

Mjx: A framework for Mahjong AI research

Sotetsu Koyamada^{*†}, Keigo Habara^{*}, Nao Goto^{*}, Shinri Okano[‡], Soichiro Nishimori[§] and Shin Ishii^{*†¶}

^{*}Graduate School of Informatics, Kyoto University, Kyoto, Japan
Email: koyamada-s@sys.i.kyoto-u.ac.jp; habara.keigo.46r@st.kyoto-u.ac.jp;
gotou.nao.54w@st.kyoto-u.ac.jp; ishii@i.kyoto-u.ac.jp

[†]Advanced Telecommunications Research Institute International (ATR), Kyoto, Japan

[‡]Faculty of Science, Kyoto University, Kyoto, Japan
Email: okano.shinri.24m@st.kyoto-u.ac.jp

[§]Faculty of Integrated Human Studies, Kyoto University, Kyoto, Japan
nishimori-soichiro358@g.ecc.u-tokyo.ac.jp

[¶]International Research Center for Neurointelligence (WPI-IRCN),
The University of Tokyo Institutes for Advanced Study, The University of Tokyo, Tokyo, Japan

Abstract—Numerous games have served as testbeds for artificial intelligence (AI) research to measure its progress. Mahjong is a highly challenging multi-agent imperfect information game with a vast player population. However, a challenge with using Mahjong as a testbed for AI is the lack of a publicly available framework that is fast, easy to use and implements popular rules for human players. We propose and describe Mjx, an open-source Mahjong framework, which implements one of the most popular Mahjong rules, riichi Mahjong (Japanese Mahjong). We compared the execution speed of Mjx with existing popular open-source software and demonstrated that it achieves 100x faster performance. Mjx is available at <https://github.com/mjx-project/mjx>.

Index Terms—Mahjong, reinforcement learning, artificial intelligence, multi-agents, imperfect information game.

I. INTRODUCTION

Artificial intelligence (AI) has made remarkable progress in recent years, and AI which excels at games, is one of the most important milestones to measure its evolution. Examples of reported AI progress through popular games include Backgammon [1], Chess [2], Go [3], [4], Shogi [5], Poker [6]–[8], and Atari 2600 [9].

Behind such remarkable achievements, enormous engineering efforts exist. For example, [9] showed the high potential of deep reinforcement learning (RL) by using Deep Q-Network (DQN) for Atari 2600 games. To this study, the development of Arcade Learning Environment [10] (ALE), which treats Atari 2600 games as a testbed for AI research, has significantly contributed. ALE is also built on top of Stella¹, an emulator of Atari 2600. Through continuous development of ALE (e.g., Python support), Atari 2600 is now one of the most commonly used benchmarks in AI research.

Mahjong is one of the most popular classic multiplayer imperfect information games. In Mahjong, a player initially observes only 13 of 136 tiles, and there is enormous uncertainty from the permutation of the remaining tiles. As

```
from mjx import MjxEnv
from mjx.agents import RandomAgent

agent = RandomAgent()
env = MjxEnv()
obs_dict = env.reset()
while not env.done():
    actions = {player_id: agent.act(obs)
               for player_id, obs in obs_dict.items()}
    obs_dict = env.step(actions)
returns = env.rewards()
```

Fig. 1. **Example usage of Mjx.** Every player acts randomly. Different agents can be used for different players depending on the player id, if necessary. As typical Mahjong rules give non-zero rewards only at the terminal state, this example calls the reward function only after the transitions terminate.

humans must make decisions under high uncertainty in the real world, Mahjong can be a good testbed for AI in this sense. In addition, one of the key indicators when using games as a benchmark for AI is the comparison with human experts (e.g., professional players). Mahjong, like Shogi, has professional players in Japan², which is another good reason to use Mahjong as a testbed for AI research.

However, using Mahjong as a testbed for AI research is currently difficult due to the lack of a framework that is easy to use, fast, and implements popular rules for humans. For example, Mjai [11], the most popular simulator in riichi Mahjong (Japanese Mahjong), does not focus on execution speed and is very slow. Slow execution speed is a significant problem, given that AI technologies such as Deep RL require huge samples in general.

