Imperfect-Information Games on Quantum Computers: A Case Study in Skat

Ulrich Armbrüster
Atos Future Maker Research Community
Vienna, Austria

Stefan Edelkamp
Faculty of Mathematics and Physics, Charles University
Prague, Czech Republic

Gabriel Maresch
Faculty of Mathematics and Geoinformation, TU Wien
Vienna, Austria

Erik Schulze
Eviden Germany GmbH
Munich, Germany

ABSTRACT

We demonstrate how quantum computing can provide a promising framework for addressing imperfect information games, using the popular German card game Skat as a case study. Our approach employs quantum registers to encode the game's hidden and public information, along with tailored quantum gates to model the game's progress while respecting its rules. To evaluate player decisions, we introduce a score operator that projects the quantum state onto the winning subspace, enabling the estimation of winning probabilities via quantum algorithms such as quantum counting. This allows the efficient exploration of alternative strategies and potential improvements in computational speed compared to classical methods. By maximizing the payoff function, our framework yields actionable recommendations for optimal play. As classical computation faces severe limitations due to the exponential complexity of the game tree, we discuss the specific structural properties of Skat that may give rise to a quantum advantage once the problem size surpasses classical feasibility.

KEYWORDS

Skat, Quantum Algorithms, Imperfect-Information Games

ACM Reference Format:
Ulrich Armbrüster, Stefan Edelkamp, Gabriel Maresch, and Erik Schulze. 2026. Imperfect-Information Games on Quantum Computers: A Case Study in Skat. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/PYJU2489

1 INTRODUCTION

Game theory has already had major impacts and influence in many different non game-related sectors. One of the best examples known among games with so-called imperfect information, where the players only have partial knowledge of the current state of the game and its variables, are economic models of labor and management negotiation. Some of the predicted economic variables are only known partially via probabilistic distributions. Therefore it is highly desirable, that, despite improving the models for predicting economic variables, the uncertainty leads to an equilibrium state of decisions that corresponds to the formal game-theoretical solution of the system. While it is yet to be determined, whether the concrete problem is actually too complex for a classical solution, there are some more general aspects among imperfect-information games that are to be considered in advance.

More specifically speaking, it has been shown in the past that n-player imperfect information games may turn out to be NP-hard [1], which makes them difficult to compute classically. In order to solve it formally, or at least get relatively close to the game-theoretical solution, we might get some significant advantage using the computational power of entangled quantum systems.

In recent years, game theory has developed in many different ways, including the use of Artificial Intelligence and Machine Learning as well as some other emerging technologies. Back in the days, when computers were not yet able to beat humans in the majority of complex strategical and combinatorial games, quantum computers were thought to remain just a prospect for the foreseeable future. This has changed drastically over the last couple of years with new theoretical concepts and hardware implementations for real quantum computers.

Generally speaking, a gate-based quantum computer is merely a device that can manipulate quantum states in a unitary way. Nowadays, quantum computers are considered to be able to solve a whole bunch of combinatorial problems that, a couple of years ago, some people thought might never be solved [25]. Some kinds of complex combinatorial problems are equivalent or can be mapped to problems in mathematical game theory. With game theory, decision processes can be modeled and this raises the question about the individual strategies a player or a party should follow in order to optimize the outcome of a predefined target function. Since some games, in general, can be incredibly difficult to be fully calculated, new methods of computation might be necessary in order to solve them as large optimization problems.

Card game play is a recent objective of research for making choices with imperfect information, and current card game solvers are beginning to challenge human supremacy. Notably we find research on Bridge [4] and Skat [6]. In this work we will show that quantum computing is suited to deal with the large state spaces in incomplete information games using the Skat as a case study. The purpose of this paper is to make use of the power of quantum computation to actually *solve* the classical game. Therefore, we must clarify what constitutes a solution for a game of this kind.. Thus, let us have a look on the classical game theory first of all.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/PYJU2489

Generally speaking, imperfect-information games are solved nowadays using different methods. Finding Nash equilibria via Lemke-Howson [17] algorithm, modeling it as a Bayesian game where the uncertainty is mapped to a probabilistic distribution [14] or the so-called Counterfactual Regret Minimization [27], where the regret function of alternative strategies is minimized, are among the most established to find the best strategies for games of this category. However, deep-learning algorithms and Monte-Carlo tree-search have become increasingly popular among more complex games lately. Which of those methods is the most beneficial to the problem depends on the problem itself, the resources that are available, and the desired precision of the solution.

2 SKAT IN CLASSICAL GAME THEORY

Skat [24] as a representative card game poses many combinatorial challenges such as cooperative and adversarial behaviors (among the players), randomness (in the deal), and partial knowledge (due to hidden cards). With 3 players and 32 cards it is more compact than Bridge, but with the additional uncertainty of two cards in the Skat, full of subtleties. At the beginning of a game, each player receives 10 cards, which are hidden to the other players. The remaining two cards, called *Skat*, are placed face down on the table. After distributing the cards, we have 4 stages of play: a) bidding, where the players are trying to become declarer by announcing lower bounds to the contract; b) game selection, where –after taking the Skat— the winner of the bidding decides the game to be played; c) skat putting, where declarer discards two cards (if taken) and announces game to be played; and d) trick-taking, where in clockwise order each player places a card on the table. The winner of the trick put them onto own stack and continues with issuing the next one.

Research work in Computer Skat includes [6, 8, 10, 11, 19, 22, 23]. The *double-dummy Skat solver*, a fast open card Skat game solver [16], was extended to partial-observable game play using Monte-Carlo sampling. The lack of knowledge information exchange between the players, however, has led to efforts to apply machine learning [3, 13, 15, 22, 23, 26]. Later on, knowledge-based paranoia search [9] and a weighted sampling of the belief space were used [5, 22]. Together with the concept of hope cards, it was possible to once beat a top player in a(n inofficial) match [11].

Fig. 1 provides a rough characterization of games in combinatorial, strategical and fortune. While chess and other purely combinatorial board games are mostly complex because of the *de facto* unlimited amount of moves, the difficulty in certain card games derives from another origin – namely the lack of perfect information. The game is only partially observable, and reasoning has to be done in the so-called belief space, the set of all possible worlds. In addition to sampling, one option is $\alpha\mu$ [18], some of its refinements [2], or $\alpha\beta$-search for graphs with partially ordered values [18].

Disregarding some peculiarities as the bidding process and the putting of the Skat (for simplicity), a game consists of 30 cards being played in a certain order. The number of permutations of these 30 cards gives an upper bound for the possible games that have to be computed, but it shrinks down a lot due to the game rules that disallow many of those permutations. The solution of the game would be a permutation of cards that is allowed by the rules and that leads to the highest value for the payoff function of

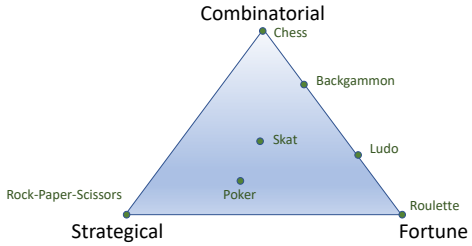


Figure 1: Skat, classified among other games in terms of strategy, combinatorics and fortune in Bewersdorff triangle.

the game (see later), while being in a stable equilibrium with all the players being satisfied with their respective choices according to the expectation value.

