005), and closed sequential patterns (Wang and Han 2004).

Sequence mining techniques have many potential applications in game analytics as they allow researchers to investigate patterns of player behavior (Makarovych et al. 2018). A combination of frequent sequence mining and clustering can be used to visualize common subsequences of player actions (Kastbjerg 2011). Kang, Kim, and Kim (2014) created sequences of user keyboard input and mouse movement and extracted the repetitive patterns within games using Lempel–Ziv–Welch (LZW) compression-based algorithms. Leece and Jhala (2014) applied sequential pattern mining in *Starcraft: Brood War* at both the micro and macro level to discover short-term and long-term patterns of player behavior.

Most prior studies in games use sequential pattern mining approaches, which are designed to find frequent exact non-contiguous subsequences. The traditional algorithms for mining sequential patterns (Agrawal and Srikant 1995; Pei et al. 2004) are appropriate for short sequences such as supermarket transactions. Consequently, traditional algorithms are ineffective for mining long sequences. Some of the traditional methods are used to process long sequences, but they require extensive run time (Lin 2003; Pei et al. 2004).

In addition, in situations where there is no clear boundaries for sequences (such as the Minecraft dataset used in this paper), sequential pattern mining approaches that allow gaps between events might blend two or more actions to create a pattern. One of the advantages of using motifs is that motifs form a continuous subsequence and hence blending is less likely.

Motif Finding Another type of pattern mining that can be applied to sequences is motif finding. Motifs are fairly short subsequences shared between multiple sequences; unlike sequential patterns, motifs are contiguous (Das and Dai 2007). This is a term borrowed from biological sciences where mutations might occur but in many cases, those mutations do not affect the functions of the genomic sequence.

There are numerous motif discovery approaches that have been developed for time series or biological sequences (Mueen et al. 2009; Bailey et al. 2009; Bailey 2011). Biological methods for finding motifs enforce constraints on the data to ensure that the discovered motifs are scientifically plausible (Das and Dai 2007). For example, methods for finding transcription factor binding site motifs extract one and only one motif for each sequence and use relatively short input sequences (Zambelli, Pesole, and Pavesi 2013). These approaches may not be well-suited for player behavior sequences that are long, may contain multiple motifs, and are unconstrained.

Since discrete sequences are the categorical analog of time series data, motif discovery approaches designed for time series can be applied to discrete sequences with minor modifications. The matrix profile approach can be utilized to discover motifs of time series. Yeh et al. (2016) extracted matrix profiles for discrete sequences but they convert the sequences to time series first using a method proposed by Rakthanmanon et al. (2012). This method converts sequences to time series by assigning an ordinal number to each symbol in the sequence; then Euclidean distance is used for measuring the distance between time series. This ordinal encoding assumes an ordinal relationship between symbols which may not exist. We developed an approach for extract-

[1]https://github.com/SamanehSaadat/ContrastMotifDiscovery

ing the matrix profile directly from discrete sequences.

Contrast Mining Understanding the differences between contrasting groups of sequences is an essential task in data mining and has numerous applications including customer behavior analysis and medical diagnosis (Bartle 1996; Wu, Li, and Chen 2019). Prior research on learning the contrast patterns across groups is primarily focused on finding contrast sequential patterns. A contrast sequential pattern is a pattern that occurs frequently in one sequence group but not in the others (Wu, Li, and Chen 2019). Contrast sequential pattern mining has the same computational limitations discussed earlier in this paper. To overcome this problem, we sought to unify the computational strengths of motif discovery with statistical testing techniques to identify contrast motifs.

Minecraft

Although Minecraft was not explicitly developed for research purposes, it has been used in many learning studies and scientific experiments (Nebel, Schneider, and Rey 2016). It is an ideal laboratory to study collaboration as the game can be modified to become more collaborative, track player activities, and manipulate team compositions (Debkowski et al. 2016; Müller et al. 2015a). Intelligent agents can be developed in Minecraft which makes Minecraft an excellent platform for studying human-agent teaming. For example, CraftAssist is an implementation of an interactive bot assistant in Minecraft (Gray et al. 2019).

