Assessing Simultaneous Action Selection and Complete Information in TAG with Sushi Go!

Carl-Magnus Embring Klang*, Victor Enhörning†, Alberto Alvarez‡, Jose Font§
Game Lab, Department of Computer Science and Media Technology
Malmö University, Sweden
Email: *carl.magnus@hotmail.com, †victor.enhorning@gmail.com
Email: {‡alberto.alvarez,§jose.font}@mau.se

***Abstract*—Digitalizing tabletop games for general game playing (GGP) AI research is a continuously growing field. Tabletop Games Framework (TAG) is a framework developed to simplify the process of implementing tabletop board games to digital form. Sushi Go! is a game that combines simultaneous action selection and complete information. This creates a unique combination of mechanics, which presents a new challenge for GGP agents. By implementing Sushi Go! into TAG, we can test different agent's performance using these mechanics and compare them to their existing performances in the other games of TAG. Results of this testing are presented, which display that the framework is capable of implementing Sushi Go! and that the agents perform with mixed results. Further developing heuristics for the agents should prove to increase their performance when faced with these types of games.**

***Index Terms*—General game playing, game artificial intelligence, tabletop games**

I. Introduction

Games have for many years been a tool to research artificial intelligence, from chess to Blood Bowl [1], [2]. To improve artificial intelligence research on games, several frameworks have been created to help simplify the process. Game frameworks are quite a helpful resource that provides artificial intelligence researchers with tools for rapid algorithm deployment, data collection, baseline algorithms, and generalized heuristics that decrease their workload and make comparative analyses possible. One such framework is GVG-AI, which provides different types of video games and tools [3] used for competition in artificial intelligence. There are also frameworks that can be used to describe and explore board games that use pre-defined mechanics like Ludii [4] and OpenSpiel [5]. However, both focuses on traditional board games, and in the case of Ludii, the use of cards is not possible yet.

Another interesting framework is the Tabletop Games Framework (TAG) that offers a way to research general AI in modern tabletop games and add games with uncommon or unique mechanics [6]. The framework is modular and open source to expand its library, and it provides analytics for AI decision making. One interesting feature that is not available in TAG is Simultaneous action selection, which is a genre of turn-based games where all players play their turn at the same time. Sushi Go! is a zero-sum card game with simultaneous action selection and end game bonuses. Sushi Go! offers a unique type of strategic gameplay yet to be implemented into TAG, where simultaneous action selection is combined with delayed complete information. Delayed complete information means that after passing the players' hands N-1 times (N = number of players) there are, technically, no longer hidden elements in the game. However, keeping track of all the hands and card movement is a difficult problem for human players, yet it is part of the game's strategy to win.

As of yet, not a great deal of research has been carried out concerning intelligent agents in tabletop games with simultaneous action selection. Soen explored using genetic algorithms and reinforcement learning to play Sushi Go!, but developed a custom artifact as a test bed [7]. The purpose of this study is therefore to determine if Sushi Go! can be translated and implemented into the TAG framework, studying how the current built-in agents perform in Sushi Go! as well as compared to their performance in the other games available in the framework. Agent performance quantitative data is collected from Sushi Go! and the other games in the framework. By comparing the win rates of the agents, the effect simultaneous action selection has on the intelligent agents of TAG can be observed and researched. The extended framework now offers a way to perform general AI research in Sushi Go! and simultaneous action selection. The framework also now has a way to execute simultaneous action selection in other games by using this developed functionality, which simplifies implementing similar games in the future.

II. Related Work

A. *Sushi Go!*

Sushi Go! [8] is a card game for two to five players where the goal is to collect as many points as possible. The game consists of 3 rounds where each player collects a set of cards to increase their final score. Different cards in the game (table I) offer different amounts of points and some cards have special requirements such as only being valuable if in a pair. The player with the most points after 3 rounds wins the game.

Each round starts by dealing a specified amount of cards for each player. Each player picks one card and places it face down, and when all players have picked a card they reveal them at the same time. Then each deck is passed to the player on the left and they get to pick cards again. Once the decks are empty, the scores are calculated and a new round is started.

