Wait a Second: Playing Hanabi without Giving Hints

Markus Eger
Universidad de Costa Rica
markus.eger@ucr.ac.cr

Daniel Gruss
Graz University of Technology
daniel.gruss@tugraz.at

ABSTRACT

Hanabi is a cooperative card game in which communication plays a key role. The game provides an interesting challenge for AI agents, because the game state is only partially observable, and the game limits what players can tell each other. This limit on communication channels is similar to a common scenario in system security research, and has been researched extensively in that context, for example by bypassing a system's isolation by establishing a covert communication channel. Such channels can be established through anything that the sending party can influence and the receiving party can observe, such as photonic emission, resource contention, or latency. In this paper, we present Hanabi agents that utilize timing as a covert channel so effectively that they can eschew the communicative actions provided by the game entirely. In addition to a thorough evaluation of the effectiveness of our approach, and a comparison to other Hanabi agents, we provide its context in the area of security, and an outlook on how it could be related to human behavior in future work.

CCS CONCEPTS

• **Computing methodologies** → **Temporal reasoning**; *Reasoning about belief and knowledge*; • **Applied computing** → **Computer games**.

KEYWORDS

Hanabi, timing, agent design

ACM Reference Format:
Markus Eger and Daniel Gruss. 2019. Wait a Second: Playing Hanabi without Giving Hints. In *Proceedings of FDG (FDG'19)*, Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, Article 4, 7 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Communication between agents plays an essential role in many domains, and thus has lead to the development of many different languages and protocols that agents can use [26]. However, just as communication across channels explicitly provided by the domain is important, there are many opportunities for communication over side channels, such as timing [8].

Hanabi is an award-winning [38], cooperative card game, which heavily relies on communication [4], but at the same time restricts the communicative actions the players have available to them. Therefore, significant research efforts have been undertaken to use the available communication channels to communicate effectively. In this paper, we demonstrate how timing can be used as an additional communication channel in Hanabi, to the point of actually eliminating the need for any other form of communication. Our contribution is two-fold: For one, we present agents that perform extremely well in the game itself, which are based on prior work but exploit the additional communication capacity. We also address some problems associated with using this communication channel. Secondly, and perhaps more importantly, our contribution combines research in game AI, as well as research in IT security in a novel way that challenges traditional assumptions in either field. We hope that this paper will spark a discussion in the games research community on how external communication channels can affect game play, not only in games where all participants are AI agent, but also in the communication between AI agents and human players. We view this contribution as important to the field, since the context in which games are played is often neglected when formulating AI agents, but it is an important aspect of human play.

FDG'19, August 26-30, 2019, San Luis Obispo, CA, USA

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

1.1 Hanabi

In Hanabi, two to five players collaborate to build fireworks, represented by cards in five colors (red, white, blue, green, yellow) with ranks from 1 to 5. Each player is dealt a hand of 4 or 5 cards, depending on the number of players, and holds these cards so that they face away from themselves, i.e. each player sees every other player's cards, but not their own. The table starts out empty, or, conceptually, with one stack per color, each with a 0 on top. Game play proceeds in turns, with each player performing one of the following actions:

- **Play a card:** The player chooses a card in their hand, and puts it on the table. If it is the next card in numerical order on its color's stack, it is placed as the new top card of that stack. Otherwise, the card is discarded and a mistake is noted.
- **Give a hint:** The player chooses any other player, and gives them information about their cards. This information is limited to a *color hint* or a *rank hint*, where the player that receives the hint is told about all cards they hold that have a particular *color* or a particular *rank*, respectively. For example, player *A* may tell player *B* which of player *B*'s cards are red. Giving a hint expends one *hint token* of which there are initially and maximally 8.
- **Discard a card:** The player chooses a card from their hand and removes it from the game, face-up. This action recovers one hint token, up to the maximum of 8.

After playing or discarding a card, the player draws a new card from the draw pile. Game play proceeds until either the draw pile is exhausted (plus one extra round), or the players have made three mistakes, cumulatively. The score of the players is equal to the

Figure 1: The board during a typical game of Hanabi, with five stacks of card on the board, a draw pile, discard pile, and hint tokens. Note that during actual game play the two players' hands would be held by them such they can only be seen by the other player.

number of cards that they played successfully onto the stacks, with a maximum of 25 points, for the five ranks in each of the five colors. Figure 1 shows a typical game state from a game of Hanabi.

2 BACKGROUND AND RELATED WORK

Our work is situated such that it could be directly compared to previous approaches to Hanabi game play, but it also utilizes prior research from a security context. As such, we will discuss the relevant prior work from both areas.