That means, the major limiting factor for the computation of the solution is the tree search. More precisely, it is the number of winning paths for all possible distributions that leads to a huge combinatorial problem. According to our estimations, it would take up to 8.7 Mio. years to compute all these paths classically and therefore we suggest making use of new computing technologies in order to tackle this part separately.

3 TOWARDS QUANTUM SKAT

While you see everything that is going on on a chessboard, you are not able to look into your opponent's cards. Thus, you have to make educated guesses to maintain a reasonable basis for your decisions in the game. In fact, the game can be regarded to be in a superposition of all possible distributions at the beginning, shrinking down to a smaller volume as the game proceeds.

While we study Skat, the results will be transferable to other card games and games with incomplete information. Without going too much into detail about the rules and the gameplay of Skat itself, let us have a look on the bare numbers, giving us a hint on the quantum feasibility.

In quantum mechanics, the general quantum state of a qubit can be represented by a linear superposition of two orthonormal basis states. We use the standard Dirac braket notation as in [21].

In general, n qubits are represented by a state vector in 2^n dimensional Hilbert space. For the initial state ϕ_{ini} in a card game, we have

$$|\Phi_{\text{ini}}\rangle = \frac{1}{\sqrt{\mathcal{N}}} \cdot (|\Phi_1\rangle + |\Phi_2\rangle + \cdots + |\Phi_{\mathcal{N}}\rangle)\,. \tag{1}$$

while each ϕ_i represents one of the possible distributions of cards and $\mathcal{N}$ is the number of possibilities that is yet to be determined and serves as a normalization constant.

We will derive a concept for description and encoding of the problem in terms of quantum states, so that we can define a gateset that properly expresses the progress of the game.

4 ENCODING

In principle, there are many different ways to map a given distribution of cards to a state vector. In addition, the cards may serve different purposes depending on the evolution of the game. While it remains to be seen if the following approach is practical, it certainly is *possible*, to fix a specific permutation of the 32 cards once and for

Table 1: Encoding card state $|q_{j_0}q_{j_1}\rangle\,|q_{j_2}q_{j_3}q_{j_4}\rangle$, auxil. omitted.

position	encoded state
$j_0 j_1$	player qubits: card with fore-, middle-, rearhand, skat
$j_2 j_3 j_4$	location qubits: hand, table, stack

all (e.g. in a non-increasing way according to the trick-taking partial order). Thus, our state vector takes the form $|\Psi\rangle = \bigotimes_{i=1}^{32} |\psi_i\rangle$, where each $|\psi_i\rangle$ contains all relevant information in the game about the card with positional index i. To capture this information, we construct these kets as

$$|\psi_i\rangle = |\psi_i^{(1)}\rangle\,|\psi_i^{(2)}\rangle \ldots |\psi_i^{(n_i)}\rangle\,, \tag{2}$$

where each $|\psi_i^{(j)}\rangle$ is a single-qubit state. We choose n_i, the number of *card qubits*, to be constant across all cards and denote this number by n_c. Thus, each $|\psi_i\rangle$ is an element of a 2^{n_c}-dimensional Hilbert space $\mathcal{H}_c$ and the full initial state $|\Psi\rangle$ is an element of the 2^{32n_c}-dimensional Hilbert space $\mathcal{H} = (\mathcal{H}_c)^{\otimes 32}$.

We refer to the basis states of $\mathcal{H}_c$ in the computational basis as *card states*. Each card state can be represented as an n_c-digit binary number. This encoding of card states will prove crucial in keeping track of the evolution of the game. In principle, one could simply list all states in which a card could possibly be and assign binary numbers to these states, but there are more suitable ways of achieving this. For Skat we can decompose the card state into

$$|q_{j_0} \ldots q_{j_{n_c-1}}\rangle = |\text{player}\rangle\,|\text{location}\rangle\,|\text{auxiliary}\rangle\,. \tag{3}$$

When using this encoding, we will occasionally rename the q_{j_2} qubits as q_t and refer to them as *table*-qubits, and the q_{j_3} qubits as *stack*-qubits q_s. The q_{j_4} qubits state will dictate whether suit must be followed. In our approach, however, the latter will not be discussed nor used. All auxiliary qubits are collected in $|\text{auxiliary}\rangle$.

The typical fate of a card will be, that it stays in the hand of a player $|\alpha\rangle\,|000\rangle$, then at some point gets played and put to the table as part of a specific sequence $|\alpha\rangle\,|01q_{j_4}\rangle$ and finally ends up in the stack of a - possibly different - player $|\alpha'\rangle\,|11q_{j_4}\rangle$.

It is important to note that the particulars of the card state encoding determine how an actual implementation in a quantum circuit will look like, but do not matter from an abstract point of view. Also, this encoding is not optimized in any way, and its redundancies are obvious. Nevertheless, it is easy to work with and so we will stick with it for now. For Skat we expect $n_c \leq 5 + 3 = 8$, a total of $8 \cdot 32 = 256$ qubits are, needed. Clever card state encoding could possibly reduce this to $5 \cdot 32 = 160$ qubits or even less.

4.1 Game Progress and Quantum Gates

To manipulate quantum states we use quantum gates. There are different types of quantum gates, like single-qubit gates, which can e.g. flip the state of a qubit, as well as allowing superposition states to be created. Then, there are also 2-qubit gates. These allow the qubits to interact with each other and can be used to create quantum entanglement: a state of two or more qubits that are correlated in a way that can not be obtained using classical bits. In contrast to classical gates, all quantum gates are unitary, which means in particular, that they are reversible.

It is a non-trivial task, to single out all state vectors $|\phi_i\rangle \in \mathcal{H}$ which correspond to a valid initial distribution of cards in the game of Skat. In particular, $|\Phi_{\text{ini}}\rangle$ will *not* be a superposition of all basis states, but only on a relatively small subset. We follow the approach presented in [20] and construct a (deterministic) Boolean function $f_{\text{valid}}\colon 2^{\dim \mathcal{H}} \to \{0,1\}$ to prepare the initial state (1). f_{valid} maps a distribution to 1 if it corresponds to a valid card distribution, otherwise to 0. Let us denote a corresponding unitary operator, which maps $|0\ldots0\rangle$ to $|\Phi_{\text{ini}}\rangle$ by U_{ini}.

On the qubit level, each step in the evolution of the game is effected by a unitary operator acting on the state vector. The rules of the game dictate how these unitaries look.