Müller et al. (2015b) collected Minecraft data and studied players' actions using the frequencies of low-level events. Additionally, they constructed a classifier that predicts the high-level action from the low-level event log. To study collaboration in Minecraft, Müller et al. (2015a) defined various types of collaboration graphs such as contact, chat, and build graphs. They introduced the collaboration index as a universal metric to assess and compare the collaborativeness of players. Their study also identified predictors of collaboration in this game, including player familiarity and similarity. This paper leverages these collaboration metrics and data structures to conduct a study of how collaboration affects contrast motifs.

Dataset

Our paper uses a dataset collected by the Heapcraft project across multiple servers (Müller et al. 2015b). The dataset contains two months of data from 45 players, forming 14 person-days worth of active game-play.

The benefit of this dataset is that it provides ground truth Minecraft actions for collections of raw events. At random intervals, players were asked to specify the high-level actions they are performing: explore, mine, build, and fight. The four action types used in their data collection were inspired by player types in Bartle's study: killers, explorers, achievers, and socializers (Bartle 1996). For the fight, explore, mine, and build actions there are 37, 124, 186, and 297 data points respectively which creates a dataset of size 644.

Several of the events were excluded by Müller et al. (2015b) from the event log due to low frequency, correlation to other events, and redundancy. Moreover, move, sprint and sneak events were transformed to their corresponding distance or duration. We followed the event cleaning procedure presented by Müller et al. (2015b) except move, sprint and sneak events were also removed as distance and duration cannot be easily converted to symbols in sequences.

The original study considered the duration of each action to be two minutes centered around the time of response received. These two-minute intervals (labeled with high level actions) were used to construct our action sequence dataset. The sequence dataset of players was created by considering all the events performed by players during the data collection period. We created the event sequences by assigning a symbol to each Minecraft event and creating an ordered list of symbols for each data point. Since our method relies solely on the order of events rather than their frequencies, consecutive repetitive events are replaced by one event. For example, *aaabbcccd* is transformed to *abcd*.

Method

This section introduces our approach for finding the top c most abundant motifs for the group of sequences. Then, we discuss our procedure for refining motifs to discover the contrasting ones.

Motif Finding In order to find the initial set of candidate motifs, we use a modified *SnippetFinder* algorithm, which was designed to detect snippets in time series (Imani et al. 2018). The *SnippetFinder* algorithm uses a single time series, rather than a discrete event sequence. The main building block of this algorithm is the matrix profile, a vector of real-valued numbers representing the pairwise distance between subsequences of two sequences (Yeh et al. 2016). Since our goal is to extract common subsequences shared across multiple discrete sequences, we made modifications to the *SnippetFinder* algorithm. Our motif finding approach is presented in Algorithm 1. The set of subsequences that are produced by the algorithm are used as the candidate motifs.

The algorithm takes the list of sequences (S), desired number of motifs (c) and window size (w) as input and outputs a list (M) containing c candidate motifs. As a data preparation step, sequences shorter than the window size are removed from the list, since it is impossible that they contain motifs of length w. The list of all possible motifs (pm) is created by moving an overlapping window of size w along all sequences. Then the matrix profile of every possible motif is created. This matrix profile contains one value for every sequence which represents the distance between the possible motif and the sequence. The matrix profile length (l) is the length of the profile for each possible motif which is equal to the number of sequences. The distance between one sequence and a possible motif is the minimum value of the sequence profile (i.e. the minimum distance between the possible motif and subsequences of the sequence). In this study, LCS distance is used to measure the distance between subsequences. LCS distance ($LCS\ distance = window\ size - LCS\ score$) is based on the well-known

```
**Input:** $S$: input sequences, $c$: candidate motifs count
    and $w$: window size
**Output:** $M$: top $c$ snippets of $S$
 1: Remove short sequences from $S$
 2: $pm$ = list of all possible motifs of length $w$ in $S$
 3: $P$ = collection of profiles between $pm$ and $S$
 4: $m = size(pm)$ (i.e. number of possible motifs)
 5: $l$ = profile length
 6: $Q$ = array of length $l$ initialized with $inf$
 7: $M$ = empty array to store candidate motifs
 8: **while** $size(M) < c$ **do**
 9:     $min\_area, min\_idx, min\_motif = inf, -1, null$
10:     **for** $i = 1$ to $m$ **do**
11:         $cur\_motif = pm[i]$
12:         $e$ = element-wise min between $P[i]$ and $Q$
13:         $cur\_area = sum(e)$
14:         **if** $cur\_area < min\_area$ **then**
15:             $min\_area = cur\_area$
16:             $min\_idx = i$
17:             $min\_motif = cur\_motif$
18:         **end if**
19:     **end for**
20:     $Q$ = element-wise min between $P[min\_idx]$ and $Q$
21:     Add $min\_motif$ to $M$
22: **end while**
```