2.1 AI Agents for Hanabi

Because Hanabi has several properties that make it interesting from an AI perspective, several different approaches to playing the game with AI agents have been proposed. Osawa developed several different agents that follow a fixed priority list of actions and use a model of the cooperator's knowledge to avoid providing redundant information [32]. Eger et al. built upon this approach to create AI agents that utilize intentionality in order to play better with human players [15], also providing their implementation as open-source software [14]. Van den Bergh et al. defined variation of rules for agents to follow and exhaustively searched through the space of available rules to find the highest scoring combination [39]. Canaan et al. also searched through a space of rule-based agents, but used an evolutionary approach to enable a larger search space that could not be exhaustively evaluated [10]. Walton-Rivers et al. compared several different approaches – including reimplementations of Osawa's and van den Bergh's agents – with each other, and also developed Monte Carlo Tree Search based agents, and addressed the problem of agents that play with different agent types [43]. Cox et al. build agents that achieve exceptionally high scores in Hanabi, by using the hints as a covert channel for information that would not otherwise be contained within them [13]. This approach is based on hat-guessing games [7], but it has the limitation that it only works with 5 players. Bouzy improved upon this work by extending it to fewer players, in particular addressing its limitations for the game with two players [6]. Finally, Sato et al. used rule-based agents to play with human players, also measuring how long the human players think about their decisions [35]. In addition to the approaches describe by these authors a two-track Hanabi competition has been held at CIG 2018 [42], with one track consisting of agents playing with a copy of themselves, and another – termed "Mixed" – in which the agents would be paired with all other agents. Table 1 shows an overview over the highest average scores from all of these sources.

Authors	Highest Score	Note
Bouzy [6]	24.92	5 players
Bouzy [6]	24.91	Seer
Cox et al. [13]	24.87	5 players
Osawa [32]	24.6	Seer
Hanabi competition [42]	20.57	
Canaan et al. [10]	19.32	
Bouzy [6]	18.98	
Sato et al. [35]	17.8	AI/Human
Eger et al. [15]	17.1	
Osawa [32]	15.85	
Van den Bergh et al. [39]	15.4	
Eger et al. [15]	14.99	AI/Human
Hanabi competition [42]	13.24	Mixed
Canaan et al. [10]	12.38	Mixed
Walton-Rivers et al. [43]	12.14	Mixed, 3 players
Walton-Rivers et al. [43]	11.91	Mixed

Table 1: Highest scores in Hanabi as reported by different authors for their approach. Unless otherwise noted, results are for games with 2 players. Seer refers to agents that can see their own cards, AI/Human refers to games in which one player was human, and Mixed refers to games using different types of AI agents.

2.2 System Security Aspects in Hanabi

Covert channels are a well-known and well-studied problem in system security [8, 27, 31] which gained an increasing amount of attention: First with the advent of personal and business computers running trusted and untrusted code (most prominently today, e.g., JavaScript), and second with the advent of cloud computing where mutually non-trusting parties run code on a shared cloud infrastructure [11, 19, 30, 34, 50]. In this scenario, a malware runs on the same hardware and possibly even software stack as benign software working on secret data. However, due to isolation implemented on the hardware- and software-level, the malware cannot

communicate with the outside world, i.e., it cannot directly transmit the secret information to a remote attacker. A covert channel consists of a side channel that can be influenced by a sending party (i.e., the malware with access to secret data), and observed by a receiving party which is not constrained by the same isolation as the sending party. Hence, the sending party encodes the secret into the side channel signal and the receiving party decodes the signal. This side channel can be any shared resource, e.g., power consumption [21, 28], temperature or light in a room [16, 20, 22, 24], storage or bandwidth contention [9, 33, 47–49], or latency [5, 31]. An illustrative example [22] is a malware running on an air-gapped computer (i.e., no network connection at all), but it exfiltrates data by accessing or not accessing the disk, which implicitly turns on and off the disk activity LED on the outside of the computer. The receiving party observes the LED activity and decodes the message containing the secret information.

From the game rules outlined in Section 1.1, it becomes clear that Hanabi is closely related to this classical problem in system security. The players are isolated, i.e., there is no direct communication allowed other than through the defined set of allowed messages. However, the players are, same as sender and receiver in a classical covert channel, cooperative, i.e., one player might deliberately try to provide another player with information that is difficult or impossible to convey in the defined set of allowed messages. While the problem in system security is largely unsolved, with an abundance of covert channels still available today [18, 50], channels between Hanabi agents are limited due to the highly constrained rule set defining legal and illegal behavior.

3 APPROACH

Our work draws directly from the work by Eger et al. [15]. In their work, they assign each card in the other player's hand a *desired action*, such that:

- A card that is playable should be played.
- A card that has already been played should be discarded.
- A card of which there are still duplicates in the deck may be discarded.
- Otherwise, the card should be kept.

