Uluslararası İleri Doğa Bilimleri ve Mühendislik Araştırmaları Dergisi
Sayı 7, S. 229-238, 6, 2023
© Telif hakkı IJANSER'e aittir
Araştırma Makalesi

IJANSER
https://as-proceeding.com/index.php/ijanser
ISSN: 2980-0811

International Journal of Advanced Natural Sciences and Engineering Researches
Volume 7, pp. 229-238, 6, 2023
Copyright © 2023 IJANSER
Research Article

Simulations of Solving a Single-Player Memory Card Game with Several Implementations of a Human-Like Thinking Computer Algorithm

Ladislav Végh [*,1] and Ondrej Takáč [1]

[1]*Department of Informatics, J. Selye University, Slovakia*

[*]*(veghl@ujs.sk) Email of the corresponding author*

(Received: 08 July 2023, Accepted: 24 July 2023)

(5th International Conference on Applied Engineering and Natural Sciences ICAENS 2023, July 10 - 12, 2023)

ATIF/REFERENCE: Végh, L. & Takáč O. (2023). Simulations of Solving a Single-Player Memory Card Game with Several Implementations of a Human-Like Thinking Computer Algorithm. *International Journal of Advanced Natural Sciences and Engineering Researches*, 7(6), 229-238.

Abstract – The memory card game is a game that probably everyone played in childhood. The game consists of n pairs of playing cards, whereas each card of a pair is identical. At the beginning of the game, the deck of cards is shuffled and laid face down. In every move of the game, the player flips over two cards. If the cards match, the pair of cards is removed from the game; otherwise, the cards are flipped back over. The game ends when all pairs of cards have been found. The game could be played by one, two, or more players. First, this paper shows an optimal algorithm for solving a single-player memory card game. In the algorithm, we defined four steps where the user needed to remember the earlier shown pairs of cards, which cards were already shown, and the locations of the revealed cards. We marked the memories related to these steps as M1, M2, M3, and M4. Next, we made some simulations as we changed the M1, M2, M3, and M4 memories from no user memory (where the player does not remember the cards or pairs of cards at all) to a perfect user memory (where the player remembers every earlier shown card or pair of cards). With every memory setting, we simulated 1000 gameplays. We recorded how many cards or pairs of cards the player would need to remember and how many moves were required to finish the game. Finally, we evaluated the recorded data, illustrated the results on graphs, and drew some conclusions.

Keywords – Simulation, Memory Card Game, Matching Game, Human-Like Thinking Computer Algorithm, Game Strategy

I. INTRODUCTION

The memory card game is a game that probably everyone knows from their childhood. The game is played with n pairs of playing cards. Each card of a pair is identical, or cards of a pair are somehow associated, e.g., one card of the pair shows an image and the other card the word (Fig. 1) [1]. At the beginning of the game, all the cards are shuffled and laid face down. The player flips over one card first, then another. If the two cards match, the pair of cards is removed from the game; otherwise, the cards are flipped back over. The game ends when all pairs of cards have been found. The game could be played by one player and two or more players [2], [3].

The memory card game has many variations. The number of pairs of cards could be different; for very young children, fewer playing cards are used (e.g., 6 pairs of cards), while for teenagers and adults, the game usually contains more playing cards (e.g., 32 pairs of cards).

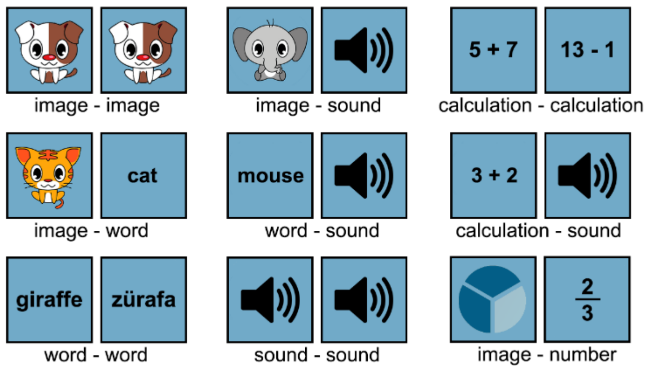

Fig. 1 Examples of types of pairs of cards

The pair of cards typically contain two identical images. However, it is possible that the two cards are not identical but somehow associated. Except for visual representations, the cards could have words, numbers, calculations, or even sounds or animations in a computer implementation of the memory card game (Fig. 1) [1].

Beyond entertainment, memory card games could be used in education as well, e.g., for learning foreign languages [4], [5], mathematics [6], or healthy life [7]. Moreover, based on a fuzzy logic approach, learners' working memory capacity could be estimated from their interactions with the educational memory card game [1]. Furthermore, research in [8] shows that playing the memory card game could prevent dementia in old age.

In the next part of this section, we deal with the optimal game strategies for single-player and two-player memory card games.

A. *Single-player game strategy*

Assuming the user has a perfect memory, the algorithm in Fig. 2 shows an optimal strategy for playing a single-player game.

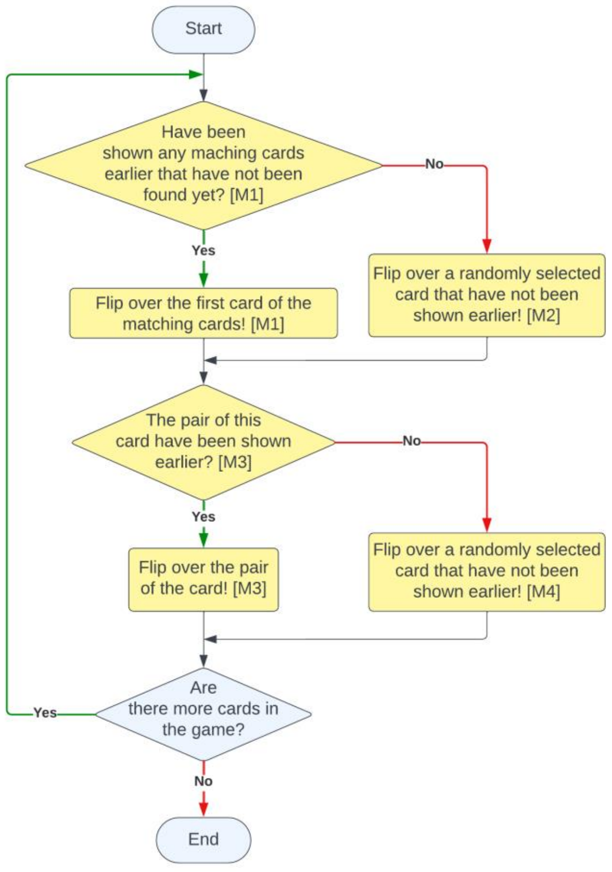
Start
Have been shown any maching cards earlier that have not been found yet? [M1]
No
Yes
Flip over the first card of the matching cards! [M1]
Flip over a randomly selected card that have not been shown earlier! [M2]
The pair of this card have been shown earlier? [M3]
No
Yes
Flip over the pair of the card! [M3]
Flip over a randomly selected card that have not been shown earlier! [M4]
Are there more cards in the game?
Yes
No
End