4.2 Playing one card

A basis operation in any card game is playing a card, i.e. transferring a card from a player's hand to the table. There is a *caveat*: A card can only be placed on the table if it currently belongs to the hand of a player. This means that we have to condition the transformation of the card state on the fact that we deal with a card that is in the hand of a player.

With an encoding as proposed in Table 1, this amounts to changing the card state (omitting the inactive suit-following qubit), $|\text{location}\rangle = |00\rangle$ to $|\text{location}\rangle = |01\rangle$, which amounts to the application of a single Pauli-X, subject to the condition of the table qubit being in state $|0\rangle$. Here a problem arises: a qubit cannot be both control *and* target of a controlled operation. Thus, we have to make use of one auxiliary qubit per card. The corresponding unitary has to transform $|00\rangle\,|0\rangle$ to $|01\rangle\,|0\rangle$. The basic quantum circuit C for this unitary involves only the q_t and q_a qubits:

$$C = I \otimes X \cdot \text{CNOT} \cdot I \otimes X, \tag{4}$$

where q_t is the target and the auxiliary q_a is the control.

4.3 Playing all cards simultaneously

At each player's turn we want to make use of the quantum capabilities and have them play all their cards at once, i.e. we want to prepare a superposition

$$|\Phi\rangle = \frac{1}{\sqrt{N}} \left(|\Phi_{i_1}\rangle + |\Phi_{i_2}\rangle + \ldots + |\Phi_{i_N}\rangle\right), \tag{5}$$

where each $|\Phi_{i_j}\rangle$ corresponds to the situation where the j-th card that according to the rules of the games can be played, has been played. Our first approach to implement this would be to construct for each N-card subset $\mathcal{S}$ of the 32-card deck a unitary operator $U_{\mathcal{S}}$ that is controlled by the N auxiliary qubits q_a and targets the q_t qubits of each card (using the notation from the paragraph above). While this may be feasible if there are only a few cards left to play, it certainly is not when there are more cards to play. Since these are serious difficulties regarding the implementation of the full game, we will consider a more modest settings, which nevertheless already exhibit some of the characteristics of the 32-card case.

Construction of the controlled CP_k^n gate. We introduce the *controlled card-play* gate, which facilitates playing one of k cards from the hand out of a deck of n cards. Its uncontrolled version is a k-qubit gate which maps $|0\ldots0\rangle$ to the equally weighted superposition

$$\frac{1}{\sqrt{k}} \left(|10\ldots0\rangle + |01\ldots0\rangle + |0\ldots01\rangle\right). \tag{6}$$

In the following, we construct the gate for the action of fore-hand 'player A', whose ID we assume to be encoded in the player-qubit state $|\text{player}\rangle = |00\rangle$. This is only to simplify notation as versions of this gate for other players are constructed completely analogously. Note, that this construction does not yet take suit-following into account and we will therefore not rely on the corresponding qubit from Table 1. For $k = 2$, the unitary gate which facilitates Equation 6 can be expressed via the well-known Hadamard gate: $\text{CNOT} \cdot H \otimes I$. For $k > 2$ the gate can be constructed using the algorithm proposed in [20] via the Boolean function

$$f : \{0,1\}^k \to \{0,1\}\ s.t. f(\vec{x}) = 1, \text{if } x_i = 1 \text{ for exactly one } i. \quad (7)$$

We will denote this gate by $SP_{i_1,\ldots,i_k}$. Note that this gate is supposed to act on the table-qubits $|q_t^{(i)}\rangle$ of the k involved cards. We extend $SP_{i_1,\ldots,i_k}$ to a unitary operator on $\mathcal{H}$ in the usual way. For each card index i let X_i be the Pauli-X operator, which flips the q_t qubits of card i. In the next step we fix a k-tuple $i_1 < \ldots < i_k \leq n$, abbreviated by $\vec{i}$, and put conditions on the operator

$$C_{\vec{i}} := X_{i_1} \cdots X_{i_k} \cdot SP_{i_1,\ldots,i_k} \quad (8)$$

which acts on the $|q_t\rangle$ qubits of each of the k card states.

To make sure, that we only apply the operator $C_{\vec{i}}$ to cards allowed to play, we use the tuple $q_{\vec{p}} = (q_0^{(1)}, q_1^{(1)}; \ldots; q_0^{(n)}, q_1^{(n)})$ of player-qubits and the tuple $q_{\vec{a}}$ of auxiliary qubits to control this gate: we condition the operator in Equation 8 on those tuples where exactly k of the n kets $|q_0^{(l)} q_1^{(l)}\rangle |q_a^{(l)}\rangle$ are in state $|00\rangle |0\rangle$, corresponding to k cards in the hand of player A. Denoting the set of all ordered k-element subsets of the index set by C_k, we can write the resulting unitary as

$$CP_k^n = \prod_{\vec{i} \in C_k} C_{\vec{i}} \big| (\vec{\alpha} = \vec{i}) \quad (9)$$

where $\vec{\alpha}$ denotes the vector of those card indices for which $|q_p q_a\rangle$ is in state $|000\rangle$ and the $|$ symbol indicates the control-condition:

$$C_{\vec{i}} \big| (\vec{\alpha} = \vec{i}) = \begin{cases} C_{\vec{i}}, & \text{if } \vec{\alpha} = \vec{i} \\ I, & \text{else.} \end{cases} \quad (10)$$

Note that $\vec{\alpha}$ is defined for basis states in the computational basis and Equation 10 extends by linearity.

If we want to emphasize that this operator encodes player A playing her cards, we will use the notation $CP_k^n = U_k^A$.

Uncomputing. After the cards have been played, their auxiliaries have to be updated as well. The easiest approach is to apply a Pauli-X gate on each q_a, conditioned on $|q_t q_s\rangle$ being in state $|10\rangle$.

4.4 Trick taking

After the cards are played, the *trick-taking* partial order will determine which player takes the trick. Whenever the subset of played cards has a unique maximal element, the corresponding player takes the trick. Only in the case when there is more than one maximal element, the order in which the cards were played takes precedence.

Construction of the TT_k gate. In the following, we will construct a quantum gate, that simulates the trick taking in the case of a *totally ordered* k-element subset of cards in the trick. We introduce a gate, which implements moving cards from the table onto the round-winner's stack. Again, we will later facilitate the controlled version of this gate. Here we make use of the fact that we can arrange the card order in our encoding in such a way that for a totally ordered subset, a higher card is always left of the lower cards, i.e. the leftmost cards will win the trick.

The uncontrolled version of the TT_k gate is a $(2+1)k$-qubit gate which acts on those parts of the card state that are associated with the location of the card, i.e. $|q_s q_t\rangle$. Trick taking consists of two distinct steps: First, we have to transfer the cards from the table to the stack, i.e., evolve the card state $|q_s q_t\rangle = |01\rangle$ to the card state $|q_s q_t\rangle = |11\rangle$ for each of the k cards. For a single card this is achieved simply by a Pauli-X gate on the stack qubit q_s.