For each card, playing is preferred over being able to discard it, which in turn is preferred over keeping it. Their agents give hints that are meant to be interpreted to follow this priority list, by assuming that each hint gives the highest priority action that can possibly be given. In other words, if a hint implies that a card is playable, even without necessarily guaranteeing it, the card will be played by the agent. Our approach uses the same idea of desired actions, but instead of relying on implicature, the actions themselves are encoded and transmitted over a side channel and can therefore be received and performed by the other agent directly.

3.1 Timing as a Communication Channel

Before we present how the actions that are transmitted are chosen, we will need to determine how we use our side channel, and its capacity, since the capacity determines how many distinct actions we can reliably convey. Our approach is similar to the approach by Ahsan and Kundur [1] and Berk et al. [5]: We use the latency *timing* as the communication channel, with information encoded in *delays*. As a simple example, consider that waiting for 1 minute before performing an action indicates that the other player should play their left-most card, waiting for 2 minutes means that they should play the second card from the left, etc. As the game itself does not enforce a time limit on the players, this would mean that we can convey any (pre-determined) information we want by associating each piece of information with a number. To "send" the information, a player waits that many minutes before performing whichever action they want to perform. The receiving player measures how long it takes from when they performed their action until the other player performs their's and performs a lookup on that number. However, while Hanabi itself does not provide a timing limitation, the Hanabi agent competition held at CIG 2018 [1] limited the time agents had to make their decision to 40ms[2]. By subdividing these 40ms into n *time slices*, we can convey n different messages, exactly as described above.

On modern computers, e.g. a 4 GHz x86 processor, timing measurements are possible with a temporal resolution of around 3 GHz using the processor's timestamp counter (`rdtsc`), or up to 4.6 GHz [37] using an increment loop. Similarly, an increment loop can be used to implement the waiting. This would allow to transmit up to 32 bit s^{-1} just via delaying the delivery of a 1 bit signal (the action).

However, the sender has to spend some cycles on introducing the right delay, resulting in a noisy delay. Similarly, the receiver also has to spend some cycles on observing the delay, resulting in a noisy observation of the delay. Additionally, the timing side channel comes another limitation: There are outside factors, such unrelated system activity, scheduling, or interrupts that may cause the agents to wait slightly longer, or measure a longer time, depending on the technique used for waiting and measurement.

Hence, for various reasons, potentially the wrong message might be sent or wrongly received. The standard approach to cope with this problem is redundancy [5, 29]. A simple idea is to split the time axis into discrete time slots by requiring requiring the lowest k bits to be zero [8], or other values used for instance for error-detection or error-correction encoding [29]. If the bits do not have a valid value, the receiving agent can deduce what the most likely value was. How many of the available bits are used for redundancy forms a trade-off between the transmission bit error rate and the transmission rate.

For Hanabi, we opted to encode only 10 different messages (less than 4 out of 32 bits), yielding a virtually error-free transmission. The 10 different messages are the following: five, encoded by waiting between 0 and 4 time slices, corresponding to playing cards, and five, encoded by waiting between 5 and 9 time slices, corresponding to discarding cards, with one action for each card in the player's hand. Note that this means that the other player is only told *which* of their cards to play, but not *what* that card is. We will discuss a limitation of this approach later, and what we can do to mitigate it. With 10 different messages, the time difference between two adjacently encoded messages is 4 ms, which provides reasonable tolerance for delays on the sending side and can be reasonably

[1] https://project.dke.maastrichtuniversity.nl/cig2018/competitions/#hanabi
[2] This time limit has since been increased to 1s

accurately measured on the receiving side. We will discuss the relation between timing and transmission error rate in more detail in section 4.

3.2 Our Agent

Using the encoding of messages presented above, our agent performs the following steps:

(1) Store current time as t_1
(2) Determine how much time δt has passed since the last turn
(3) Decode message as round($\frac{\delta t}{4\text{ms}}$), and determine the corresponding play/discard action a
(4) Determine which action b the other player should perform, and which wait time t this corresponds to
(5) Sleep until the difference between the current time and t_1 is equal to t
(6) Perform the action a

Note that, to determine which action a the agent should perform, they *round* to the nearest whole time slice, thus requiring an accuracy of ±2ms for the wait time, as well as for time measurement. On the other hand, since one of the actions has the encoding "Wait for zero seconds", steps 1 to 4 must not need more than 2ms, i.e. the agent has to make its decision within 2ms[3]. Because of this time limit, our agent uses a very simple heuristic to determine what it wants the other player to do, which is directly based on Eger et al's priority list: If the other player has playable cards, they should play the lowest ranked playable card they have. Otherwise, if they have a card that are no longer needed, they should discard that card. Otherwise, they should discard the highest ranked card that is not a 5 (since there is only one copy of every 5 available). The agent finds the card in the other player's hand that has the highest priority action associated with it according to this list, and encodes the appropriate message as described above.