Fig. 2 Optimal algorithm for solving the single-player memory card game with perfect player's memory

Using the optimal game strategy, the expected number of moves for n pairs of cards is $(3 - 2 \ln 2)\, n + 7/8 - 2 \ln 2 \approx 1.61\, n$, as $n \rightarrow \infty$ [2]. Applying this formula, we can calculate that the expected number of moves for a game with 32 pairs of cards is around 51.

B. *About the two-player game strategy*

In a two-player memory card game, when a player finds a matching pair of cards, the player keeps the cards, gets 1 point, and goes again. The goal of every player is to maximize their total points, i.e., to find as many pairs of cards as possible during the gameplay [3].

The game strategy is quite different in a two-player memory card game than in a single-player game. Surprisingly, in the optimal game strategy, the players sometimes want to sacrifice their turns by intentionally not making a match. In some situations, the reason for not flipping over new cards is to prevent the opponent from gaining information. Furthermore, at some point in the game, it is possible that both players prefer such a strategy, and consequently, the game ends in a draw [3], [9].

The best way to play a two-player memory card game is described in [3] and mathematically proved in [9].

II. Materials and Method

This paper describes how we used computer simulations to get exciting observations about the optimal single-player memory card game strategy. Computer simulations are often used in education and research, e.g., where real-life experimentation is impossible because of safety issues, vast numbers of needed experiments, expensive or time-consuming experiments, or when mathematical modeling of a designed system is not possible [10]–[16].

We used MATLAB (ver. R2023a) software to simulate thousands of single-player gameplays and record how many cards or pairs of cards would be needed to remember at some gameplay points. Furthermore, we also estimated how many moves are necessary to finish the game. Because nobody has perfect memory, we also made some simulations using various scales of randomness to simulate the gameplay with weaker or stronger players' memory.

A. *Definitions of variables and notions*

In Fig. 2, which shows the optimal algorithm for solving the single-player memory card game, we marked some steps with M1, M2, M3, or M4. These marks refer to the variables related to the player's memory, i.e., to the shown cards or pairs of cards needed to remember during the gameplay.

In the step where the player chooses the first card to flip over, the M1 refers to the memory related to the pairs of cards shown earlier during the gameplay. If any matching cards were revealed earlier, the player with a perfect memory will choose one of these cards. In this step, a player with a perfect memory must remember only 0 or 1 pair of cards because if there is such a matching pair, it will be chosen to flip over. So, there will not be a situation when the player with a perfect memory needs to remember more than one pair of matching cards. However, if the player has no perfect memory, more pairs of matching cards might have been shown earlier during the gameplay, so the size of the M1 could increase. We also marked the step where the player flips over the first card (Fig. 2) with M1. The reason is that the player might remember that there are matching cards, but with no perfect memory, the player might accidentally flip over the wrong card.

If no matching cards were revealed earlier, the player with a perfect memory would try to flip over a card that has not been flipped over yet during the gameplay. In this step, the variable M2 refers to memorizing the cards shown earlier, anytime during the gameplay, but have not been found yet.

In the next step, the player must find the pair of the first card (flipped-over card). So, the player needs to remember the pairs of the already-shown cards. Variable M3 refers to these pairs of cards. A player with a perfect memory must remember only 0 or 1 pair of cards here for a similar reason that we mentioned in the description of the variable M1. However, if the player has no perfect memory, more pairs of matching cards might have been shown earlier during the gameplay, so the size of the M3 could increase. We also marked the step where the player flips over the second card with M3 (Fig. 2). The reason is that the player might remember that pair of the first card; however, with no perfect memory, the player might accidentally flip over the wrong card.

Finally, if the pair of the first card has not been revealed yet during the gameplay, the player with a perfect memory would try to flip over a card that had not been flipped over earlier. In this step, the variable M4 refers to memorizing the cards shown earlier, anytime during the gameplay, but have not been found yet.

As we can see, variables M1 and M2 are related to flipping over the first card, while similar variables M3 and M4 are connected to flipping over the second card.

In the following parts of the article, when we say M1, M2, M3, or M4 "memory," we usually refer to the cards or pairs of cards that would be needed to remember by the user, i.e., the cards or the pairs of cards that have not been found yet by the player but were already shown (flipped over) anytime during the gameplay.

When we say M1, M2, M3, or M4 "user memory" in the following parts of the paper, we try to refer to the player's memory, i.e., how the player remembers the cards and pairs. With perfect (100%) user memory, the player remembers all earlier revealed cards and pairs correctly. However, e.g., with 50% user memory, the player remembers the already shown cards or pairs correctly only in 50% of the cases; in the other 50%, the player chooses a random card instead. With no (0%) user memory, the player does not remember the already revealed cards or pairs, so the cards to flip over are randomly selected in every case.