The contribution of this research is to make Mahjong easy to use as a testbed for AI research, just as ALE did for Atari 2600 games. We developed Mjx, based on the popular riichi Mahjong, which focuses on execution speed, and provides human-friendly APIs. In the following sections,

S. Nishimori is now with the University of Tokyo.

¹<https://stella-emu.github.io>

²<https://m-league.jp>

we first describe the characteristics of Mahjong as a testbed for AI research (section II) and then explain Mjx in contrast to Mjai (section III). We then compare the execution speeds of Mjx and Mjai and show that Mjx is capable of high-speed simulations (section IV).

II. GAME OF MAHJONG AS AI TESTBED

This chapter summarizes the nature of the game of Mahjong as a testbed for AI research. Note that the properties of Mahjong described here are not only limited to riichi Mahjong but also apply to other Mahjong rules. Therefore, we do not describe the detailed rules of Mahjong in this paper; see [12] for the rules of riichi Mahjong in detail.

Multi-agent. Mahjong is a four-player game. There is no popular two-player rule, unlike poker. Also, there are no teams, unlike contract bridge.

Imperfect Information. Mahjong, like poker and contract bridge, is an imperfect information game. In Mahjong, information about the opponent’s hand and the pile of tiles are hidden.

Vast hidden state space. What characterizes Mahjong most is the vast space of hidden states. In Mahjong, except for the 13 tiles in hand, all 136 tiles are initially unobservable. Thus, at least $\frac{(136-13)!}{(13!)^3(4!)^{34}} \approx 5 \times 10^{128}$ possible states correspond to each (initial) observation. This number is much larger than in poker, which implies that large randomness behind the observable information critically impacts Mahjong playing. Moreover, this characteristic makes it hard to apply planning techniques to the game.

Sparse Rewards. As in Go, Shogi, and Chess, non-zero rewards are given only at the end of the game; there is no immediate feedback on whether each action choice was good (or not). In Mahjong, one game consists of 8 or more hands, and the player takes approximately 10 to 20 actions per hand, so the non-zero reward is given only after more than a hundred actions. Note that seeking the win in every hand does not lead to a higher final ranking.

III. MJX

In this section, we first describe the design principles which express our motivation for the development of Mjx (section III-A). Then, we explain how we implemented these design principles by describing the Mjx features (section III-B). Finally, to better understand Mjx, we compare the features against a popular existing Mahjong simulator (section III-C).

A. Mjx design principles

We designed Mjx such that it can complement prior research and existing software in the following three points:

1) Rules popular with human players. Like poker, there are several rule variants in Mahjong. One of the key benchmarks in AI research is comparing the performance of AIs with human experts. Some frameworks offer simplified Mahjong as a testbed for AI research [13]. While simplified variants are also helpful for research, the number of human players is small

(or zero), and AIs are not comparable with human experts such as professional Mahjong players.

2) Fast. Elemental technologies in AI, such as deep RL, require huge samples via simulation. For this reason, Mjx focused on simulator speed. We will demonstrate that Mjx achieved much faster simulations than another current popular Mahjong simulator in section IV.

3) Friendly interface for researchers. While it is vital to speed up the computation time, it is also crucial to reduce the time required for human trial and error in research. We intended to make it easy for researchers to develop and experiment with Mjx by adapting its interface to the current community standards for AI research.

B. Mjx features

We describe the features of Mjx, which implements the design principles.

Mahjong rule. Mjx follows riichi Mahjong rule. It especially follows the rule of the Tenhou online platform [14]. Tenhou is a popular online platform for competitive players. There are several reasons for this choice:

- riichi Mahjong is one of the most popular rules in the world,
- it has competitive human players (professional Mahjong players), and
- several previous Mahjong studies follow this Tenhou rule [15]–[17].

Programming language. We provide Mjx as a Python library since Python is the *lingua franca* in current AI research and development. However, Python is tens of times slower in execution speed than speed-oriented programming languages such as C++. Therefore, we implemented the core part of Mjx in C++ to enable fast simulations and wrapped it in Python. This wrapping makes it possible for AI researchers to use Python for a better development experience without sacrificing fast simulations.