3.3 Failure Cases

As we will discuss in section 4 below, our agents as described above already perform reasonably well in Hanabi, but there are still some limitations to consider. One problem is that our agents play too efficiently, in a way: When agent b tells agent a to play a card, they may do so by playing a card themselves that they were just told to play by agent a. However, they may have just told agent b to play the exact same card. For example, if agent a and agent b both have the red 1 at the beginning of the game, agent a tells agent b to play that card by waiting the appropriate amount of time. Agent b, in turn, sees that agent a has a playable 1 and tells them to play it by *playing their own red* 1. This will cause agent a to *also* play the red 1, resulting in a mistake. Such situations are common enough that addressing them is a worthwhile endeavor. One easy solution is to avoid playing two cards in a row: If the other player just played a card, and the agent would do so, too, they will instead give a random hint (if any are available). Since our agents do not interpret, or even need, hints, it is irrelevant which hint is given, the action is basically just used as a "skip". We call this variant of our agents the *careful* one, as it avoids making plays that could result in a mistake, even though it does not know for sure. As we will show below, this small modification leads to significantly better scores, with the remaining main limitation being communication errors.

[3]It is possible to mitigate this, e.g. by using time slices of 2ms, and always waiting for at least 20ms. The drawback of this technique is that it then requires an accuracy of ±1ms.

3.4 More Than Two Players

Much of prior work focused either on the two-player case, or utilized larger player counts to encode information into the hints, such as the work by Cox et al. [13]. Our approach, on the other hand, while originally devised for two players, naturally extends to any player count. The only requirements are that the players can measure the length of individual player's turns, or, in terms of the game, are informed of every other player's action as it is performed. Then, instead of measuring the time that elapsed since the last turn the agent performed themself, they measure how much time has passed since the action two seats to their right, to determine how long the player to their immediate right waited. The agent, in turn, then informs the player to their immediate left of what action they should perform. Note that games with more players would open up more options for which other player to communicate with, but since each player *needs* to perform some action on their turn, it would rarely be useful not to tell the next player what to do.

3.5 Isn't This Cheating?

Since the technique we present in this paper seems to push the rules of Hanabi in a major way, one might object that we are "cheating" by utilizing time as a communication medium. We actually have three responses to these objections:

First, the rules do not explicitly mention the use of timing, so one could say that we are not violating the letter of the law. We will note, however, that the rules say, quote "Communication (and non communication) between the players is essential to Hanabi. If you follow the rules closely, *you can only communicate with your teammates when you give them information placing a blue token*." (Emphasis ours) While this seems to imply that using any communication channel other than hint giving is forbidden, the game is built around all sorts of implicit communication. For example, this would also ban players from performing actions that imply information, or performing such deductions, all of which are actually an integral part of game play [4] The rules address this by subsequently relaxing the restriction on communication by stating "However, you can play whichever way suits you best: set your own rules regarding communication."

Second, *human players* routinely use timing, consciously or not, to make decisions in the game. When a player draws a card, and the other player gives them a hint about the new instantaneously after seeing it, this hint will be interpreted differently than the exact same hint that is only given with hesitation. Our agents do not perform any actions that human players do not also perform, they just make more effective use of them. Note, however, that human non-verbal communication can be very effective as well: The game *The Mind* [44], which works like Hanabi in that it tasks players with collaboratively playing cards in increasing order, but does not

[4]For an example for such an action, see the "Sarcastic Discard" in this Hanabi conventions document: https://github.com/Zamiell/hanabi-conventions/blob/master/Reference.md The description of this move even states that "it communicates to the other player that they 100% have the discarded card", which would be a violation of the strict interpretation of the rules.

afford them any way of communicating the contents of their hands, relies entirely on timing, hesitation, and other cues for game play.

Third, and perhaps most importantly, we actually believe that whether or not our agents are considered to be "cheating" is irrelevant. The purpose of our work was not to build agents to submit to the Hanabi competition (where the organizers could justifiably say that we are violating the spirit of the competition), but rather to present a novel approach to communication in games. To this end, we use the most extreme form of our approach that is viable by eschewing all other communication, to show just how powerful this technique is. However, it is also important to consider the implications of our work: As mentioned above, *human players* utilize timing when communicating with each other, so it would behoove us as a community to incorporate it into our models. Of course, we make no claims that our agents would be suitable for play with human players, but we demonstrate the potential value timing can have as a form of communication. In fact, Sato et al. have already shown that the timing exhibited by human players is significant [35], even though they have not yet been able to actually utilize it to improve the agent's score. In section 5 we will discuss future applications with human players in more detail.