B. Implementation and simulation of the algorithm with a perfect player's memory

Implementation of the algorithm with a perfect player's memory is based on the flow chart shown in Fig. 2. We simulated the gameplays of the single-player memory card game with 32, 16, 8, and 4 pairs of cards 1000-1000 times. During the simulations, we recorded the sizes of the M1, M2, M3, and M4 memories in every player's move. Finally, we calculated the average sizes of M1, M2, M3, and M4 memories for every move and illustrated the data on graphs.

C. Implementation and simulation of the algorithm with imperfect player's memory

Implementing the algorithm with not perfect player's memory is similar to implementing the algorithm with a perfect player's memory. We used the memory card game for these simulations only with the 32 pairs of cards. Next, we introduced randomness for every step we marked with M1, M2, M3, or M4. For the not-perfect player's memory, we calculated m1, m2, m3, and m4 logical variables with the following MATLAB codes, where the randi(10,1,1) function generates a random integer number between 1 and 10, and the mem variable value is set from 0 to 10 using an outside for loop.

```
m1 = randi(10,1,1) <= mem;
m2 = randi(10,1,1) <= mem;
m3 = randi(10,1,1) <= mem;
m4 = randi(10,1,1) <= mem;
```

The mem variable describes the player's memory (mem=0 for 0% player's memory, mem=10 for 100% player's memory).

The true values of the m1, m2, m3, or m4 variables mean that the player correctly remembers the earlier shown cards, pairs of cards, and their locations. However, the false values of the m1, m2, m3, or m4 variables mean that the player does not remember the earlier shown cards, pairs of cards, or their locations (in the simulation, we have chosen a randomly selected card in this case).

If we want to simulate the player's perfect memory with this algorithm implementation, we need to set the m1, m2, m3, and m4 variables to true values instead of using randomness. The following source code shows how it is possible to set all these variables to true value in MATLAB.

```
m1 = true;
m2 = true;
m3 = true;
m4 = true;
```

Because of using randomness, we needed to modify the algorithm in Fig. 2 slightly. The modified algorithm for solving the single-player memory card game is shown in Fig. 3.

We needed to add instructions to the steps marked in Fig. 3 with M2 and M3 that deal with the situations when all the cards were shown earlier. In these situations, the player should flip over a randomly selected card.

In the next step, we wanted to estimate, using the simulations, how the number of already shown cards (M2 and M4 memory), the number of shown pairs of cards (M1 and M3 memory), and the number of moves needed to finish the game will change, if only one of the M1, M2, M3, or M4 user memory is not perfect, but the others are perfect. For this reason, in various simulations, we set only one of the m1, m2, m3, and m4 variables to a random value but the others to the true values. Finally, in the last simulation, we set all the m1, m2, m3, and m4 variables to random values to determine how the measured data will change if all the M1, M2, M3, and M4 user memories are imperfect. With every memory setting, we simulated the gameplay of the single-player memory card game 1000 times. We recorded the sizes of the M1, M2, M3, and M4 memories in every player's move and the number of moves needed to finish the game. Finally, we calculated the average sizes of M1, M2, M3, and M4 memories for every move and illustrated the data on graphs.

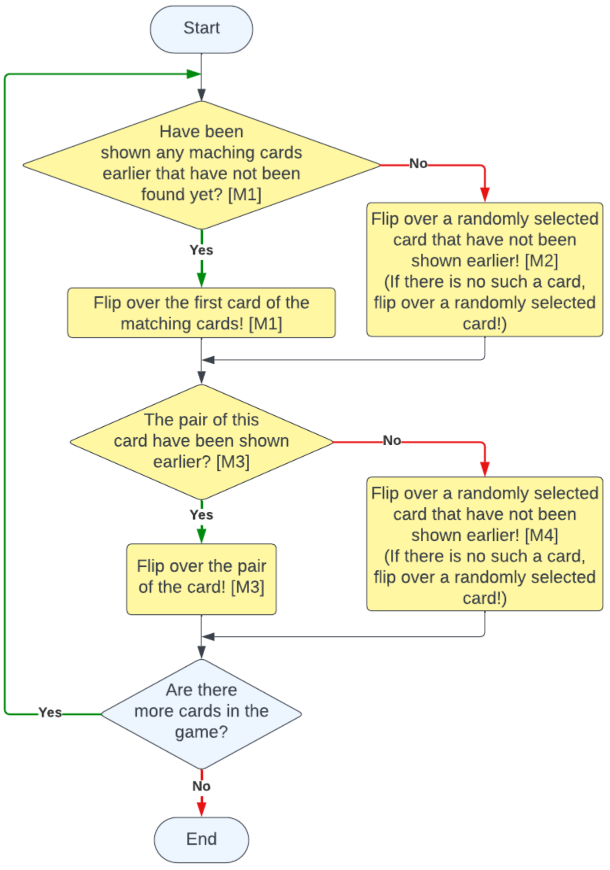
Start
Have been shown any maching cards earlier that have not been found yet? [M1]
No
Yes
Flip over a randomly selected card that have not been shown earlier! [M2] (If there is no such a card, flip over a randomly selected card!)
Flip over the first card of the matching cards! [M1]
The pair of this card have been shown earlier? [M3]
No
Yes
Flip over a randomly selected card that have not been shown earlier! [M4] (If there is no such a card, flip over a randomly selected card!)
Flip over the pair of the card! [M3]
Are there more cards in the game?
Yes
No
End

Fig. 3 Algorithm for solving the single-player memory card game with not perfect player's memory

III. Results

The results of the completed simulations are shown in Fig. 4 to Fig. 14.