Secondly, we have to identify the winner of the trick and assure that all player-qubits of the associated cards are set to the state of the winner's player-qubit. Suppose that the trick consists of cards with indices $i_1 < i_2 < \ldots < i_k$. Due to the total order in trick taking, all player qubits $|q_0^{(i_l)} q_1^{(i_l)}\rangle$ for $l \geq 2$ have to be set to the same state as $|q_0^{(i_1)} q_1^{(i_1)}\rangle$. We work around the No-Cloning Theorem by using knowledge of the particular states of the player qubits:

- $k = 2$ In the case of two players, only one qubit $|q_p\rangle$ is necessary in the encoding of the players. It suffices to apply a Pauli-X gate to $|q_p^{(i_2)}\rangle$ because we know that $|q_p^{(i_1)}\rangle$ and $|q_p^{(i_2)}\rangle$ are initially in orthogonal states.
- $k > 2$ In the case of three (or more) players we necessarily need auxiliary qubits, since $k-1$ orthogonal states (corresponding to the losing players) are all mapped to the same state (the player-state of the winner). The 3-qubit CC-NOT$(q_{c_1}, q_{c_2}; q_t)$ gate conditioned on $|q_{c_1} q_{c_2}\rangle$ being either in state $|10\rangle$ or $|10\rangle$ maps states of the form $|q_1\rangle |q_2\rangle |q_1\rangle$ to $|q_1\rangle |q_2\rangle |q_2\rangle$ and thus may be used to rewrite player qubits of a lower valued card. In this case, for each player-qubit we need an auxiliary qubit, that up to this part of the circuit is identical to the player-qubit.

For $i_1 < i_2$, denote by D_{i_1,i_2} the unitary gate acting on the player qubits of the $i_1 \otimes i_2$ card states and transforming the vectors as

$$|\text{player}\rangle_{i_1} |\text{player}\rangle_{i_2} \to |\text{player}\rangle_{i_1} |\text{player}\rangle_{i_1}.$$

Similarly, to the CP_k^n-gate in Equation 9, we have to control all the possible unitaries D_{i_1,i_2} for a given k-card configuration and then take the product over all these configurations, i.e. we condition on states where exactly k of the kets $|q_s^{(l)} q_t^{(l)}\rangle$ are in state $|01\rangle$. To do so, we need one auxiliary for each stack qubit q_s. For $\vec{i} = (i_1, \ldots, i_k) \in C_k$ let

$$D_{\vec{i}} = \prod_{r=2}^{k} D_{i_1,i_r}(q_{\vec{p}_{i_1}}, q_{\vec{a}_{i_r}}; q_{\vec{p}_{i_r}}) \cdot \prod_{l=1}^{k} X(q_s^{(i_l)}) \quad (11)$$

where the first factor acts on the trick-losing players qubits and the right factor on all players stack qubits. Finally we define

$$TT_k = \prod_{\vec{i} \in C_k} D_{\vec{i}} \big| (\vec{\alpha} = \vec{i}). \quad (12)$$

where, analogously to the definition of $\text{CP}_\text{k}^\text{n}$, $\vec{\alpha}$ denotes those card indices, for which $|q_s q_t\rangle$ is in in state $|01\rangle$. We will describe the implementation for TT_2 in detail in a later section. When we want

to emphasize the unitary nature of Equation 12 and k is fixed, we will write U_{TT} instead of TT_k.

4.5 Full and partial game

A game of Skat with a 32-card deck and three players consists of ten rounds of trick-taking. We model each round by the unitary round operator

$$R_i = TT_3 \cdot U^C_{11-i} \cdot U^B_{11-i} \cdot U^A_{11-i}, \tag{13}$$

where for the sake of simplicity we have omitted the updating of the auxiliaries. The game evolves by consecutively applying the round operators $R_{10} \cdot \ldots \cdot R_1$. Adaptations for less cards, players and rounds are obvious.

Often, one is given additional information about a particular game, e.g. we could know the starting hand of player A. In this case, one has to prepare a different initial state $|\Phi_{\text{ini}}\rangle = U_{\text{ini}} |0 \ldots 0\rangle$. Slightly abusing notation, let us denote by C_{i_j} the operator that plays the j^{th} card from player A's hand. In this fashion we come up with a different final state for each j:

$$|\Phi^j_{\text{fin}}\rangle = R_{10} \cdot \ldots \cdot R_2 \cdot \tilde{R}^j_1 \tag{14}$$

where, ceteris paribus, $\tilde{R}^j_1 = TT_3 \cdot U^C_{10} \cdot U^B_{10} \cdot C_{i_j}$ models the game, in which player A plays the j^{th} card from her hand. Again, adaptations, for knowledge in different rounds etc. should be easy to implement.

4.6 Evaluation Measurement

The following table shows the value function $\mathcal{V}$ for each of the cards. Note that the suit does not affect the value.

Table 2: Value of the cards. There are 30 points available for each suit what leads to a grand total of 120 points in the game. The declarer needs more than half of it (61) in order to win.

Card	7	8	9	10	J	Q	K	A
$\mathcal{V}$	0	0	0	10	2	3	4	11

For the sake of demonstration, we treat every game as a *spades* game. We can map other trump-suit choices by the declarer to this problem by introducing simple swap gates. This might increase the circuit depth by a small margin, but will not affect the principle feasibility of the approach.

Score Operator. To check who has finally won the game, we define an observable $\mathcal{S}_A$ (the score operator) which returns the point values of the cards in player A's stack. For each card, let us denote by P^i_A the projector onto the states where the player-qubit(s) of the ith card are in a state that decodes to the fact, that this card is the stack of player A. Then

$$\mathcal{S}_A = \sum_{i=1}^{32} \mathcal{V}(i) \cdot P^{(i)}_A, \tag{15}$$

where v_i is the value of card i. The expectation value $\langle \Phi_{\text{fin}} | \mathcal{S}_A | \Phi_{\text{fin}} \rangle$ corresponds to the expectation value of player A's score.

Slightly different and probably more practical would be to ask not for the final score's expectation value, but for the percentage

Table 3: Basic value of the suits, wrt. choice of the declarer.

Suit	♢	♡	♠	♣	G
$\mathcal{V}$	9	10	11	12	24

p_{win} of favorable outcomes. This amounts to describing the subspace of states that are considered "favorable" in a feasible way. Once the projection P_{fav} onto this subspace is available, the overlap $\langle \Phi_{\text{fin}} | P_{\text{fav}} | \Phi_{\text{fin}} \rangle$ equals the probability of a favorable outcome, that is, the probability of player A winning the game.