# Players	Time/turn	Careful?	Average Score (std dev)
2	10ms	No	5.06 (3.96)
2	10ms	Yes	10.5 (6.31)
2	20ms	No	19.1 (5.5)
2	20ms	Yes	23.1 (2.2)
2	40ms	No	19.4 (5.34)
2	40ms	Yes	23.2 (2.1)
2	100ms	No	19.3 (5.45)
2	100ms	Yes	23.2 (2.1)
3	40ms	No	19.14 (6.05)
3	40ms	Yes	23.3 (1.94)
4	40ms	No	19.08 (5.7)
4	40ms	Yes	22.64 (1.95)
5	40ms	No	18.36 (5.85)
5	40ms	Yes	21.7 (1.9)

Table 2: A summary of simulation results for different numbers of players, time given to the players for each turn, and the basic and careful agent variants.

4 RESULTS

We have performed several experiments to evaluate different aspects of our approach. In each of these experiments the agents played 10000 games with the same settings. There are two responses of interest that we measured: The score the agents achieved in the game, and the error rate of the timing-based communication. Of course there is a relation between these two, since a high error rate will cause the agent to play or discard the wrong card, which leads to a reduced score in most cases. As the main result of our work, we present how our agent performs in the CIG competition setting, with 40ms per turn, and 2 players. In this scenario, our agent as described above reaches an average of 19.4 (std dev: 5.34) points in the basic version, and 23.2 (std dev: 2.1) points in the careful version, which is a highly competitive score, since the winner of the actual competition scored 20.57 points on average. Figure 2 shows the histogram of scores for the two cases. As can be seen, when using the careful variant, our agent reaches the maximum of 25 points in almost 40% of the games. There are two main causes for the agent to score less: Communication errors caused by unrelated system activity or scheduling, and unfavorable shuffling of the deck with regards to the order the cards are played in.

To determine the effect of communication errors on the score, we reduced the time given to the agents, which in turn increased the error rate of miscommunication. We measured the error rate by comparing the action the agent communicated to the other player that they should perform with the action that was actually performed. In the standard setting, with 40ms per turn, requiring measurements within ±2ms, 1.46% of the messages were miscommunicated. Figure 3 shows how the error rate relates to the time limit per turn. As expected, lower times result in higher error rates, as the influence of other system activity and scheduling makes the time measurements less reliable. Conversely, with higher error rates, the scores obtained by the agents decreased significantly, with an average of 5.06 (std dev: 3.96) points when the agents are given 10ms per turn. Notably, though, even though the scores are low, communication over the covert channel was still possible in this case, albeit with an error rate of 47%. On the other hand, giving the agents 100ms did not improve their scores compared to the case where they had 40ms in a statistically significant matter. Additionally, we performed an experiment where the agents communicated the intended wait time directly using a global variable, to simulate the case in which the communication error rate is 0%. Even in this case, the average score was 23.2 (std dev: 2.1), with no statistically significant difference to the scores by agents that used timing as a communication channel with 40ms per turn.

Finally, we also performed experiments where more than two agents played the game by communicating the desired move to the next player in turn order, as described above. The main challenge in this scenario is that the last few cards may be distributed unfavorably among the players, and it would require a certain amount of planning to determine the optimal order to play them in, as well as which players should play cards, and which should not, in order to prolong the game to allow the players with playable cards to actually play them. Since our agent does not perform any such reasoning, the score it obtains for higher player counts is slightly lower than for the 2-player game. For example, for 5 players, the average score was 21.7 (std dev: 1.9). Table 2 shows a summary of all results from our experiments.

5 CONCLUSION

We have presented an approach to the cooperative card game Hanabi that utilizes timing as a covert communication channel between the players. Our agents completely eschew the built-in communication mechanism of the game and instead encode all information the other player needs to know into how quickly they perform their own action, by dividing their own turn into time slices where each time slice represents one action for the other agent to perform. The agents then perform their own action, which was communicated to