First, we simulated gameplays of single-player memory card games with a perfect player's memory and four different-sized decks of cards (32 pairs, 16 pairs, 8 pairs, and 4 pairs). Fig. 4 shows how the M1, M2, M3, M4 memory usage was changed during the gameplay. As we mentioned before, the pairs of matched cards (but not found by the player) are remembered in the M1 and M3 memory, and for the player with a perfect memory, there is always only 0 or 1 such a pair of cards during the gameplay. The first graph in Fig. 4 shows that, in most cases, such a pair is remembered in the middle of the gameplay for M1 (when the player chooses the first card to flip over) and right before the end of the gameplay for M3 (when player chooses the second card to flip over). Memory M2 and M4 refer to the number of earlier shown cards the player needs to remember during the gameplay. The M2 memory is observed when the player chooses the first card to flip over, and the M4 memory is examined when the player chooses the second card to flip over. As we can see in the second and fourth graphs of Fig. 4, the most shown cards are remembered in the middle of the gameplays: 16–17 cards in a game with 32 pairs of cards, 8–9 cards in a game with 16 pairs of cards, 4–5 cards in a game with 8 pairs of cards, and 2–3 cards in a game with 4 pairs of cards. The game ends after 51.1 moves on average (game with 32 pairs of cards), after 25.3 moves on average (game with 16 pairs of cards), after 12.4 moves on average (game with 8 pairs of cards), or after 5.9 moves on average (game with 4 pairs of cards).

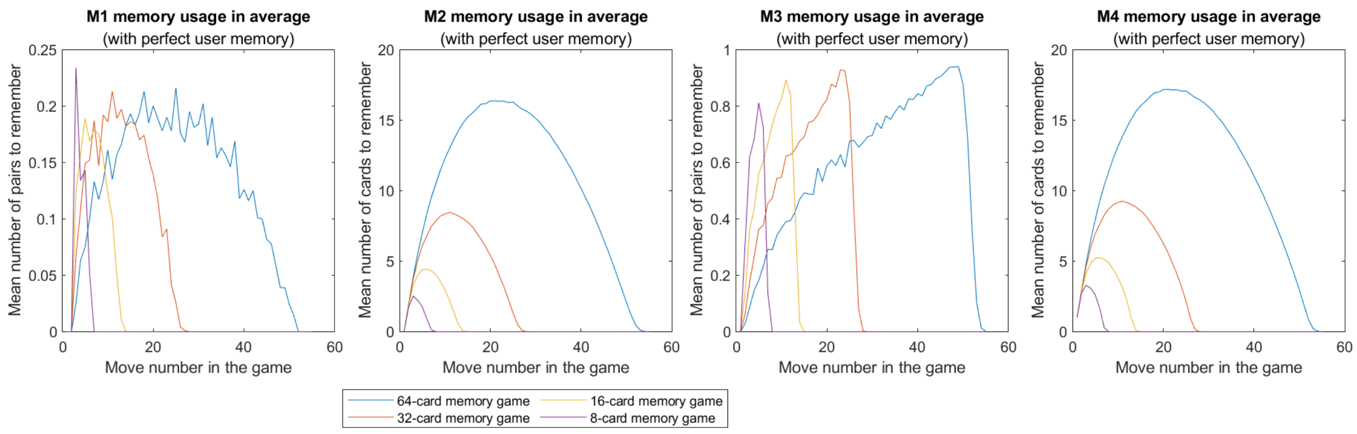

Fig. 4 Mean number of cards or pairs to remember in M1–M4 memories during the gameplays with perfect M1–M4 user memories (game with 32, 16, 8, and 4 pairs of cards)

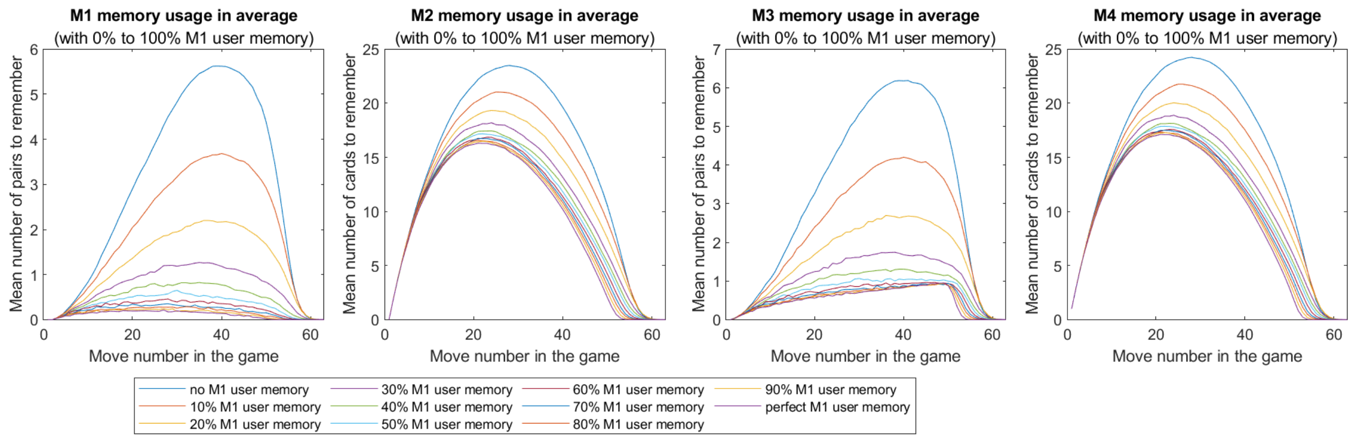

Fig. 5 Mean number of cards or pairs to remember in M1–M4 memories during the gameplays with various M1 user memory and perfect M2, M3, and M4 user memories (game with 32 pairs of cards)