TABLE I
ALL THE CARDS IN SUSHI GO! (AND IMPLEMENTED), AND THEIR RESPECTIVE AWARDED POINTS

Card	Points
Maki Roll	The player with the most maki at the end of a turn gets 6 points while the second gets 3. If equal split evenly discarding any remainers.
Tempura	A set of 2 is worth 5 points, otherwise 0.
Sashimi	A set of 3 is worth 10, otherwise 0.
Dumpling	Points received increase with each dumpling the player has collected. 1 dumpling is 1 point, 2 are 3 points, 3 are 6 points, 4 are 10 points, and 5+ are 15 points.
Squid Nigiri	Worth 9 points if stacked on a wasabi card, otherwise 3.
Salmon Nigiri	Worth 6 points if stacked on a wasabi card, otherwise 2.
Egg Nigiri	Worth 3 points if stacked on a wasabi card, otherwise 1.
Wasabi	Worth 0 points unless combined with a nigiri card.
Chopsticks	0 points, is used to swap cards.
PuddingsCar	The player with the most puddings gets 6 points while the person with the least puddings gets -6 points. If equal split evenly.

B. Tabletop Games Framework - TAG

The TAG framework [9] was developed to enable developers to easily digitalize board games by utilizing a standard implementation strategy based on a fixed amount of game components and states. Thanks to this, TAG can use the same AI for every game implemented into the framework. This in turn, allows for extensive research regarding AI performance in complex board games, such as Pandemic or Exploding Kittens, and Colt Express. It enables researchers to compare results between different types of AI, compare results between different games, and if a specialized AI is required, it can be implemented into the framework, allowing all users to use it in their research. TAG contains a set of pre-made agents to play the games within the framework that facilitate the analysis and evaluation of games and how agents play them. The agents can be random, or use either one step look ahead (OSLA), Rolling Mutation Hill-Climbing (RMCH), or monte carlo tree search (MCTS) [6].

C. Simultaneous Action Selection

Simultaneous action selection is a game element that has yet to be widely researched in AI. Wang et al. [10] showed that in a simple rock-paper-scissors game an agent is not only fully capable of understanding and utilizing Simultaneous Action Selection, but also proves to be an efficient player that reliably wins over human opponents. Shafiei et al. [11] use two agents (Upper Confidence Bound (UCT) and Counterfactual Regret (CFR)) to play and test their performance playing simultaneous action selection games with imperfect information. CFR is more robust and proves to be the optimal solution for the tested games, which might point out towards a promising algorithm to be implemented in TAG. Finally, Soen [7] used reinforcement learning and genetic algorithms to play in a custom-made Sushi Go! implementation, but with sub-optimal results. The agents did not have a high win-rate and it was speculated to be either tied to the environment developed not being complex enough to properly handle the mechanics of Sushi Go!, or that the game itself did not function well with the tested approaches for learning. In part, the work by Soen motivates the implementation of Sushi Go! in TAG, which, given it's generic capabilities, should pass-through the limitations he found in his work.

III. SUSHI GO! TAG IMPLEMENTATION

One of the main functionalities of TAG is the ability to easily develop and implement new games for the agents within the framework to play. This is done by a set of core classes which every new game added derives from, which allows the agents to run the same functions for all games encountered in the framework. These classes are: *ForwardModel*, *GameState*, *GameParameters* and *TurnOrder*. These classes contain the set up and core game mechanics, generic game data and functionality, game constants and functionality to keep track of each player's turn.

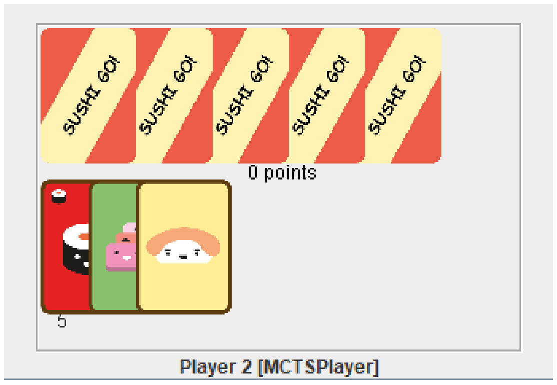

Fig. 1. Details how Sushi Go! is visualized for the player. See the MCTS agent's hand at the top, with their cards hidden from the other players. At the bottom part of the playing field, the active cards played by the agent are visible for all players.