Algorithm 1: Candidate motif finding algorithm

Longest Common Subsequence (LCS) algorithm.

After creating the collection of profiles (P), the next step is to find the candidate motifs. The algorithm iteratively finds candidate motifs and continues to look for new candidates until it reaches the user-specified number of motifs. In Algorithm 1, Q is an array that stores the element-wise minimum values of the profiles related to candidate motifs so far. This array is initialized with inf values and is used in successive iterations to find candidate motifs in new sequences. In each iteration, the algorithm loops over possible motifs to determine which of them has the minimum area under the e curve. The e curve shows how close the possible motif is to the sequences for which no motif has been discovered. The algorithm stores the candidate motifs in a list (M), which is the output of the algorithm.

Running this algorithm requires $O(w^2n^2)$ time; where n is the total length of input sequences. Since w, which is the motif length, is a small constant number, it can be disregarded. Therefore, the time complexity of this algorithm is $O(n^2)$.

Motif Refinement Algorithm 1 generates a list of candidate motifs for a group of sequences. Since our goal is to discover contrasting motifs that are more similar to their group while being distant from other groups, we need to filter candidate motifs of each group. This is a subgroup discovery task: identifying interesting subgroups of objects with respect to a particular feature. A subgroup of objects is interesting when the feature values within the subgroup differ in a statistically significant way from the feature values of the other objects (Langohr et al. 2012). In this paper, we constructed interesting subgroups of motifs using a *Mann–Whitney U test*. Motifs are only selected 1) if the average distance to the sequences in their group is lower than the distances to the sequences from other groups and 2) this difference is statistically significant (i.e. the *Mann–Whitney U test* has a *p-value* lower than 0.05). Our algorithm does not assume that motifs exist in all sequences of the group, but it detects the motifs that are closest to all sequences of that group.

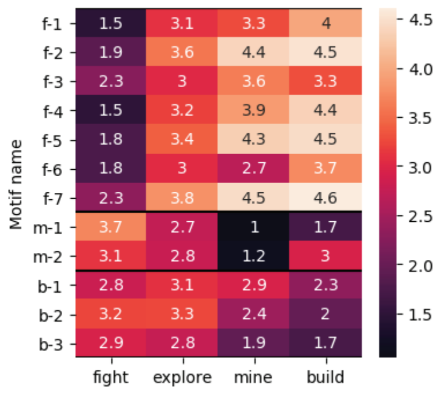

Figure 1: The average distance between motifs and sequences of actions. Rows represent motifs, and columns denote the action labels. For example, row **f-1** and column **fight** shows the average distance between motif **f-1** and **fight** sequences. Motif names are comprised of the action symbol (**f**, **m**, and **b** for fight, mine and build, respectively) and an ordinal number. Darker colors on the heatmap denote a lower distance between the motif and sequences of that action.

Minecraft Action Contrast Motifs

This section describes the application of our proposed algorithm for analyzing Minecraft action sequences. From the HeapCraft dataset, we extracted sequences labeled with the ground truth action. Each action type is considered to be a group; sequences labeled as that action belong to that group. Therefore, there are four groups of sequences for actions: *fight (f), explore (e), mine (m),* and *build (b)*.