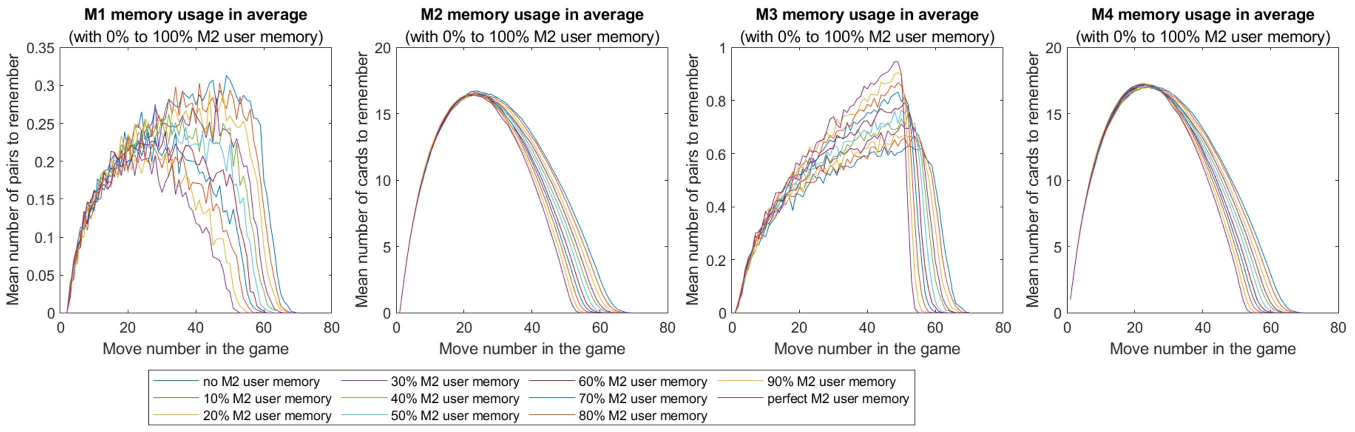

Fig. 6 Mean number of cards or pairs to remember in M1–M4 memories during the gameplays with various M2 user memory and perfect M1, M3, and M4 user memories (game with 32 pairs of cards)

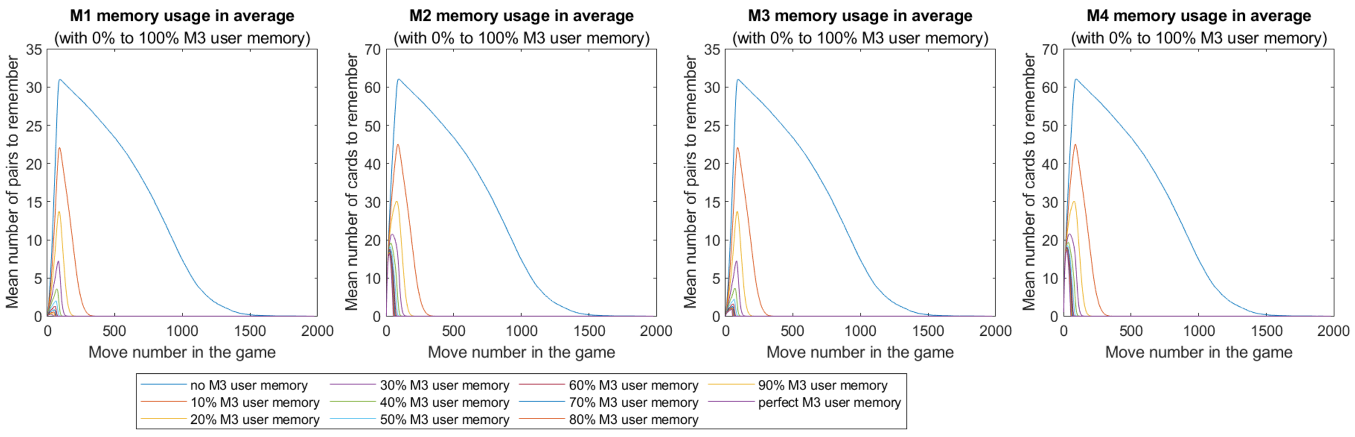

Fig. 7 Mean number of cards or pairs to remember in M1–M4 memories during the gameplays with various M3 user memory and perfect M1, M2, and M4 user memories (game with 32 pairs of cards)

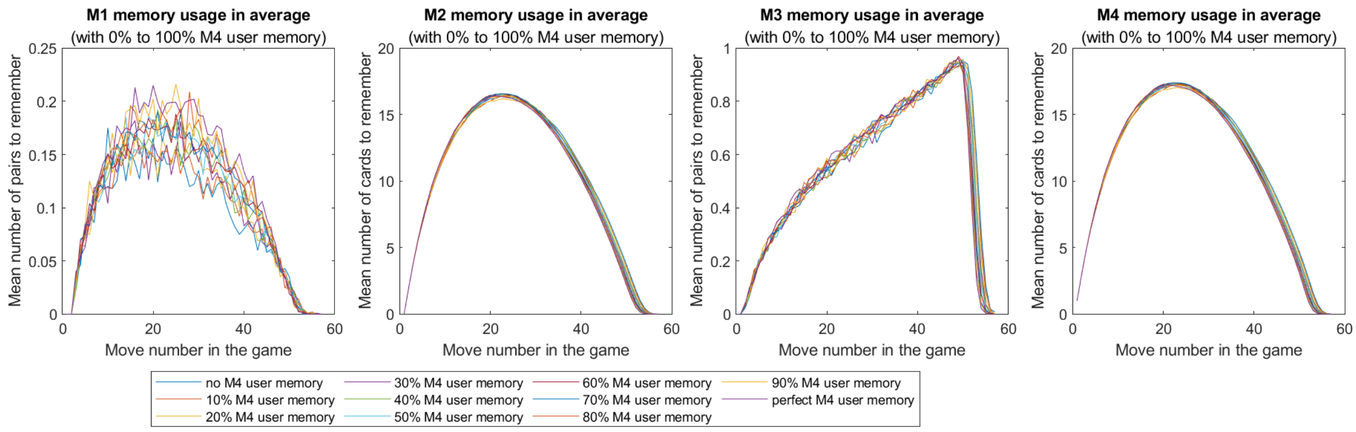

Fig. 8 Mean number of cards or pairs to remember in M1–M4 memories during the gameplays with various M4 user memory and perfect M1, M2, and M3 user memories (game with 32 pairs of cards)