In this implementation of Sushi Go!, the *ForwardModel* handles the setup of the different card decks used in the game. These include a draw deck, a discard deck, a player deck for each player, and an active card deck (i.e., current played cards) for each player. It also handles the computation of available actions, which in turn are given by the cards present in a player's deck. The decks are created in the *GameState* class containing the game's generic data. The cards that fill the different decks are created in the *GameParameters* class. These are a direct copy from the tabletop version of Sushi Go! shown in table I. Finally, the *TurnOrder* holds methods for progressing the turn order, ending the game when the end of game criteria is reached, and keeping track of which player's turn it

TABLE II
SHOWCASES THE WIN RATES FOR THE DIFFERENT AGENTS PLAYING SUSHI GO!. CALCULATED FROM 100 GAMES. ONE INSTANCE OF EACH AGENT, ALL COMPETING AGAINST THE OTHERS.

Algorithms	*Random*	*OSLA*	*RMHC*	*MCTS*
Win Rate	21%	39%	19%	21%

is. The framework primarily offers support for games using an altering turn order, to implement simultaneous action selection and delayed complete information the *TurnOrder* class was extended to control when turns and rounds increase. This way each player plays their turn simultaneously, simulating simultaneous action selection properly.

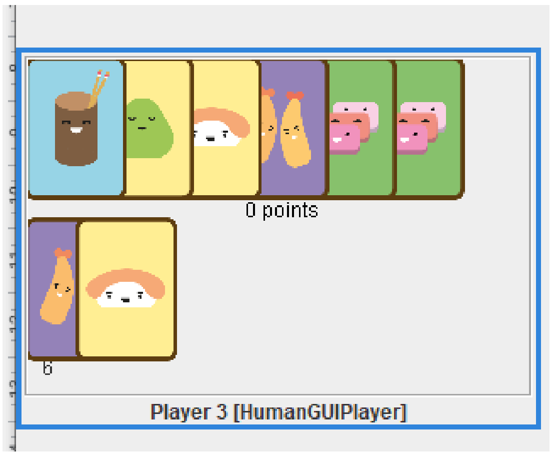

Fig. 2. Showcases what a human player sees when they play Sushi Go!. At the top is their hand with cards available for play. At the bottom is the active cards that have already been played. The number in the bottom left corner is the amount of cards in the player's hand.

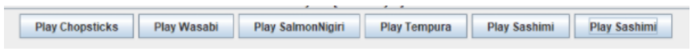

Fig. 3. Showcases how available actions are presented to a human player, related to every playable card in the player's hand (Figure 2).

Figure 1 shows a sample screenshot of the game board from Player 2's perspective. Figure 2 shows a screenshot a human player's own hand of cards. Figure 3 shows the player interface containing all the available actions. In Sushi Go!, actions are tied to the cards that each player has in their hand. For instance, if a player has a wasabi card in play, a nigiri card can be played on top of it to double the points of the played nigiri. Finally, the current implemented heuristic and the one used by agents simply calculates the current score accumulated by the player with no consideration to possible combos or future rewards.

IV. EXPERIMENT AND RESULTS

Our experiment consisted of using the built-in agents in TAG to determine how well they play Sushi Go! when introduced to the simultaneous action selection mechanic, which is the implementation's goal. We ran 100 games where each player is one of the four built-in GGP agents competing against each other. We evaluate each agent based on their win percentage. This statistic is chosen due to the fact that the same data is used in [6] to determine effectiveness of the different agents. As such, it can be used to directly compare the agents playing Sushi Go! with how well the agents played the other games in TAG. Similarly to Gaina et al. [9], we collected statistics pertaining the random agent in 1000 Sushi Go! games to highlight and analyze the challenge provided for AI agents. The following measurements (shown in table III) present *averages* observed in our experiment:

- d1 Decisions: The number of decisions an agent makes per game.
- d2 Rounds: The number of rounds per game.
- d3 Actions Per Turn: The amount of actions an agent makes per turn.
- d4 Ticks: The number of game loop iterations.
- d5 Action Space Size: The number of actions available to an agent on their turn.
- d6 State Size: The number of components in a state.
- d7 Time Action Compute: Microseconds it takes for an agent to perform an action.