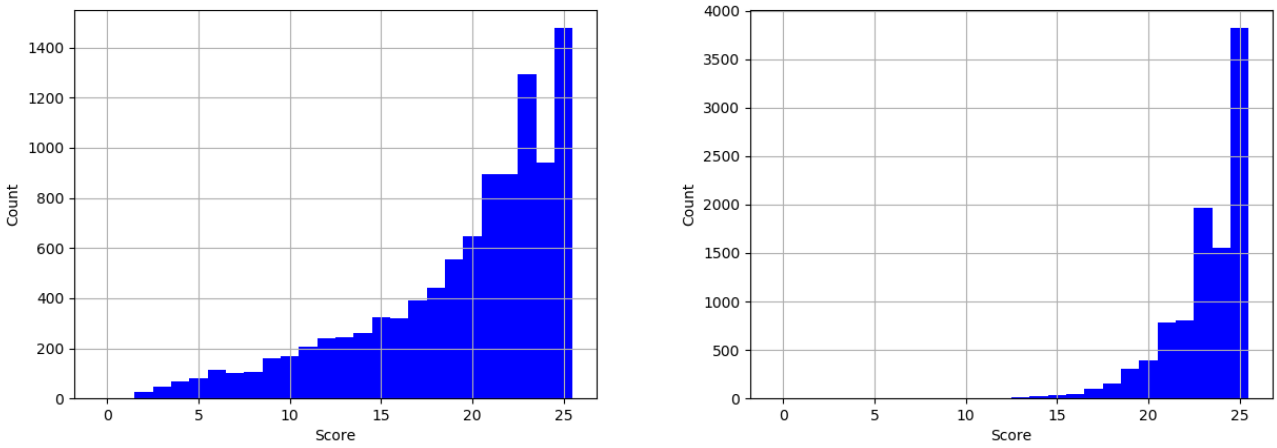

Figure 2: Histogram of scores obtained by our agents in games with 2 players, and 40ms per turn. The left graph shows the scores of the basic agent, and the right graph shows the scores of the careful agent.

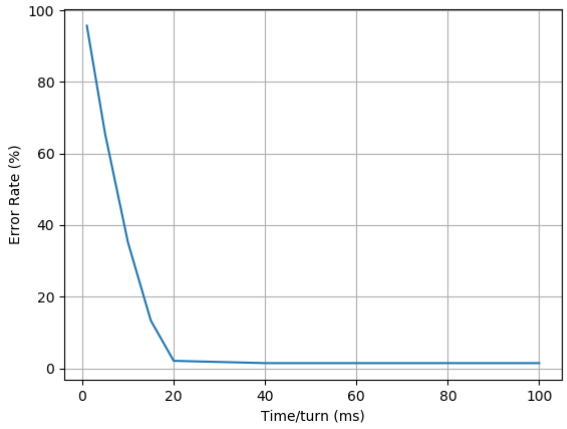

Figure 3: Communication error rate for different values of time given to the agents for each turn.

them by the other player, in the time slice corresponding to the action they have determined the other player should perform. Based on the time limit given by the CIG Hanabi competition of 40ms per turn, we have opted to encode 10 different action, requiring 10 different time slices of 4ms each, which can be achieved reliably even in the presence of other system activity and unpredictable task scheduling. Using this approach, our agents reach an average of 19.4 points without giving any hints, which can be improved upon to reach an average of 23.1 points by using hints to avoid the most common failure case of the two players playing the same card successively.

While our agents only need 40ms to achieve a high score, and often even reach the maximum possible score, the CIG Hanabi AI competition has recently increased the time limit to 1s per turn, to allow for more complex strategies. As we have outlined above, there are methods to transmit up to 32 bits per second using high precision timing measurements, which would open up new possibilities. For example, in addition to telling the other player which card they should play by index, it would also be possible to tell them which card it is, to avoid the problem of two agents playing the same card successively. Alternatively, the additional bits could be used to transmit information external to the game of Hanabi, such as playing a game of Connect Four over the covert channel inside the Hanabi game. While this may not seem very useful at first glance, video games have already been proposed as surveillance- and censorship-resilient carriers for covert communication [41].

As we discussed, a strict interpretation of the rules may indicate that our approach is not permitted, but the elimination of such covert channels is a non-trivial problem, for which many approaches have been proposed [2, 12, 23, 25, 51, 52], including the removal of the ability to measure time [2, 3, 12, 17, 23, 25, 40, 46]. While this would make our concrete implementation unusable, there are also other timing primitives that could be used instead [36, 37, 45]. Additionally, *any* kind of state change that can be observed directly or indirectly by the receiving party makes data transmission possible. In fact, the approach by Cox et al. [13], while likened by them as a hat-guessing game strategy, actually constitutes a covert channel that encodes extra information into the regular game actions.

Finally, we want to reemphasize that while our agents rely on high-precision timing, the general approach of communicating information over delays is not unique to computers. Hesitation plays a key role in non-verbal communication between humans, and a better understanding of how to interpret it can greatly benefit human-computer interaction. The goal of our work was to demonstrate the possibilities of such communication, but the interpretation of the more subtle, human-based timing of communication is left for future work.

REFERENCES

[1] Kamran Ahsan and Deepa Kundur. 2002. Practical data hiding in TCP/IP. In *Workshop on Multimedia Security at ACM Multimedia*.

[2] Aslan Askarov, Danfeng Zhang, and Andrew C Myers. 2010. Predictive black-box mitigation of timing channels. In *ACM Conference on Computer and Communications Security*.