API (Application Programming Interface). We designed Mjx’s API similar to OpenAI Gym’s API, one of the most popular APIs in RL. See Fig. 1 for an example of the usage. In particular, we also provide a strictly compatible API with OpenAI Gym, in which users fix three of the four agents to a particular agent as an opponent and use the feature extraction function provided by Mjx (or defined by users). It allows the implementation of algorithms working with OpenAI Gym to be used directly with Mjx. Note that Mjx has a server feature, and users can use any programming language to implement the agent if they do not stick to Python API.

Visualization. Mjx supports the visualization of observations and states as image (Fig. 2). This visualization feature must help researchers, with domain knowledge in Mahjong, develop sophisticated AIs.

C. Comparison to Mjai

To better understand the characteristics of Mjx, we compare Mjx and Mjai. Mjai [11] is currently one of the most popular

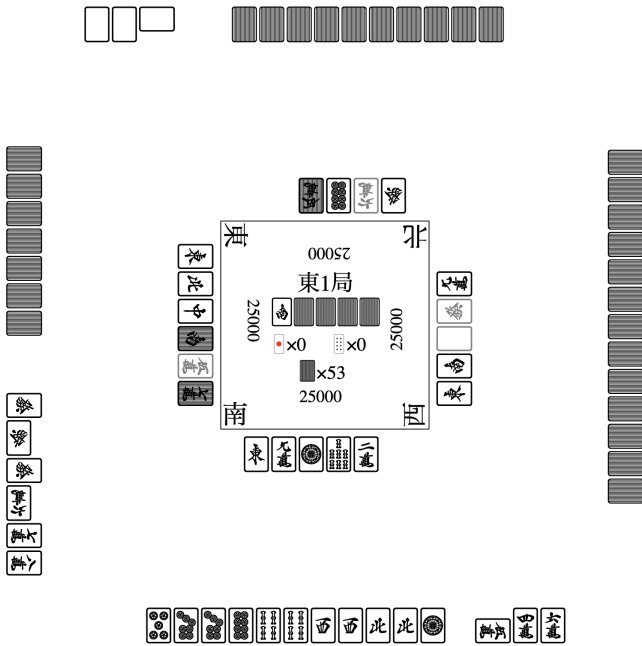


Fig. 2. Example visualization in Mjx.

TABLE I
MJX AND MJAI COMPARISON

	Mjx (proposed)	Mjai
Mahjong Rule	Tenhou rule	Tenhou rule
Programming Language	Python (C++)	Ruby
Gym-like API	✓	
Stateless Protocol	✓	

open-source simulators available for riichi Mahjong. Table I compares the features of Mjx and Mjai.

Both Mjx and Mjai use the same Tenhou rule. While Mjai uses the Ruby programming language, Mjx uses Python. Both are not computation speed-oriented programming languages, but Mjx also uses C++ at its core to accelerate the computation speed. As Mjx uses Python, it can also provide a *gym-like* API, similar to the popular OpenAI Gym environment in RL. Another difference is the communication protocol between the agents and the simulator. Mjai’s protocol uses a *stateful* protocol, while Mjx’s is *stateless*. In Mjai, the agents must *remember* the current situation during the game. Stateless architecture design is generally easier to accelerate computation through parallel and distributed computing. This feature is vital when the agents infer the actions in batch using neural networks.

IV. PERFORMANCE ON SPEED BENCHMARK

To compare the computational speed of Mjx and Mjai, we compared the time required to run 100 games for both Mjx and Mjai using two different agents (Fig. 3).

Setup. We measured the execution time of 100 games in each setting five times and averaged the results. We used an Amazon Web Service (AWS) instance as the execution

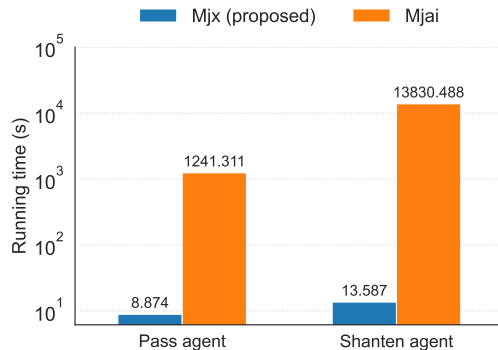


Fig. 3. Running time for 100 games. *Pass agent* plays to retain its hand, and *shanten agent* plays to win its hand ignoring the opponents.