Quantum Counting. The quantity $\langle \Phi_{\text{fin}} | P_{\text{fav}} | \Phi_{\text{fin}} \rangle$ can be obtained by the following observation: If

$$|\Phi_{\text{fin}}\rangle = \sum_{i \in I_{\text{fav}}} c_i |\phi_i\rangle + \sum_{j \in J} c_j |\phi_j\rangle$$

is the orthogonal decomposition of $|\Phi_{\text{fin}}\rangle$ according to $\mathcal{H} = \mathcal{H}_{\text{fav}} \oplus \mathcal{H}^{\perp}_{\text{fav}}$, the expectation value can be obtained by looking up how many indices $i \in I_{\text{fav}}$ lead to non-vanishing amplitudes c_i.

This is the setup for quantum counting, a combination of Grover's algorithm and quantum phase estimation (QPE), cf. [21, pp. 261-263]., which can be used to gain quantum advantage over the classical algorithms. The number of paths N inside the subset of winning paths can then be counted viacc$N = N^{\text{tot}} \cdot \sin^2 \frac{\phi}{2}$, where ϕ is the result of the QPE.

Quantum counting is used to sum up the winning paths and therefore get a reasonable value for the winning probability. In order to receive a reasonable recommendation for a player to act in the game, it would be necessary to fix the first card and evaluate the score of the game to get a proper winning probability and repeat the process for each card that is allowed according to the rules, in order to compare the card qualities and maximize your expectation value of outcome. Since this value serves as a proper target function to define the *payoff function*, that is yet to be maximized, we dive a bit deeper into its definition accordingly.

Payoff Function. In order to understand the outcome of this calculation and how it defines the game-theoretical solution of the game, we have to introduce a proper payoff function that is maximized by the approach. The payoff function $\mathcal{P}$ of player x after n games is

$$\mathcal{P}_n(x) = \sum_{i=1}^{n} \Big(p_{\text{win}}(x,i) \cdot \text{Val}(x,i)^{\text{won}} - 2 \cdot [1 - p_{\text{win}}(x,i)] \cdot \text{Val}(x,i)^{\text{lost}} \Big). \tag{16}$$

Depending on the choice of the declarer, each suit chosen as the trump by the player x leads to a different value $\text{Val}(x,i)$ of the ith game. It is given by the length of the sustained number of present or missing trumps beginning with the highest, added by one, and multiplied by a certain multiplier $\mathcal{V}(i)$ that is defined purely by the choice of the declarer for the ith game (see Table 3).

It is also important to mention that the value for a *lost* game is *negative* and *doubled*. So in order to outperform your opponents and maximize your expectation value you need a lot more than 50% chance for winning a single game. Those percentages shift drastically as you use the so-called *Seeger-Fabian-System*, which is widely used in all (semi-)professional tournaments. It adds another 50 points for winning and -50 points for losing a game. It usually

Table 4: Encoding the card state $|j_0 j_1 j_2 j_3\rangle$ in the four-card toy-example with $n_c = 3 + 1$.

pos.	encoded state
j_0	player A/player B
$j_1 j_2$	card location
j_3	auxiliary

$j_1 j_2$	loc.
00	hand
10	table
11	stack

does not play a role whether you win a game by a high margin or not. It is only important to *win* it at all. Therefore our goal is to maximize the percentage of games won by player x by finding the best order to play his cards. This order is the one that beats the highest number of possible distributions, no matter what margin.

To check if we found an equilibrium state of the game, the payoff function had to be investigated in terms of stability. There should be at least one *allowed* permutation of cards being played that leads to a situation where every player is satisfied with all of their respective choices according to the information given at the point of the game.

5 FOUR-CARD TOY-EXAMPLE

In this section we explicitly construct the quantum circuit corresponding to Example 1, a card game with four cards and two players and the deck ♣A, ♣10, ♣K, ♣Q.

While the particular selection of cards does not matter much in this setting, it is important to notice the selected cards are a *totally* ordered subset with respect to the trick-taking relation. This first example will already be helpful for representing the superposition of all possible distributions, playing a card, taking the trick, and counting the points at the end of the game.

Encoding. We choose the card-state encoding of Table 4. We arrange the cards according to the trick-taking order, the corresponding values for the scoring are $\mathcal{V}(1) = 11, \mathcal{V}(2) = 10, \mathcal{V}(3) = 4$ and $\mathcal{V}(4) = 3$. The state vector is an element of a $2^{4\cdot4}$-dimensional Hilbert-space and takes the form of the ket

$$|\Phi\rangle = |\phi_{\text{card }1}\rangle\,|\phi_{\text{card }2}\rangle\,|\phi_{\text{card }3}\rangle\,|\phi_{\text{card }4}\rangle\,. \tag{17}$$

and swap the 4 auxiliary bits to the end of the state vector

$$\begin{aligned}|\phi\rangle &= |\overbrace{q_0 q_1 q_2}^{\text{card 1}} \ldots\ldots\ldots q_9 q_{10} q_{11} \overbrace{q_{13} q_{14} q_{15} q_{16}}^{\text{auxiliaries}}\rangle \\ &= |\text{card }1\rangle\,|\text{card }2\rangle\,|\text{card }3\rangle\,|\text{card }4\rangle\,|\text{auxiliaries}\rangle\,.\end{aligned} \tag{18}$$

The qubit q_{12} is used as an external auxiliary for the implementation of the $SP_{1,2}$ gate and thus omitted. Let us review an example: The state for a card on the table is $|010\rangle$ (played by player A) or $|110\rangle$ (played by player B). The four card states are encoded using a total twelve qubits. As an example, the qubit state (omitting the auxiliaries) $|000000100100\rangle$ decodes to card 1 and 2 being on the hand of player A while card 3 and 4 are on the hand of player B.

Evolution of the game. The game consists of different phases, summarized in Table 5. In each phase the state vector evolves by applying the corresponding unitary operator, finally ending up with

$$|\Phi_{\text{fin}}\rangle = \prod_{i=1}^{2} \left(U_{\text{TT}}\, U_i^B\, U_i^A\right)\, U_{\text{ini}}\, |0\ldots0\rangle\,. \tag{19}$$

Measurement of the score-operator $\mathcal{S}_A$ equals the expectation of player A's final score $\langle\Phi_{\text{fin}}|\,\mathcal{S}_A\,|\Phi_{\text{fin}}\rangle$ and Player A's probability

Table 5: Game phases and corresp. unitary transformations.

phase	content	unitary
Round 0	initial superposition	U_{ini}
Round 1	Player A plays a card	U_2^A
	Player B plays a card	U_2^B
	1st trick taking	TT_2
Round 2	Player A plays a card	U_1^A
	Player B plays a card	U_1^B
	2nd trick taking	TT_2
Round 3	Measurement	-

of winning is given by $\langle\Phi_{\text{fin}}|\,P_{\text{fav}}\,|\Phi_{\text{fin}}\rangle$. In this example, $P_{\text{fav}}(\mathcal{H})$ turns out to be three-dimensional. We will now review each factor of Equation 19 in detail.