[3] Amittai Aviram, Sen Hu, Bryan Ford, and Ramakrishna Gummadi. 2010. Determinating timing channels in compute clouds. In *ACM Cloud Computing Security Workshop*.

[4] Antoine Bauza. 2010. Hanabi. https://boardgamegeek.com/boardgame/98778/hanabi

[5] Vincent Berk, Annarita Giani, George Cybenko, and N Hanover. 2005. Detection of covert channel encoding in network packet delays. *Rapport technique TR536, de lUniversité de Dartmouth* 19 (2005).

[6] Bruno Bouzy. 2017. Playing Hanabi Near-Optimally. In *Advances in Computer Games*. Springer, 51–62.

[7] Steve Butler, Mohammad T Hajiaghayi, Robert D Kleinberg, and Tom Leighton. 2009. Hat guessing games. *SIAM review* 51, 2 (2009), 399–413.

[8] Serdar Cabuk, Carla E Brodley, and Clay Shields. 2004. IP covert timing channels: design and detection. In *Conference on Computer and Communications Security*. ACM.

[9] Serdar Cabuk, Carla E Brodley, and Clay Shields. 2009. IP covert channel detection. *ACM Transactions on Information and System Security (TISSEC)* 12, 4 (2009), 22.

[10] Rodrigo Canaan, Haotian Shen, Ruben Torrado, Julian Togelius, Andy Nealen, and Stefan Menzel. 2018. Evolving Agents for the Hanabi 2018 CIG Competition. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 1–8.

[11] Yanpei Chen, Vern Paxson, and Randy H Katz. 2010. What's new about cloud computing security. *University of California, Berkeley Report No. UCB/EECS-2010-5 January* 20, 2010 (2010), 2010–5.

[12] David Cock, Qian Ge, Toby Murray, and Gernot Heiser. 2014. The last mile: An empirical study of timing channels on seL4. In *ACM Conference on Computer and Communications Security*.

[13] Christopher Cox, Jessica De Silva, Philip Deorsey, Franklin HJ Kenter, Troy Retter, and Josh Tobin. 2015. How to make the perfect fireworks display: Two strategies for Hanabi. *Mathematics Magazine* 88, 5 (2015), 323–336.

[14] Markus Eger and Chris Martens. 2017. A Browser-based Interface for the Exploration and Evaluation of Hanabi AIs. Tech Demo at Foundations of Digital Games. (2017).

[15] Markus Eger, Chris Martens, and Marcela Alfaro Córdoba. 2017. An Intentional AI for Hanabi. In *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*. IEEE.

[16] Julie Ferrigno and Martin Hlavác. 2008. When AES blinks: introducing optical side channel. *IET Information Security* 2, 3 (2008), 94–98.

[17] Bryan Ford. 2012. Plugging side-channel leaks with timing information flow control. In *Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing*.

[18] Qian Ge, Yuval Yarom, David Cock, and Gernot Heiser. 2018. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. *Journal of Cryptographic Engineering* 8, 1 (2018), 1–27.

[19] Annarita Giani, Vincent H Berk, and George V Cybenko. 2006. Data exfiltration and covert channels. In *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense V*, Vol. 6201. International Society for Optics and Photonics, 620103.

[20] Mordechai Guri, Matan Monitz, Yisroel Mirski, and Yuval Elovici. 2015. Bitwhisper: Covert signaling channel between air-gapped computers using thermal manipulations. In *Computer Security Foundations Symposium (CSF)*. IEEE.

[21] Mordechai Guri, Boris Zadov, Dima Bykhovsky, and Yuval Elovici. 2018. PowerHammer: Exfiltrating Data from Air-Gapped Computers through Power Lines. *arXiv:1804.04014* (2018).

[22] Mordechai Guri, Boris Zadov, and Yuval Elovici. 2017. LED-it-GO: Leaking (a lot of) Data from Air-Gapped Computers via the (small) Hard Drive LED. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer.

[23] Wei-Ming Hu. 1992. Reducing timing channels with fuzzy time. *Journal of computer security* (1992).

[24] Michael Hutter and Jörn-Marc Schmidt. 2013. The temperature side channel and heating fault attacks. In *International Conference on Smart Card Research and Advanced Applications*. Springer.

[25] Boris Köpf and Markus Dürmuth. 2009. A provably secure and efficient countermeasure against timing attacks. In *22nd IEEE Computer Security Foundations Symposium*.

[26] Yannis Labrou, Tim Finin, and Yun Peng. 1999. Agent communication languages: The current landscape. *IEEE Intelligent systems* 2 (1999), 45–52.

[27] Butler W Lampson. 1973. A note on the confinement problem. *Commun. ACM* 16, 10 (1973), 613–615.