environment to facilitate reproducibility. The instance is an *m6i.large* instance in the *us-east-1* region with two vCPUs and 8 GiB mem. The launched instance had an Intel(R) Xeon(R) Platinum 8375C CPU @ 2.90GHz. We used two types of agents: *pass agent* and *shanten agent*. The pass agent does nothing during the play, and thus its hand does not change during the play, and no one wins. The shanten agent always plays to reduce the *shanten number*, the minimum number of tile swaps until the hand wins. In other words, it always plays only for the win of its hand, ignoring the other players’ moves. As a result, the pass agent’s decision-making time is short, and thus it is convenient to measure the execution speed of the Mahjong engines. On the other hand, the shanten agent has a longer decision-making time, but we can measure the computation time in more practical situations. We implemented both agents using the Mjx and Mjai libraries, respectively.

Results. Fig. 3 shows that Mjx exhibits 100x faster performance than Mjai for the pass agent scenario. Also, for the shanten agent scenario, Mjx performed 1000x faster on running time than Mjai. The difference in speed improvement between the two scenarios is due to the difference in the speed of the shanten calculations provided by Mjx and Mjai. These results indicate that Mjx can execute Mahjong games dramatically faster than Mjai. Therefore, we expect that Mjx can be used to evaluate and train Mahjong AI more efficiently.

V. RELATED WORK

We have provided Mjx as a tool for AI development in the game of Mahjong, especially as an environment for the development and evaluation of RL algorithms. Thus, this section describes recent developments in RL environments for research and progress in Mahjong research.

RL environments. There are numerous environments for the research of RL algorithms. One popular and commonly used environment is the Arcade Learning Environment [10] (ALE), where researchers can train and evaluate RL agents for Atari 2600. ALE became a popular benchmark for discrete action space with visual observation, especially after DQN

research [9]. OpenAI Gym [18] provides various environments that range from classical tasks to control in continuous space with the MuJoCo physics engine [19]. OpenAI Gym’s learning API has significantly impacted other RL environments, and researchers have developed gym-like environments and agents compatible with OpenAI Gym. PettingZoo [20] provides several benchmark environments for multi-agent RL with gym-like API. RLCards [13] offers various environments related to card games, such as poker. Particularly, RLCards has a simplified version of the Mahjong environment, of which PettingZoo provides a wrapper in its gym-like API. However, this Mahjong environment is simplified and is different from riichi or other Mahjong variants with a large human playing population. OpenSpiel [21] provides a collection of various game environments for research in RL. MinAtar [22] has five simplified versions of Atari games intended to enhance the reproducibility and speed of RL research. Rogue-Gym [23] proposed a roguelike game environment to evaluate RL algorithms’ generalization performance. OpenHoldem [24] provides a toolkit for research using No-limit Texas Hold’em.

Mahjong research. Bakuuchi [15], the first well-known Mahjong AI, used Monte Carlo simulation and performed better than the average human player. Kurita and Hoki [16] built an AI player comparable to Bakuuchi by constructing Markov decision processes using state aggregation. Suphx [17] outperforms most top human players in Mahjong using deep learning and RL.

VI. CONCLUSION

Mahjong is a highly challenging imperfect information game with a massive number of human players and has high potential as a testbed for the development of AI. However, owing to its rule’s complexity, no simulator exists that is reasonably fast and compatible with the popular riichi Mahjong rule. The lack of a fast simulator is a bottleneck for Mahjong AI research and development since recent deep RL and planning techniques require a large sample size. We have developed Mjx, a Mahjong engine that is fast, compatible with the popular Tenhou rule, and friendly with recent AI community standard programming languages and APIs. We demonstrate the usefulness of Mjx by comparing its qualitative differences and computation speed with Mjai, one of the popular simulators currently available. We believe that Mjx has the potential to facilitate the development of Mahjong AI research and enhance research on AI capable of making difficult decisions under imperfect information.