U_{ini}: Construction of the initial superposition. First, we have to construct all initial distributions. Here, one player gets two out of the four-card deck, and his opponent receives the remaining two cards. There are $\binom{4}{2} = 6$ ways to do this. We construct the superposition of these six basis states iteratively according to the algorithm proposed in [20]:

$$|\Phi^{(0)}\rangle = \frac{1}{\sqrt{6}} \sum_{i=1}^{6} |\phi_i\rangle = U_{\text{ini}}\,|0\ldots0\rangle\,. \tag{20}$$

This results in a superposition with equally weighted probabilities of $\frac{1}{6} \approx 0.167$ as shown in Figure 2.

$U_2^B U_2^A$: Playing the first card. In the first round 1, player A has two cards on his hand, the card index set is $\{1, 2, 3, 4\}$, so the index set in Equation 6 is $C_2 = \{(i, j) : 1 \le i < j \le 4\}$. With the chosen encoding, playing a card transforms the state $|q_0 00\rangle \to |q_0 10\rangle$, which amounts to flipping the middle qubit, i.e. q_1, q_4, q_7 or q_{10} depending on the card. Each factor CP_2^4 in Equation 9 thus acts on two qubits via $X_l X_m SP_{l,m}$ for $i, j \in \{1, 4, 7, 10\}$. Note, that we have to apply a version of the circuit, that is controlled by the condition that exactly two of the player qubits $\{q_0, q_3, q_6, q_9\}$ are in state $|0\rangle$. We do not have to condition on the table-qubits resp. their auxiliaries, because in the first round they all are in state $|0\rangle$. After applying all factors of the corresponding unitary U_2^A the state is now an equally weighted superposition of twelve basis states with probabilities of $\frac{1}{12} \approx 0.083$ as shown in Figure 3a. The same circuit, with the only change that the condition is now on two player-qubits being in state $|1\rangle$, implements the unitary U_2^B for player B. The result is now an equally weighted superposition of 24 basis states with probabilities of $\frac{1}{24} \approx 0.042$ as shown in Figure 3b.

5.1 Preparing the auxiliaries

In this toy example we only need one auxiliary per card, which keeps track if a card is still in play, i.e. the stack qubits. For simplicity, these auxiliaries are updated only now. We condition on stack qubits q_2, q_5, q_8, q_{11} being in state $|1\rangle$ and apply a series of CNOT-gates: $\prod_{i=1}^{4} \text{CNOT}_{3i-1,12+i}$.

TT_2: First trick taking. The two cards on the table can have card indices $1 \le i < j \le 4$. The operator $D_{i,j} = X_{3(j-1)}$ flips the player qubit of the card with index j. We will apply this operator only on states, where the cards with index i and j belong to different players and hence end up with a state where both cards belong to the

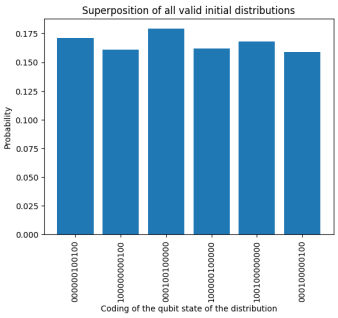


Figure 2: Measurement of $\Phi^{(0)}$ with 1 000 shots.

same player. The unconditioned unitary from Equation 11 is given byv $X_{3j-3} \cdot X_{3j-1} \cdot X_{3i-1}$, e.g. the unitary $D_{1,2} = X_3X_5X_2$ would take the state $|\text{card}\,1\rangle\,|\text{card}\,2\rangle = |010\rangle\,|110\rangle$ to the state $|011\rangle\,|011\rangle$ where both card 1 and 2 end up in the stack of player A. The fully conditioned version iterates the six index configurations $i < j$ in general where the qubits $|q_{3i-2}q_{12+i}\rangle$ and $|q_{3j-2}q_{12+j}\rangle$ are exactly in state $|10\rangle$. The resulting state is again an equal superposition of 24 basis states, albeit different from before the application of U_{TT}, cf. Figure 3c.

$U_1^B U_1^A$*: Playing the second card.* For playing the last remaining card, there is no choice left for the players. Thus the unitaries $U_1^{A/B}$ simply flip the qubits q_1, q_4, q_7 or q_{10} subject to the condition that the corresponding auxiliary is in state $|0\rangle$.

TT_2*: 2nd trick taking.* We apply the same unitary as in the 1st trick taking. All card states are now of the form $|q_p 11\rangle$. The following outcomes are possible:

- all cards on the stack of player A (probability $\frac{1}{4}$)
- all cards on the stack of player B (probability $\frac{1}{4}$)
- two cards on each of the two stacks (probability $\frac{1}{12}$ for each of the six possible combinations)

This results in a superposition of 8 states, as shown in Figure 3d.

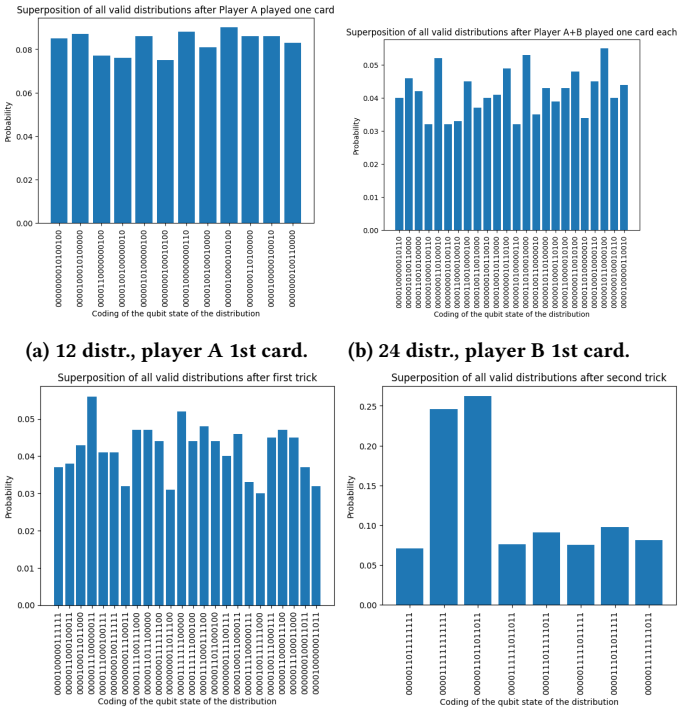
Measurement: Scoring. Finally, we can apply the score operator $\mathcal{S}_A$ from Equation 15, which returns the expected point values of the cards in player A's stack. In this particular example, we can easily also construct the subspace of favorable outcomes: player A wins if either $\{\clubsuit A, \clubsuit 10, \clubsuit K, \clubsuit Q\}$, $\{\clubsuit A, \clubsuit 10\}$ or $\{\clubsuit A, \clubsuit K\}$ are in her stack. Each situation corresponds to a particular basis state. Here the subspace is 3-dimensional and thus $p_{\text{win}} = \frac{5}{12}$.