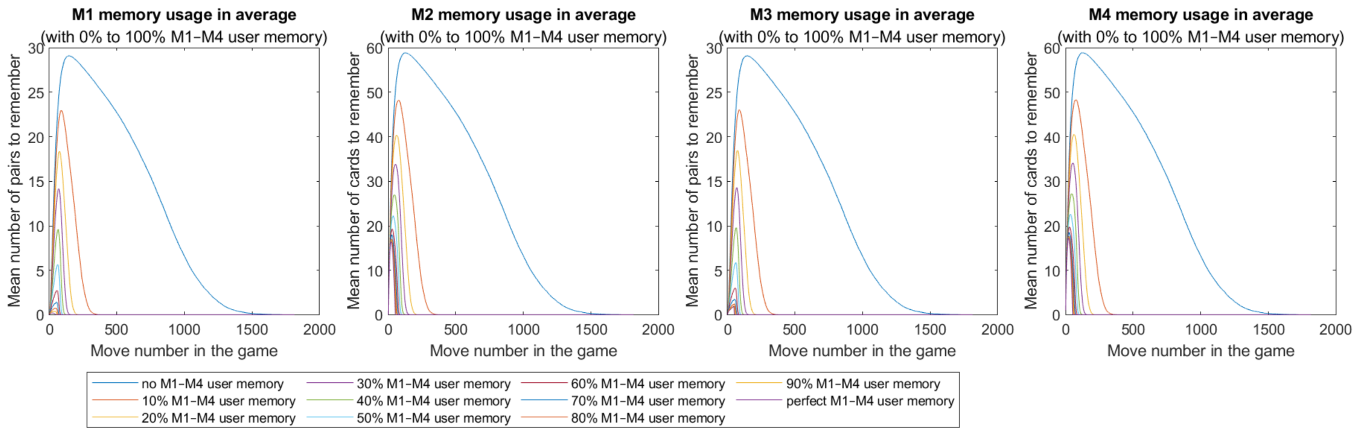

Fig. 9 Mean number of cards or pairs to remember in M1–M4 memories during the gameplays with various M1–M4 user memories (game with 32 pairs of cards)

Next, in the second to fifth simulations, we examined how these graphs would change if only one of the M1, M2, M3, and M4 user memories were not perfect, but the others were perfect.

As we can see in the first and third graphs of Fig. 5, if there is no M1 user memory at all (plots with blue lines), the maximum number of the mean number of pairs to remember is around 6 (i.e., there were maximum 6 pairs of matching cards on average, that were already revealed during the gameplay, but not found by the player). The second and fourth graphs of Fig. 5 show that, with no M1 user memory at all (plots with blue lines), the maximum number of the mean number of cards to remember is around 24 (i.e., there were maximum 24 cards on average, that were already shown during the gameplay, but not found by the player). We can also observe in Fig. 5 that for 40%–100% M1 user memory settings, there is not much difference in the M1, M2, M3, and M4 memory usage (the plots for 40%–100% M1 user memory settings are almost identical). Fig. 10 shows the number of moves necessary to finish the game with the examined M1 user memory settings. As we can see, the medians of the number of moves are between 51 and 56, i.e., the number of moves is not much affected if the player chooses the first card randomly instead of a careful selection.

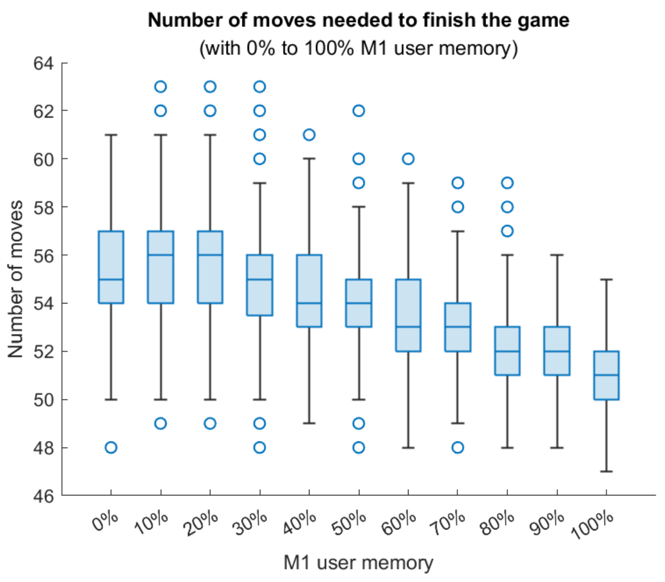

Fig. 10 Number of moves needed to finish the game with various M1 user memory and perfect M2, M3, and M4 user memories (game with 32 pairs of cards)

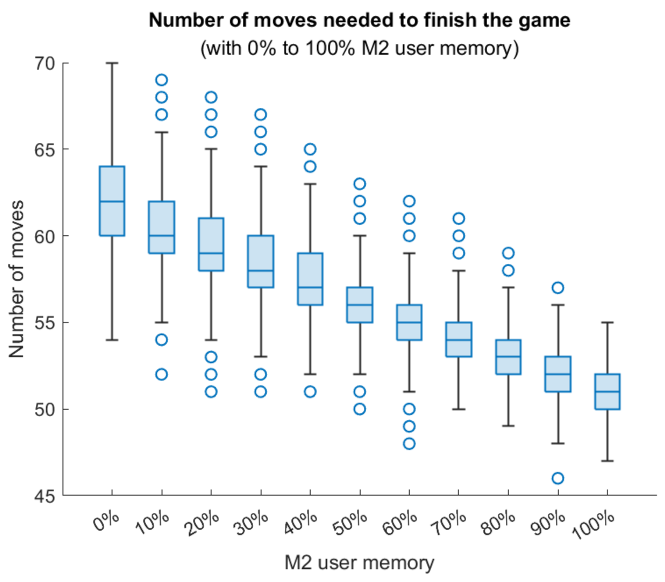

Fig. 11 Number of moves needed to finish the game with various M2 user memory and perfect M1, M3, and M4 user memories (game with 32 pairs of cards)

Fig. 6 shows how M1, M2, M3, and M4 memory usage changes during the gameplay with various M2 user memory settings, i.e., how the number of remembered cards or pairs changes during the gameplay if the player uses certain randomness in the selection of the first card when there are no earlier shown matching cards. As we can see, various M2 user memory settings do not really affect the number of cards or pairs the player needs to remember; the plots for different settings are very similar. Fig. 11 shows the number of moves necessary to finish the game with the examined M2 user memory settings. As we can see, the medians of the number of moves are between 51 and 62, i.e., the number of moves is not much affected by adding randomness to the M2 user memory.