As observed in table II, the highest win rate agent is the OSLA agent. The other agents achieve roughly the same win rate at just around 20%. The OSLA agent is performing noticeably better than the other agents despite all of them using the same heuristic (apart from the random agent). Further, as observed in table III, the agents are able to execute Sushi Go! noticeably swiftly, taking only 61.2 microseconds to compute and execute an action (d7), based on 4.86 available actions per turn (d5). This is also in a state with on average 70 components that are present on the game board and part of the evaluation.

On average, the game length of Sushi Go! is fairly short when compared to the more complex games implemented into TAG[1], such as Virus! which has a tick amount of 319.56 as opposed to Sushi Go! which only has a tick amount of 92 (d4). Sushi Go! differs from most of the other games in TAG due to the limited amount of rounds per game, which is locked at three. Decisions presented to the agents are also on the smaller side when compared to, for example, Virus! and their 317.09 decisions, instead of Sushi Go! and their decision amount of 81 (d1). As can be seen in table III, most of the values concerning game length for Sushi Go! are in line with the less than average complex games of TAG, such as Love Letter or Exploding Kittens [9].

V. DISCUSSION

As shown in table II, the agent using the OSLA algorithm managed to win most games (39%). This deviates from the expected results, as MCTS shows to be overall the most well-performing agent across the already implemented games in TAG [6]. OSLA functions by looking one step ahead in the game state when evaluating what action the agent should take

[1]data extracted from both TAG papers up-to-date [6], [9].

TABLE III
SHOWCASES THE GAME LENGTH AND GAME ANALYTIC. CALCULATED FROM 1000 GAMES, PLAYED WITH RANDOM AGENTS: DECISIONS, ROUNDS, ACTIONS PER TURN, TICKS, ACTION SPACE SIZE, STATE SIZE, SCORE AND TIME ACTION COMPUTE.

	d1	*d2*	*d3*	*d4*	*d5*	*d6*	*d7*
Sushi Go!	81	3	1.105	92	4.86	79	61.2

in order to maximize their own gain, instead of looking deeper as, for instance, the MCTS algorithm does. While this would normally mean that the MCTS agent bases their decisions on a much greater scale then the OSLA agent, we speculate that the improved performance by the OSLA agent is due to a lack in advanced heuristics. As such, an agent that always chooses the most beneficial card for their own score when deciding what action to play, and on average end up in the most winning situation. When compared to other games within TAG, our implementation lacks this advanced and well-tuned heuristic, that could take into consideration different possible combos, long-term rewards such as puddings, or the other decks and other player's cards. While agents can play the game following the game's rules, making use of the combination of complete information and simultaneous action selection, and it is possible to assess their win percentage and game stats, not having a proper heuristic makes it difficult to draw final conclusions on how they handle the specific mechanics.

Furthermore, Sushi Go! was easily implemented in TAG using the existing methods and guides for implementing new games. The game is considerably efficient, as simulating 1000 games gathering data took roughly five minutes on a medium-specs laptop. Our implementation allows customization such as the possibility to change both card values, the amount of cards each player has and how many of each card is in the deck to test different setups. As it can be seen in table III, Sushi Go! is rather limited when it comes to game length and amount of decisions. When compared to the more complex games, the amount of decisions and rounds per game are on the smaller side, which proves beneficial for the performance of the game as it can't exponentially keep going or get stuck in an infinite game loop, due to the limited amount of rounds. This results in an implementation that, while not optimized in every aspect, should prove to be able to execute at very reasonable speeds.