6 QUANTUM FEASIBILITY OF FULL GAME

The quantum nature of the state leads to a probabilistic distribution. Thus, each state is measured by a certain probability. To reduce statistical error, the quantum circuit is performed several times in the same state. As described above, the three players get ten cards plus two more cards that are accessible during a bidding process. Talking about the bare numbers, we obtain

$$\mathcal{N} = \binom{32}{10}\binom{22}{10}\binom{12}{10} \approx 2.753 \cdot 10^{15}$$

different deals. So, there are roughly 2.75 quadrillion possibilities for the cards to be distributed among the three players (neglecting the



(a) 12 distr., player A 1st card. **(b) 24 distr., player B 1st card.**

(c) 24 distr., 1st trick. **(d) 8 distr., 2nd trick.**

Figure 3: Superposition of possible card distributions after certain time steps, using 1 000 shots.

three possible positions of the issuer). Following the well-known *birthday paradox*, the probability p that at least one deal is repeated in k deals is $p = 1 - (\prod_{i=0}^{k-1}(n-i)/n^k)$; and fixing $p \geq 50\%$ yields $k \geq 40$ million games, and, thus, highly unlikely.

In an empirical analysis, we played hundreds of millions of Skat games with three AIs, with an average CPU time of approx. 3 s per game (on a one-core contemporary PC), which roughly is 1s analysis time per player per game. Even if we reduce this computation time to 0.1 s for less analysis and more computation power, the scaling of the problem will still cause an incalculable tree search. Computing the card proposals for a selected player for all $2.753 \cdot 10^{15}$ deals and dividing this number by $(3600 \times 24 \times 365)$ would result to about 8 730 639 years. Search mechanisms for the players employ, e.g., mini-max search to find the value of an open game, which assumes full card knowledge and both opponent players to collaborate, followed by a voting on different samples [12], search on the partition of the cards to suit factors [7], or a so-called paranoia search, where opposing cards were sampled in the search tree with, e.g., the knowledge on non-obeying is being propagated [9]. These tree search algorithms do not count the exact number of winning branches in the search tree, but propagate the game value bottom-up. Moreover, in search practice is pruning in transposition tables, and via $\alpha\beta$ search or the equivalent of a moving target search. Our estimation suggests that it would take up to 8.7 million years to compute all game paths for all possible distributions, assuming a tenth of a second of computation time per game. Note that these values are very rough estimates of the classical computation time,

as with smaller numbers of cards in the deal the time per game will decrease as well. However, note that in a real AI player implementation, the fewer the cards, the more analysis, e.g., earlier search to find forced wins, or drawing more samples for counterbalancing the uncertainty in the unknown cards are in the voting.

Partial Knowledge. As mentioned above, the basic idea of a game with imperfect information is not to find *the* one and only way to win this particular game, but to maximize the expectation value of your payoff function. In order to be successful, it is necessary to maintain a high level of concentration among long tournaments to make the right decisions more often than your opponents. Interestingly, you can make *correct* decisions that turn out to be totally wrong and misleading in a particular situation. After all, the *solution* to a certain situation inside the game is to find the card that gives you the win (and, therefore, the points) in most of the remaining scenarios. Accordingly, since each player knows ten cards, it is only necessary to have a closer look at the remaining cards. If the Ace of Spades is on your side and as every card is unique, the superposition, and, therefore, the Hilbert Space dimension reduces.

$$|\psi\rangle = |\phi_1\rangle + \underbrace{\cancel{|\phi_2\rangle}}_{\spadesuit\text{A on opponent's hand}} + |\phi_3\rangle + \cdots + |\phi_N\rangle\,. \tag{21}$$

For a given hand on player A, the problem size for the suggestion of a strategy shrinks a lot, since the number of possibilities reduces. For the total number of remaining states $\widetilde{N}$ one gets

$$\widetilde{N} = \binom{22}{10}\binom{12}{10} = 42\,678\,636. \tag{22}$$

Note that solvers for given situations and distributions already exist. But that does not necessarily mean, that this reflects the way you *should* play in order to maximize the expected payoff.

Upper Bound. Let us consider three players playing the game, everyone equipped with their belonging Quantum Computer. If everyone plays 100% rational, they should all tend towards a Nash equilibrium, with consecutively getting better results as the game proceeds, since the information the game provides to you is growing alongside. The state, where everything about the distribution is known, might occur at some point earlier than step ten as well, depending on the present distribution. In that case, the superposition of possible states enters the classical limit and can be solved using the existing devices. In earlier stages, where distributions that differ heavily from the present one are still possible, it might lead to different conclusions, giving you suggestions that are not valuable in that particular case, rather giving you the lead in the long run.

It has to be mentioned, that, with this approach we solely consider cards to be played perfectly by both players. This might be an unrealistic scenario for a real-game approach, since the partner might always misinterpret the card. One has to take into account, that for a real game *not all the possible distributions left are equally likely*, because the opponents do always seek for an attack. Therefore, the probabilities will change and the paths have to be "weighted" according to the cards that appear on the table.

Since the internal rules of the game do not allow for all the cards to be played at all times, it has to be mentioned that this is an upper boundary for the possibilities and the computational resources to be needed. In fact, for an actual game prediction, the computational effort that has to be put in is much less, since you have to follow rules that significantly reduce your options.

The upper bound for this scenario is quite easy to find. If *everyone* was allowed to play *every* single card at *all times*, they would all end up with a total of $N_\mathrm{i} = 10!$ options. That gives a total sum of possible paths in the decision tree for a given distribution of $N_\mathrm{tot} = (10!)^3$. As already seen in Equation 22 the total number of distributions once you see your own cards is 42 678 636. That leads to a grand total of $\widetilde{N}_\mathrm{tot} = 2\,039\,386\,920\,479\,691\,374\,592\,000\,000 \approx 2 \cdot 10^{27}$ possible games to be played with 22 of 32 cards unknown. This number is the upper bound, since the game rules highly restrict the paths that a game can technically take. It can go from an unrestricted choice of card to a forced card you have to play without any other options. Taking all these considerations into account, this huge solution space shrinks drastically, depending on the present distribution.

7 CONCLUSION

We have shown on a conceptual level that we can encode the game information of an imperfect-information game, such as *Skat*, into quantum states, prepare the game's initial state in a superposition using unitary gates. The uncertainty of the random deal is represented by this initial quantum register. We can also model trick-taking, measure scoring and map it to the subspace of winning distributions. Hence, calculations to evaluate the belief space for incomplete-information card game play can be efficiently executed on a quantum computer. Evaluation of states is possible by a model-counting procedure through the comparison of the winning subspace after the measurement of the scoring. We have discussed computational aspects in terms of reducing the number of paths in the quantum computation as well.

It turns out, that, according to rough estimations, there might be a quantum advantage in the tree search as the complexity of the game size increases at around ∼ seven cards in each player's hand. Since the full game itself is played with ten cards per player, we expect quantum advantage certainly to occur up to this point in the future. Generally speaking, the iteration over different initial states and game rules for the various type of games, the exploration results may influence the bidding and Skat putting and game selection strategies. The quantum computation can be invoked as a recommendation at any card to be played in the game.