REFERENCES

- [1] G. Tesauro, “Temporal difference learning and TD-Gammon,” *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, Mar. 1995.
- [2] M. Campbell, A. J. Hoane, and F.-H. Hsu, “Deep blue,” *Artificial Intelligence*, vol. 134, no. 1, pp. 57–83, Jan. 2002.
- [3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.
- [4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, “Mastering the game of go without human knowledge,” *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.
- [5] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, “A general reinforcement learning algorithm that masters chess, shogi, and go through self-play,” *Science*, vol. 362, no. 6419, pp. 1140–1144, Dec. 2018.
- [6] N. Brown and T. Sandholm, “Superhuman AI for heads-up no-limit poker: Libratus beats top professionals,” *Science*, vol. 359, no. 6374, pp. 418–424, Jan. 2018.
- [7] M. Moravčík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling, “DeepStack: Expert-level artificial intelligence in heads-up no-limit poker,” *Science*, vol. 356, no. 6337, pp. 508–513, May 2017.
- [8] N. Brown and T. Sandholm, “Superhuman AI for multiplayer poker,” *Science*, vol. 365, no. 6456, pp. 885–890, Jul. 2019.
- [9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
- [10] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, “The arcade learning environment: An evaluation platform for general agents,” *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, Jun. 2013.
- [11] H. Ichikawa, “Mjai: Game server for Japanese mahjong AI,” <https://github.com/gimite/mjai>, accessed: 2022-3-4.
- [12] European Mahjong Association, “Rules for Japanese Mahjong,” <http://mahjong-europe.org/portal/images/docs/Riichi-rules-2016-EN.pdf>, accessed: 2022-3-4.
- [13] D. Zha, K.-H. Lai, Y. Cao, S. Huang, R. Wei, J. Guo, and X. Hu, “RLCard: A toolkit for reinforcement learning in card games,” *arXiv:1910.04376*, Oct. 2019.
- [14] S. Tsunoda, “Tenhou,” <https://tenhou.net/>, accessed: 2022-3-4.
- [15] N. Mizukami and Y. Tsuruoka, “Building a computer mahjong player based on Monte Carlo simulation and opponent models,” in *IEEE Conference on Computational Intelligence and Games*, Aug. 2015.
- [16] M. Kurita and K. Hoki, “Method for constructing artificial intelligence player with abstractions to markov decision processes in multiplayer game of mahjong,” *IEEE Transactions on Games*, vol. 13, no. 1, pp. 99–110, Mar. 2021.
- [17] J. Li, S. Koyamada, Q. Ye, G. Liu, C. Wang, R. Yang, L. Zhao, T. Qin, T.-Y. Liu, and H.-W. Hon, “Suphx: Mastering mahjong with deep reinforcement learning,” *arXiv:2003.13590*, Mar. 2020.
- [18] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “OpenAI Gym,” *arXiv:1606.01540*, Jun. 2016.
- [19] E. Todorov, T. Erez, and Y. Tassa, “MuJoCo: A physics engine for model-based control,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012.
- [20] J. Terry, B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan, L. S. Santos, C. Dieffendahl, C. Horsch, R. Perez-Vicente *et al.*, “PettingZoo: Gym for multi-agent reinforcement learning,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 15 032–15 043, 2021.
- [21] M. Lanctot, E. Lockhart, J.-B. Lespiau, V. Zambaldi, S. Upadhyay, J. Pérolat, S. Srinivasan, F. Timbers, K. Tuyls, S. Omidshafiei, D. Hennes, D. Morrill, P. Muller, T. Ewalds, R. Faulkner, J. Kramár, B. De Vylder, B. Saeta, J. Bradbury, D. Ding, S. Borgeaud, M. Lai, J. Schrittwieser, T. Anthony, E. Hughes, I. Danihelka, and J. Ryan-Davis, “OpenSpiel: A framework for reinforcement learning in games,” *arXiv:1908.09453*, Aug. 2019.
- [22] K. Young and T. Tian, “MinAtar: An Atari-inspired testbed for thorough and reproducible reinforcement learning experiments,” *arXiv:1903.03176*, Mar. 2019.
- [23] Y. Kanagawa and T. Kaneko, “Rogue-Gym: A new challenge for generalization in reinforcement learning,” in *IEEE Conference on Games*, Aug. 2019.
- [24] K. Li, H. Xu, M. Zhang, E. Zhao, Z. Wu, J. Xing, and K. Huang, “OpenHoldem: An open toolkit for large-scale imperfect-information game research,” *arXiv:2012.06168*, 2020.