Fig. 7 illustrates how M1, M2, M3, and M4 memory usage changes during the gameplay with various M3 user memory settings, i.e., how the number of remembered cards or pairs changes during the gameplay if the player uses certain randomness in matching the second card to the first card. As shown in all the graphs of Fig. 7, this is a crucial part of the game; when the player does not choose the right second card that matches the first card, all the M1, M2, M3, and M4 memory usage dramatically increase. E.g., by randomly selecting the second card instead of choosing the earlier revealed second card that matches the first card (no M3 user memory at all – plots with blue lines in Fig. 7), there are more than 30 pairs of chards that match and were shown earlier during the gameplay, but not found by the player (first and third plot of Fig. 7). Similarly, the second and the fourth graphs of Fig. 7 shows that there is a point in the gameplay when more than 60 cards (almost all the cards used in the game) were revealed earlier, but not removed from the game by the player (the matches were not found for these cards by the player). However, we can also see that the number of remembered cards or pairs decreases significantly even using only 10% less randomness for M3 user memory (red plots on the graphs, i.e., the second plots from the top). Fig. 12 shows the number of moves necessary to finish the game with the examined M3 user memory settings. We can observe that using at least 10% M3 user memory (i.e., not using randomness at least in 10% of the cases) significantly decreases the number of moves needed to finish the game.

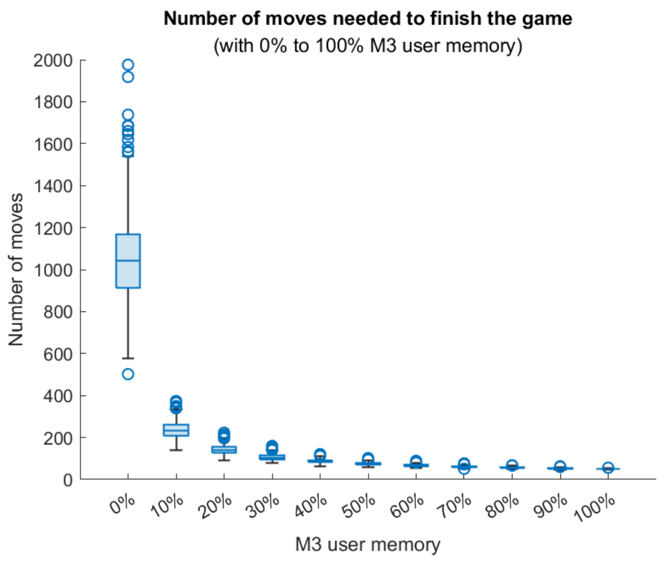

Fig. 12 Number of moves needed to finish the game with various M3 user memory and perfect M1, M2, and M4 user memories (game with 32 pairs of cards)

Fig. 8 shows how M1, M2, M3, and M4 memory usage changes during the gameplay with various M4 user memory settings, i.e., how the number of remembered cards or pairs changes during the gameplay if there is no match during the selection of the second card, and the player uses certain randomness in the selection of the second card (i.e., in certain percent of cases the user selects a random card of all the cards in the game, instead of choosing a random card of the earlier yet not revealed cards). As we can observe on all graphs of Fig. 8, there is not much difference in the plots (all plots are almost identical), i.e., the selection of the second card when there is no match to the first card, does not really affect the M1, M2, M3, or M4 memory usage during the gameplay. Fig. 13 shows the number of moves needed to finish the game with the examined M4 user memory settings. As we can see, the medians of the number of moves are between 51 and 53, i.e., the number of moves is not much affected if the player chooses the second card randomly instead of a careful selection (in the case there is no math to the first card).

Finally, we were curious how the M1, M2, M3, and M4 memory usage is affected during the gameplay and how many moves are necessary to finish the game if we add randomness in all the M1, M2, M3, and M4 user memory setting together. Fig. 9 illustrates the M1, M2, M3, and M4 memory usage during these simulations. Fig. 14 shows the number of moves needed to finish the game with the examined user memory settings.

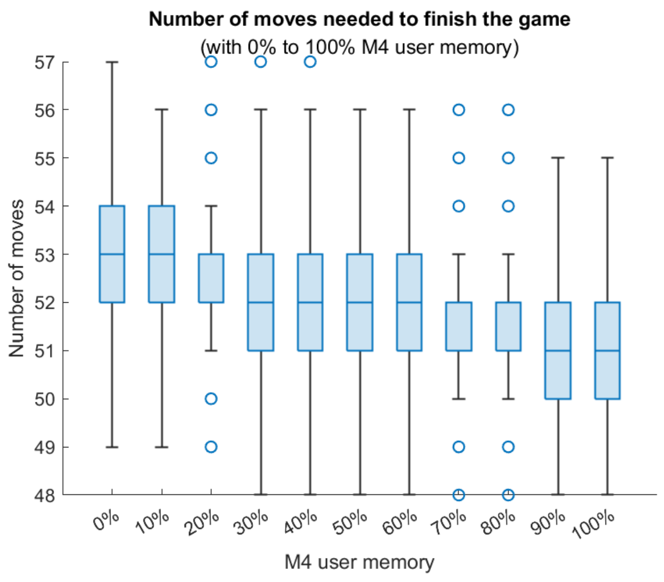

Fig. 13 Number of moves needed to finish the game with various M4 user memory and perfect M1, M2, and M3 user memories (game with 32 pairs of cards)