Our aims with this paper were more general. We chose Skat as a prototype of one selected partially-observable planning-problem to show its applicability. There will be a wider range of other problems where our derivations to exploit quantum computational gains will prove helpful. This work is, therefore, best interpreted as an inspiring, pioneering theoretical and pedagogical accessible workout to discuss quantum feasibility for solving partially observable problems. So far, the exposition comes without the implementation on a real quantum computer, but we are certain that in the visible future, we can expect applications for quantum computing in incomplete-information games.

ACKNOWLEDGMENTS

We appreciate discussions with M. Bauer, M. Knufinke, I Seipp, J. Wehling, S. Seegerer, A. Hirschmeier and W. T. Strunz. Stefan Edelkamp's contribution was supported by GAČR project 24-12046S.

REFERENCES

[1] Jean R. S. Blair, David Mutchler, and Cheng Liu. 1993. Games with Imperfect Information. *AAAI Technical Report* FS-93-02 (1993), 59–67.

[2] Samuel Bounan and Stefan Edelkamp. 2024. Trick Costs for $\alpha\mu$ and New Relatives. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*.

[3] Michael Buro, Jeffrey Richard Long, Timothy Furtak, and Nathan R. Sturtevant. 2006. Improving State Evaluation, Inference, and Search in Trick-Based Card Games. In *Simulation*. 135–147.

[4] Tristan Cazenave and Véronique Ventos. 2019. The $\alpha\mu$ Search Algorithm for the Game of Bridge. *CoRR* abs/1911.07960 (2019). arXiv:1911.07960 http://arxiv.org/abs/1911.07960

[5] Gal Cohensius, Reshef Meir, Nadav Oved, and Roni Stern. 2020. Bidding in Spades. In *ECAI (Frontiers in Artificial Intelligence and Applications, Vol. 325)*, Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang (Eds.). IOS Press, 387–394. https://doi.org/10.3233/FAIA200117

[6] Stefan Edelkamp. 2019. Challenging Human Supremacy in Skat. In *Symposium on Combinatorial Search (SOCS)*. 52–60. https://aaai.org/ocs/index.php/SOCS/SOCS19/paper/view/18328

[7] Stefan Edelkamp. 2020. Dynamic Play via Suit Factorization Search in Skat. In *KI (Lecture Notes in Computer Science)*. Springer, 18–32. https://doi.org/10.1007/978-3-030-58285-2_2

[8] Stefan Edelkamp. 2021. ELO System for Skat and Other Games of Chance. *CoRR* abs/2104.05422 (2021).

[9] Stefan Edelkamp. 2021. Knowledge-Based Paranoia Search. In *2021 IEEE Conference on Games (CoG), Copenhagen, Denmark, August 17-20, 2021*. IEEE, 1–8. https://doi.org/10.1109/COG52621.2021.9619073

[10] Stefan Edelkamp. 2021. On the Power of Refined Skat Selection. *CoRR* abs/2104.02997 (2021).

[11] Stefan Edelkamp. 2022. Improving Computer Play in Skat with Hope Cards. In *Computers and Games - International Conference (CG), Revised Selected Papers (LNCS, Vol. 13865)*. Springer, 133–145. https://doi.org/10.1007/978-3-031-34017-8_12

[12] Stefan Edelkamp. 2024. Could the Declarer Have Discarded It? Refined Anticipation of Cards in Skat. In *KI (Lecture Notes in Computer Science)*, Andreas Hotho and Sebastian Rudolph (Eds.). Springer, 60–72.

[13] Timothy Michael Furtak. 2013. *Symmetries and Search in Trick-Taking Card Games*. Ph.D. Dissertation. University of Alberta.

[14] John C. Harsanyi. 1967/1968. Games with Incomplete Information Played by Bayesian Players, I-III. *Management Science* 14(3), 14(5), 14(7) (1967/1968), 159–183 (Part I), 320–334 (Part II), 486–502 (Part III).

[15] Thomas Keller and Sebastian Kupferschmid. 2008. Automatic Bidding for the Game of Skat. In *KI*. 95–102.

[16] Sebastian Kupferschmid. 2006. *Entwicklung eines Double-Dummy Skat Solvers mit einer Anwendung für verdeckte Skatspiele*. Master's thesis. University of Freiburg.

[17] C. E. Lemke and J. T. Howson, Jr. 1964. Equilibrium Points of Bimatrix Games. *J. Soc. Indust. Appl. Math.* 12, 2 (1964), 413–423. https://doi.org/10.1137/0112033 arXiv:https://doi.org/10.1137/0112033

[18] Junkang Li, Bruno Zanuttini, Tristan Cazenave, and Véronique Ventos. 2022. Generalisation of Alpha-Beta Search for AND-OR Graphs With Partially Ordered Values. In *IJCAI*. 4769–4775. https://doi.org/10.24963/IJCAI.2022/661

[19] Jeffrey Richard Long. 2011. *Search, Inference and Opponent Modelling in an Expert-Caliber Skat Player*. Ph.D. Dissertation. University of Alberta.

[20] Fereshte Mozafari, Heinz Riener, Mathias Soeken, and Giovanni De Micheli. 2021. Efficient Boolean Methods for Preparing Uniform Quantum States. *IEEE Transactions on Quantum Engineering* 2 (2021), 1–12. https://doi.org/10.1109/TQE.2021.3101663

[21] Michael A. Nielsen and Isaac L. Chuang. 2010. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press.

[22] Douglas Rebstock, Christopher Solinas, and Michael Buro. 2019. Learning Policies from Human Data for Skat. *CoRR* abs/1905.10907 (2019). arXiv:1905.10907 http://arxiv.org/abs/1905.10907

[23] Douglas Rebstock, Christopher Solinas, Michael Buro, and Nathan R. Sturtevant. 2019. Policy Based Inference in Trick-Taking Card Games. *CoRR* abs/1905.10911 (2019). arXiv:1905.10911 http://arxiv.org/abs/1905.10911

[24] F. Schettler and G. Kirschbach. 1988. *Das große Skatvergnügen*. Urania Verlag, Leipzig, Jena, Berlin.

[25] Peter W. Shor. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* 26, 5 (Oct. 1997), 1484–1509. https://doi.org/10.1137/s0097539795293172

[26] Christopher Solinas, Douglas Rebstock, and Michael Buro. 2019. Improving Search with Supervised Learning in Trick-Based Card Games. *CoRR* abs/1903.09604 (2019). arXiv:1903.09604 http://arxiv.org/abs/1903.09604

[27] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. 2007. Regret Minimization in Games with Incomplete Information. In *Advances in Neural Information Processing Systems*, J. Platt, D. Koller, Y. Singer, and S. Roweis (Eds.), Vol. 20. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2007/file/08d98638c6fcd194a4b1e6992063e944-Paper.pdf