Contrast Motif Distances We ran our contrast motif finding algorithm for window sizes ranging from 3 to 9. The number of sequences larger than the window size dramatically decreases by increasing the window size. In order to avoid discarding short sequences, a window size of 5 was selected. Using this window size, 567 sequences were considered (34, 102, 171, and 260 for fight, explore, mine, and build, respectively).

The c parameter in Algorithm 1 is a user-specified input determining the desired number of motifs. This parameter has been set experimentally in our analysis. We started from a low value for c parameter and then we increased this pa-

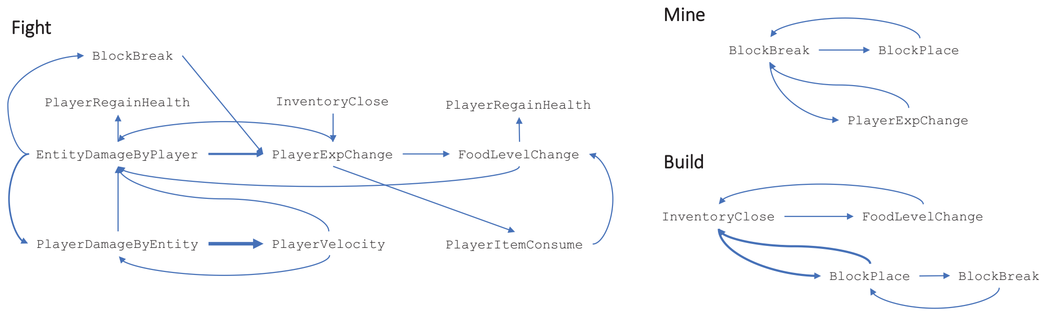

Figure 2: Contrast motifs of different actions represented in directed graphs. Nodes are events, and there is an edge between two events if they appeared consecutively in at least one motif. The thickness of edges represents the number of times that relationship was observed in the motif set.

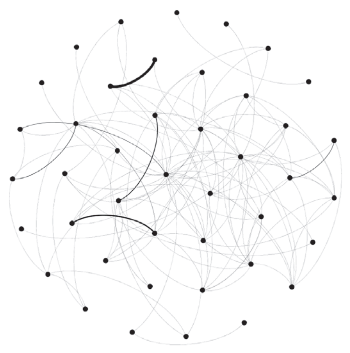

Figure 3: Collaboration graph between players. Nodes are players, edges show collaboration, and the thickness of the edge represents the duration of the collaboration.

rameter until the number of contrast motifs in each group is less than the c parameter.

Running the algorithm on the HeapCraft action sequence data with window size 5 results in 7, 2, and 3 contrast motifs for fight, mine, and build actions, respectively. Figure 1 shows the average distance between motifs and sequences of different actions. This heatmap illustrates that motifs of an action are similar to sequences of the same action class and distant from other action classes.

More contrast motifs were discovered in the fight action compared to mine and build. This may occur because there is more variety in player fighting styles compared to other actions. Unfortunately, no motif was found for the explore action. In the Müller et al. study, which attempts to classify actions, the majority of classifier errors involve the explore action. As mentioned by the authors, this could be due to the nature of the game or could be a result of their data collection procedure. In addition, Figure 1 shows that fight motifs are less likely to appear in other sequence actions. This shows that fighting is a more distinctive behavior.

The mining action has the most conservative motifs. Motifs of mine, with an average distance of 1.1, are closest to mine sequences as compared to fight and build with average distances of 1.9 and 2, respectively.

Although m-1 is closer to build sequences than some of the build motifs and can be considered a motif of build, it is not a contrast motif for the build action as it is closer in distance to mine sequences than build sequences.

Minecraft Events of Contrast Motifs To have a visual representation of the motif set for each action, we constructed a graph for each action. Figure 2 illustrates the contrast motifs discovered for various actions in a graph format.