VI. CONCLUSIONS AND FUTURE WORK

Our work provides a detailed example on the Sushi Go! implementation in the TAG framework and the potential that both the framework and the game have for AI research. Sushi Go! allows us to test and investigate complete information games and simultaneous action selection, a common mechanic in board games, and provides the framework with yet another game to further research GGP agents. Through this, we researched how capable built-in agents are at playing Sushi Go!. This particular pair of mechanics gives a unique approach to playing the game. Since every player knows all the cards in play, it is possible to predict what your opponents play or create an adversarial model. However, due to the simultaneous action selection, the agent can't adapt their own action by observing their opponents, but must decide what to play based on their predictions.

Furthermore, the artifact produced can be used in the future to research the combined mechanics of complete information and simultaneous action selection, while the game can also work as a blueprint for future game implementations with the same or similar mechanics. The heuristic used with each agent was very basic; thus, investigating more advanced and generic heuristics to be used by agents in Sushi Go! and games with similar mechanics is an important area of improvement. For instance for Sushi Go!, it would include the benefit of collecting multiple dumpling cards to reach higher score multipliers, and a much higher value in playing wasabi cards due to their score multiplier when combined with a nigiri card.

As presented by Shafiei et al. [11], CounterFactual Regret is a very interesting and promising algorithm that could be implemented and used to cope with simultaneous action selection and delayed complete information. Another interesting area would be to implement self-play agents [12] into Sushi Go! to continuously observe if they could improve and adapt to the mentioned combination of mechanics. With board games continuously growing more and more intricate with the use of digitalization, analyzing intelligent agents playing them in order to further advance them is valuable.

REFERENCES

[1] S. Risi and M. Preuss, "From chess and atari to starcraft and beyond: How game ai is driving the world of ai," *KI - Künstliche Intelligenz*, 2020.

[2] N. Justesen, L. M. Uth, C. Jakobsen, P. D. Moore, J. Togelius, and S. Risi, "Blood bowl: A new board game challenge and competition for ai," in *2019 IEEE Conference on Games (CoG)*, 2019, pp. 1–8.

[3] D. P. Liébana, S. M. Lucas, R. D. Gaina, J. Togelius, A. Khalifa, and J. Liu. The gvg-ai competition. [Online]. Available: http://www.gvgai.net/

[4] E. Piette, D. J. Soemers, M. Stephenson, C. F. Sironi, M. H. Winands, and C. Browne, "Ludii–the ludemic general game system," *arXiv preprint arXiv:1905.05013*, 2019.

[5] M. Lanctot, E. Lockhart, J.-B. Lespiau, V. Zambaldi, S. Upadhyay, J. Pérolat, S. Srinivasan, F. Timbers, K. Tuyls, S. Omidshafiei *et al.*, "Openspiel: A framework for reinforcement learning in games," *arXiv preprint arXiv:1908.09453*, 2019.

[6] R. D. Gaina, M. Balla, A. Dockhorn, R. Montoliu, and D. Perez-Liebana, "Tag: A tabletop games framework," in *Proceedings of the EXAG Workshop at AIIDE*, 2020.

[7] A. Soen, "Making tasty sushi using reinforcement learning and genetic algorithms," *Scientific Reports*, 2019.

[8] Walker-Harding, Phil, "*Sushi Gò*' Tabletop [Card Game], Newton, US., 2013, gamewright, Massachusetts, US. Last played December 2019.

[9] R. D. Gaina, M. Balla, A. Dockhorn, R. Montoliu, and D. PerezLiebana, "Design and implementation of tag: A tabletop games framework," 2020.

[10] L. Wang, W. Huang, Y. Li, J. Evans, and S. He, "Multi-ai competing and winning against humans in iterated rock-paper-scissors game," *Scientific Reports*, vol. 10, no. 1, pp. 1–8, 2020.

[11] M. Shafiei, N. R. Sturtevant, and J. Schaeffer, "Comparing uct versus cfr in simultaneous games," in *IJCAI Workshop on General Game Playing*, 2009.

[12] D. Hernandez, K. Denamganaï, Y. Gao, P. York, S. Devlin, S. Samothrakis, and J. A. Walker, "A generalized framework for self-play training," in *2019 IEEE Conference on Games (CoG)*, 2019, pp. 1–8.