[28] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. 2008. *Power analysis attacks: Revealing the secrets of smart cards*. Vol. 31. Springer Science & Business Media.

[29] Clémentine Maurice, Manuel Weber, Michael Schwarz, Lukas Giner, Daniel Gruss, Carlo Alberto Boano, Stefan Mangard, and Kay Römer. 2017. Hello from the other side: SSH over robust cache covert channels in the cloud. In *Proceedings of the Network and Distributed System Security Symposium*.

[30] Jonathan Millen. 1999. 20 years of covert channel modeling and analysis. In *Symposium on Security and Privacy*. IEEE.

[31] Steven J Murdoch and Stephen Lewis. 2005. Embedding covert channels into TCP/IP. In *International Workshop on Information Hiding*. Springer.

[32] Hirotaka Osawa. 2015. Solving Hanabi: Estimating Hands by Opponent's Actions in Cooperative Game with Incomplete Information. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[33] Dag Arne Osvik, Adi Shamir, and Eran Tromer. 2006. Cache attacks and countermeasures: the case of AES. In *Cryptographers' Track at the RSA Conference*. Springer.

[34] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. 2009. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and Communications Security*. ACM.

[35] Eisuke Sato, Takuya Kato, and Hirotaka Osawa. 2018. Development and evaluation of the game agent to add opponent thinking time as indicator of strategy in Cooperative game Hanabi. *Game Programming Workshop* 2018 (2018), 30–34.

[36] Michael Schwarz, Clémentine Maurice, Daniel Gruss, and Stefan Mangard. 2017. Fantastic Timers and Where to Find Them: High-Resolution Microarchitectural Attacks in JavaScript. In *Financial Cryptography and Data Security*.

[37] Michael Schwarz, Samuel Weiser, Daniel Gruss, Clémentine Maurice, and Stefan Mangard. 2017. Malware guard extension: Using SGX to conceal cache attacks. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer.

[38] Spiel des Jahres. 2013. Spiel des Jahres Award 2013. http://www.spieldesjahres.de/en/hanabi

[39] Mark JH van den Bergh, Anne Hommelberg, Walter A Kosters, and Flora M Spieksma. 2016. Aspects of the cooperative card game Hanabi. In *Benelux Conference on Artificial Intelligence*. Springer, 93–105.

[40] Bhanu C Vattikonda, Sambit Das, and Hovav Shacham. 2011. Eliminating fine grained timers in Xen. In *ACM Cloud Computing Security Workshop*.

[41] Paul Vines and Tadayoshi Kohno. 2015. Rook: Using video games as a low-bandwidth censorship resistant communication platform. In *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society*. ACM, 75–84.

[42] Joseph Walton-Rivers. 2018. Fireworks agent competition. http://hanabi.aiclash.com.

[43] Joseph Walton-Rivers, Piers R Williams, Richard Bartle, Diego Perez-Liebana, and Simon M Lucas. 2017. Evaluating and modelling Hanabi-playing agents. In *Evolutionary Computation (CEC), 2017 IEEE Congress on*. IEEE, 1382–1389.

[44] Wolfgang Warsch. 2018. The Mind. http://www.nsv.de/spielregeln/TheMind_GB.pdf.

[45] John C Wray. 1992. An analysis of covert timing channels. *Journal of Computer Security* 1, 3-4 (1992), 219–232.

[46] Weiyi Wu, Ennan Zhai, Daniel Jackowitz, David Isaac Wolinsky, Liang Gu, and Bryan Ford. 2015. Warding off timing attacks in Deterland. *arXiv:1504.07070* (2015).

[47] Zhenyu Wu, Zhang Xu, and Haining Wang. 2012. Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.. In *USENIX Security Symposium*.

[48] Zhenyu Wu, Zhang Xu, and Haining Wang. 2014. Whispers in the Hyper-space: High-bandwidth and Reliable Covert Channel Attacks inside the Cloud. *IEEE/ACM Transactions on Networking* (2014).

[49] Yuval Yarom and Katrina Falkner. 2014. Flush+Reload: A High Resolution, Low Noise, L3 Cache Side-Channel Attack. In *USENIX Security Symposium*.

[50] Sebastian Zander, Grenville Armitage, and Philip Branch. 2007. A survey of covert channels and countermeasures in computer network protocols. *Communications Surveys & Tutorials* 9, 3 (2007), 44–57.

[51] Yinqian Zhang, Ari Juels, Alina Oprea, and Michael K. Reiter. 2011. HomeAlone: Co-residency Detection in the Cloud via Side-Channel Analysis. In *IEEE Symposium on Security and Privacy*.

[52] Ziqiao Zhou, Michael K. Reiter, and Yinqian Zhang. 2016. A software approach to defeating side channels in last-level caches. In *ACM Conference on Computer and Communications Security*.