To ensure that our results are not entirely dependent on the window size parameter, we examined the motifs generated by other window sizes. It appears that smaller or larger window sizes create motifs that are subsequences or supersequences of the motifs identified for this window size.

Our approach provides detailed insight into player Minecraft actions. Müller et al. (2015b) conducted a frequency based analysis of Minecraft actions and found that the build action is highly correlated with the frequency of BlockPlace, but our approach reveals that other events such as InventoryClose and BlockBreak also occur in the building process. Unlike prior work, our approach also extracts their temporal ordering.

Contrast Motifs of Collaborative Players

There are various types of collaboration in Minecraft including building together, sharing building/farming infrastructure, mutual protection, and practice fights to hone skills (Müller et al. 2015a). Prior research on Minecraft quantifies collaboration as the duration of the time players spend in contact with each other; two active players are considered to be in contact if their distance is less than 15 blocks (Müller et al. 2015a). Figure 3 illustrates the collaboration graph of the players in the dataset.

In the collaboration graph, nodes represent players and edges indicate collaboration. There are 42 nodes, and 123 edges in this graph, demonstrating that 3 players have no collaboration at all and there are many pairs of players who don't collaborate with each other. The collaboration graph has an average degree of 5.8, showing that each player collaborates with 6 other players on average.

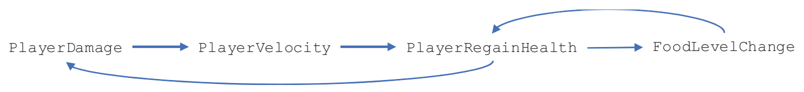

Figure 4: The highly collaborative players contrast motif. This motif is similar to the fight motif showing that fighting is the action shared amongst highly collaborative players and fighting is what distinguishes these collaborative players from less collaborative players.

To make the collaborativeness of players comparable, Müller et al. normalizes the amount of collaboration of players by the total active time of the player and calls it the collaboration index. We calculated the collaboration index for players in the dataset. The collaboration index of players in this dataset has a skewed distribution with minimum, median, and maximum of 0, 0.014, and 0.79, respectively.

We considered players to be *highly* collaborative if their collaboration index is higher than 90% of players, and *hardly* collaborative if their collaboration index is in $10th$ percentile. Collaboration index is between zero to one with a $10th$ percentile and $90th$ percentile of 0.002 and 0.19, respectively.

To compare players who are more collaborative with less collaborative players, we applied our algorithm to find contrast motifs of these two groups of players. The median length of player sequences was 2011 in our dataset. We removed players who barely played the game (with sequences shorter than 10) and conducted the experiments with remaining 41 players. We tested window sizes from 3 to 15 for the motif. A contrast motif was discovered for window size 8 for highly collaborative players, but no contrast motif was found for players with lower collaboration index. Figure 4 shows the graph visualization of the contrast motif of highly collaborative players.

Comparing the motif illustrated in Figure 4 with the action motifs of Figure 2 shows that the motif of highly collaborative players has the most intersection with the fighting behavior. In other words, the behavior that is shared between highly collaborative players is fighting. This finding is aligned with prior research that indicates a strong correlation between fighting and collaborativeness in Minecraft (Müller et al. 2015a).

Additionally, we attempted to apply PrefixSpan algorithm (Han et al. 2001), which is a classic sequential pattern mining algorithm to find sequential patterns of these two groups of players. On a computer with 16GB RAM and a 4-core 1.9 GHz CPU, the PrefixSpan algorithm ran out of memory and failed while our motif discovery approach successfully completed within two hours.

Conclusion and Future Work

By making data collection inexpensive and convenient, games such as Minecraft advance our understanding of social science. Sequence mining can assist in this endeavor by summarizing a large volume of player data into a more intuitive format. This paper presented a sequence mining approach to facilitate the analysis of players' actions and collaborative behavior in Minecraft.