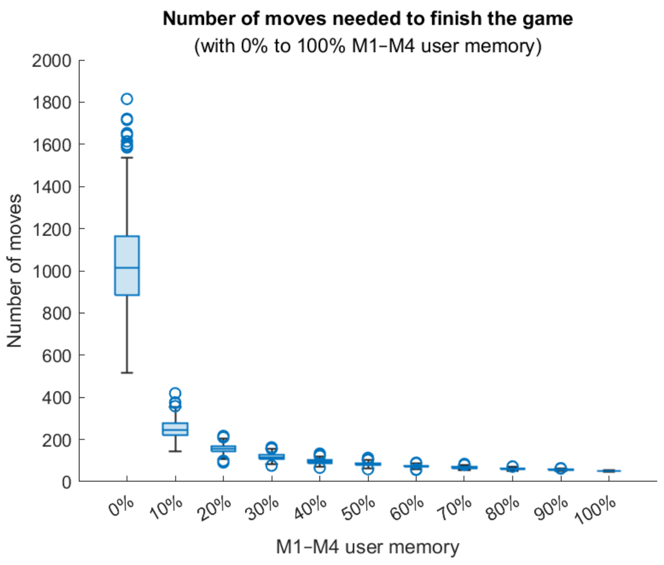

Fig. 14 Number of moves needed to finish the game with various M1–M4 user memories (game with 32 pairs of cards)

We can view that Fig. 9 and Fig. 14 (where we added randomness to all user memories) are very similar to Fig. 7 and Fig. 12 (where we added randomness only to the M3 user memory). This observation supports that a significant part of the gameplay is matching the second card to the first card; the selection of cards in the other steps of the optimal algorithm affects the results only slightly.

IV. Conclusion

In this paper, we defined a human-like thinking optimal algorithm for solving single-player memory card games. We marked those steps of the algorithm with M1, M2, M3, and M4, where the players need to use their memory to select the correct cards. Afterward, we executed several simulations with different M1, M2, M3, and M4 user memory settings by adding randomness to the card selections. The results showed many interesting observations; however, the most important was that the crucial part of the algorithm is related to M3 memory when the user matches the second card to the first card; adding randomness to the other algorithm steps affects the result only very slightly.

We believe the results could be used in future research, e.g., in the psychological examination of the player's working memory, designing efficient educational memory card games, or developing computer opponents for memory card games.

Acknowledgment

The paper was supported by project KEGA 013TTU-4/2021 "Interactive animation and simulation models for deep learning".

References

[1] M. A. Khenissi, F. Essalmi, M. Jemni, Kinshuk, T.-W. Chang, and N.-S. Chen, "Unobtrusive monitoring of learners' interactions with educational games for measuring their working memory capacity," *British Journal of Educational Technology*, vol. 48, issue 2, pp. 224–245, 2016. https://doi.org/10.1111/bjet.12445

[2] D. J. Velleman, and Gregory S. Warrington, "What to Expect in a Game of Memory," *The American Mathematical Monthly*, vol. 120, issue 9, pp. 787–805, 2013. https://doi.org/10.4169/amer.math.monthly.120.09.787

[3] P. Talwalkar. (2015) The Best Way to Play Memory (Card Game) According to Math – Game Theory Tuesdays. [Online] Available: https://mindyourdecisions.com/blog/2015/12/15/the-best-way-to-play-memory-card-game-according-to-math-game-theory-tuesdays/

[4] P. Iverson, B. P. Herrero, and A. Katashima, "Memory-card vowel training for child and adult second-language learners: A first report," *JASA Express Letters*, vol. 3, issue 1, p. 015202, January 2023. https://doi.org/10.1121/10.0016836

[5] N. Murincsáková, "Memóriajáték készítése idegen nyelvek gyakorlására," Bachelor thesis, J. Selye University, Komárno, Slovakia, June 2023.

[6] L. Végh. (2013) Matek.ide.sk – Matematikai versenyek feladatai. Letölthető oktatóprogramok. [Online]. Available: https://matek.ide.sk/index.php?pld=oktprg.htm

[7] D. Yonathan, Susandi, and Y. Arifin, "Designing Memory Game for Learning Healthy Life," *Procedia Computer Science*, vol. 179, pp. 670–676, 2021. https://doi.org/10.1016/j.procs.2021.01.054

[8] T. Chaisewikul, P. Wattanapanich, S. Komgris, and D. Wongsawang, "Memory Skill Games for Elderly People to Prevent Dementia," in *2018 Seventh ICT International Student Project Conference (ICT-ISPC)*, Nakhonpathom, Thailand, 2018, pp. 1–4. https://doi.org/10.1109/ICT-ISPC.2018.8523928

[9] U. Zwick, and M. S. Paterson, "The memory game," *Theoretical Computer Science*, vol. 110, issue 1, pp. 169–196, March 1993. https://doi.org/10.1016/0304-3975(93)90355-W

[10] N. Annuš, D. Paksi, and M. Csóka, "Interactivity and Animation-Simulation Tools in the Digital Presentation of Educational Material," in *INTED2023 Proceedings*, 2023, pp. 6975–6980. https://doi.org/10.21125/inted.2023.1895

[11] K. Czakóová, "Mathematical Model Based Interactive Simulations in Education," in *ICERI2019 Proceedings*, 2019, pp. 10120–10125. https://doi.org/10.21125/iceri.2019.2479

[12] V. Stoffová, M. Zboran, and H. Hyksová, "Application of simulation tools in educational robotics," in *EDULEARN21 Proceedings*, 2021, pp. 9214–9221. https://doi.org/10.21125/edulearn.2021.1856

[13] J. Udvaros, and K. Czakóová, "Developing of computational thinking using microcontrollers and simulations," in *EDULEARN21 Proceedings*, 2021, pp. 7945–7951. https://doi.org/10.21125/edulearn.2021.1619

[14] V. Stoffová, and V. Gabaľová, "Interactive Simulation Models for Teaching and Learning of Computer Science," in *INTED2023 Proceedings*, 2023, pp. 3945–3953. https://doi.org/10.21125/inted.2023.1050

[15] K. Czakóová, "Virtual programming environments and simulations in favor of active learning of programming," *International Journal of Advanced Natural Sciences and Engineering Researches*, vol. 7, no. 5, pp. 105–109, 2023. https://doi.org/10.59287/ijanser.910

[16] D. Paksi, N. Annuš, I. Štempeľová, and D. Dancsa, "Random number-based Brownian motion and practical examples of its implementing in high school computer science lessons," *International Journal of Advanced Natural Sciences and Engineering Researches*, vol. 7, no. 4, pp. 463–467, 2023. http://doi.org/10.59287/ijanser.792