We introduced a new contrast motif discovery technique and applied it to a Minecraft dataset. First, we analyzed the low-level sequences of high-level actions by extracting contrast motifs that distinguish actions from one another. The motifs of each action were visualized in a graph to facilitate the comparison between motifs of different actions. Many of the events shown in the graphs are consistent with our intuitions on how players would achieve the tasks. Some of them are aligned with prior research while others were uncovered only through the use of our algorithm.

Second, we employed our algorithm to compare highly collaborative Minecraft players with hardly collaborative ones. We created a collaboration graph across players and calculated the collaboration index of every player. Our method discovered a motif that is shared between more collaborative players but that does not occur in the sequences of less collaborative users. Comparing the graph of this motif with the action motif graphs shows that the behavior shared between highly collaborative players is fighting.

Our sequence mining approach can be used for the implementation of agents who possess theory of mind about their human teammates while also providing a glass box for social scientists to enhance their interpretations of human behavior. An assistant agent could use these motifs as part of a plan recognition system to understand what the human player is doing. Our technique can be employed by psychologists to analyze player behavior by identifying contrast motifs of various groups of players. For example, what are the motifs that occur in event sequences of expert players but not in the sequences of novice players?

Another application of contrast motifs is knowledge tracing. Knowledge tracing estimates the expertise of the player across a set of concepts or skills (Kantharaju et al. 2018). Measuring how often the player follows the motif pattern while performing an expected action could help scientists measure the players' mental models.

In future work, we are collecting our own datasets of human teams playing Minecraft. There are opportunities for improvement in the data collection process. For instance, the ground truth actions could be collected either by recording players describing their behavior using a think out loud protocol or by hand-labeling behaviors in the replay of a recorded game.

Acknowledgement

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. W911NF-20-1-0008. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA or the University of Central Florida.

References

Agrawal, R., and Srikant, R. 1995. Mining sequential patterns. In *Proceedings of the IEEE International Conference on Data Engineering*, 3–14.

Bailey, T. L.; Boden, M.; Buske, F. A.; Frith, M.; Grant, C. E.; Clementi, L.; Ren, J.; Li, W. W.; and Noble, W. S. 2009. Meme suite: tools for motif discovery and searching. *Nucleic Acids Research* 37(suppl_2):W202–W208.

Bailey, T. L. 2011. Dreme: motif discovery in transcription factor chip-seq data. *Bioinformatics* 27(12):1653–1659.

Bartle, R. 1996. Hearts, clubs, diamonds, spades: Players who suit MUDs. *Journal of MUD Research* 1(1):19.

Bay, S. D., and Pazzani, M. J. 2001. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery* 5(3):213–246.

Das, M. K., and Dai, H.-K. 2007. A survey of DNA motif finding algorithms. In *BMC Bioinformatics*, volume 8, S21. Springer.

Debkowski, D.; Marrero, A.; Yson, N.; Yin, L.; Yue, Y.; Frey, S.; and Kapadia, M. 2016. Contained: Using multiplayer online games to quantify success of collaborative group behavior. In *Artificial Intelligence and Interactive Digital Entertainment Conference*.

Gray, J.; Srinet, K.; Jernite, Y.; Yu, H.; Chen, Z.; Guo, D.; Goyal, S.; Zitnick, C. L.; and Szlam, A. 2019. CraftAssist: A framework for dialogue-enabled interactive agents. *arXiv preprint arXiv:1907.08584*.

Han, J.; Pei, J.; Mortazavi-Asl, B.; Pinto, H.; Chen, Q.; Dayal, U.; and Hsu, M. 2001. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the International Conference on Data Engineering*, 215–224.

Imani, S.; Madrid, F.; Ding, W.; Crouter, S.; and Keogh, E. 2018. Matrix profile XIII: Time series snippets: A new primitive for time series data mining. In *IEEE International Conference on Big Knowledge (ICBK)*, 382–389.

Kang, S. J.; Kim, Y. B.; and Kim, S. K. 2014. Analyzing repetitive action in game based on sequence pattern matching. *Journal of Real-time Image Processing* 9(3):523–530.

Kantharaju, P.; Alderfer, K.; Zhu, J.; Char, B.; Smith, B.; and Ontañón, S. 2018. Tracing player knowledge in a parallel programming educational game. In *Artificial Intelligence and Interactive Digital Entertainment Conference*.

Kastbjerg, E. E. 2011. Combining sequence mining and heatmaps to visualize game event flows. *Master's thesis, IT University of Copenhagen*.

Langohr, L.; Podpečan, V.; Petek, M.; Mozetič, I.; and Gruden, K. 2012. Contrast mining from interesting subgroups. In *Bisociative Knowledge Discovery*. Springer. 390–406.

Leece, M. A., and Jhala, A. 2014. Sequential pattern mining in Starcraft: Brood War for short and long-term goals. In *Artificial Intelligence and Interactive Digital Entertainment Conference*.

Liao, V. C.-C., and Chen, M.-S. 2013. Efficient mining gapped sequential patterns for motifs in biological sequences. *BMC Systems Biology* 7(S4):S7.

Lin, T. Y. 2003. Granular computing. In *International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing*, 16–24. Springer.

Luo, C., and Chung, S. M. 2005. Efficient mining of maximal sequential patterns using multiple samples. In *Proceedings of the SIAM International Conference on Data Mining*, 415–426.

Makarovych, S.; Canossa, A.; Togelius, J.; and Drachen, A. 2018. Like a DNA string: Sequence-based player profiling in Tom Clancy's The Division. In *Artificial Intelligence and Interactive Digital Entertainment Conference*. York.

Mueen, A.; Keogh, E.; Zhu, Q.; Cash, S.; and Westover, B. 2009. Exact discovery of time series motifs. In *Proceedings of the SIAM International Conference on Data Mining*, 473–484.

Müller, S.; Frey, S.; Kapadia, M.; Klingler, S.; Mann, R. P.; Solenthaler, B.; Sumner, R. W.; and Gross, M. 2015a. Heapcraft: Quantifying and predicting collaboration in Minecraft. In *Artificial Intelligence and Interactive Digital Entertainment Conference*.

Müller, S.; Kapadia, M.; Frey, S.; Klinger, S.; Mann, R. P.; Solenthaler, B.; Sumner, R. W.; and Gross, M. 2015b. Statistical analysis of player behavior in Minecraft. In *Proceedings of the International Conference on the Foundations of Digital Games*.

Nebel, S.; Schneider, S.; and Rey, G. D. 2016. Mining learning and crafting scientific experiments: A literature review on the use of Minecraft in education and research. *Journal of Educational Technology & Society* 19(2):355–366.

Pei, J.; Han, J.; Mortazavi-Asl, B.; Wang, J.; Pinto, H.; Chen, Q.; Dayal, U.; and Hsu, M.-C. 2004. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering* 16(11):1424–1440.

Rakthanmanon, T.; Campana, B.; Mueen, A.; Batista, G.; Westover, B.; Zhu, Q.; Zakaria, J.; and Keogh, E. 2012. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 262–270.

Saraf, P. 2015. Improved prefixspan algorithm for efficient processing of large data.

Wang, J., and Han, J. 2004. Bide: Efficient mining of frequent closed sequences. In *Proceedings. International Conference on Data Engineering*, 79–90. IEEE.

Wu, R.; Li, Q.; and Chen, X. 2019. Mining contrast sequential pattern based on subsequence time distribution variation with discreteness constraints. *Applied Intelligence* 49(12):4348–4360.

Yeh, C.-C. M.; Zhu, Y.; Ulanova, L.; Begum, N.; Ding, Y.; Dau, H. A.; Silva, D. F.; Mueen, A.; and Keogh, E. 2016. Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In *IEEE International Conference on Data Mining (ICDM)*, 1317–1322.

Zambelli, F.; Pesole, G.; and Pavesi, G. 2013. Motif discovery and transcription factor binding sites before and after the next-generation sequencing era. *Briefings in Bioinformatics* 14(2):225–237.