The Importance of Rank in Trick-Taking Card Games

Édouard Bonnet[1] and Abdallah Saffidine[2]

1 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
2 **CSE, The University of New South Wales, Sydney, Australia**

Abstract

Recent years have seen a surge of interest in computational complexity results in card games. We focus in this paper on trick-taking card games, extending a study started in 2013. This class ranges from WHIST and CONTRACT BRIDGE to SKAT and TAROT. We investigate how the number of cards per suit, the ranks, impacts the complexity of solving arbitrary positions. We prove that 2 distinct ranks are sufficient to induce PSPACE-hardness for the question of determining if a team can make all tricks. This strengthens a 2013 result that assumed at least 5 cards per suit and no constraint on the number of tricks. Indeed, our analysis indicates that the concept of card discarding is expressive enough to encode universal quantification. Conversely, tractability ensues as soon as a second dimension is bounded. Our results provide a complete picture of the computational influence of the rank parameter.

1998 ACM Subject Classification F.2.2

Keywords and phrases trick-taking card games, computational complexity

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

We can partition the ongoing effort of the Artificial Intelligence community to address games in two different methodologies. The first one, experimental in nature, aims at developing algorithms and techniques to play and solve concrete games. Some algorithms such as CounterFactual Regret minimization (CFR) aim at finding Nash equilibria in imperfect information games and have been crucial in solving a two-player variant of Poker [5]. Other card-game-related approaches trade convergence guarantees for scalability and lead to developments such as Perfect Information Monte Carlo sampling, motivated by trick-taking card games such as CONTRACT BRIDGE [10] and SKAT [6], and Information Set Monte Carlo Tree Search which was applied to the Chinese card game Dou Di Zhu [7].

The second methodology has ties in theoretical computer science and considers the computational complexity of generalizations of strategy games. In terms of card games, the first formal complexity result we can trace is due to [9]. Recently, a surge of interest in the topic has lead to results in games ranging from UNO [8] to HANABI [1] and the Russian DURAK [2]. In this paper, we continue the investigation of the complexity of the class of *trick-taking card games* started by [4]. This class encompasses numerous popular games besides BRIDGE and SKAT: HEARTS, TAROT, and WHIST are also examples of trick-taking card games.[1]

Note that contrary to the best defense model assumed by [9]'s NP-completeness result, this paper is inscribed in a stream of research assuming perfect information and a compact input. This is in

[1] A detailed description of these games and many other can be found on `http://www.pagat.com/class/trick.html`.

line with the rest of the research on the complexity of card games [17, 18, 8, 4, 1, 2]. There are several reasons for focusing on perfect information. First, it provides a lower bound to the imperfect information case when compact input is assumed. More importantly, perfect information trick-taking card games actually do appear in practice, both among the general population in the form of DOUBLE DUMMY BRIDGE problems, but also in research as perfect information Monte Carlo sampling is used as a base component of virtually every state-of-the-art trick-taking game engine [11, 10, 16, 13].

The rules of the quintessential trick-taking card game are fairly simple. A set of players is partitioned into teams and arranged around a table. Each player is dealt a given number of cards t called *hand*, each card being identified by a *suit* and a *rank*. The game consists in t tricks in which every player plays a card. The first player to play in a given trick is called lead, and the other players proceed in the order defined by the seating. The single constraint is that players should follow the lead suit if possible. At the end of a trick, whoever put the highest ranked card in the lead suit wins the trick and leads the next trick. When there are no cards remaining, after t tricks, we count the number of tricks each team won to determine the winner.[2]

It is natural to define fragments of this class of decision problems, for instance, by limiting the number of hands, the number of different suits, or even limiting the number of cards within each suit. We recall the lattice of such fragments in Section 2. The decision problem in the foundation paper was to determine whether team A can ensure winning at least k tricks against any defense of team B, for some input integer k [4]. A relatively easy reduction from Generalized Geography (GG) showed that, in full generality, this problem is PSPACE-complete [4, Theorem 2].

In this paper, we considerably improve this result by proving PSPACE-hardness on a significantly restricted fragment. As a warming-up, we first show that two cards per suit are sufficient to induce hardness. Due to space limitations, we defer this proof and the definition of GG, respectively, to Section B and Section A of the appendix.

We then focus on the special case of deciding whether there exists a strategy ensuring all tricks are collected, *i.e.* requiring $k = t$. There are two convergent motivations behind the analysis of this fragment. The first motivation is pragmatical: in several trick-taking card games, there is a special status to making all the tricks (the so-called *grand slam* in Bridge, or the *Schwarz* Skat contract). Could it be that there is an efficient algorithm deciding if team A can do all the tricks? After some thought, it appears that this variant is NP-hard. This leads to the second and more philosophical motivation. Whenever team B has the lead, the game is over (team B has made one trick). Under these circumstances, *can team B make some meaningful decisions?* Is the grand slam *a 1 or a 2-player game (*NP*-complete vs* PSPACE*-complete)?* In Section 3, we answer this question: the problem remains PSPACE-hard, even if we require team A to make all tricks *and* the number of ranks is 2. Let us observe that with more than 2 ranks, the *decisions* of team B could have come from the choice of the rank played in the lead suit. With only two ranks, such a choice is impossible and the only influence of team B will come from the discards. By an intricate construction relying on a card mechanism called *squeeze*, we show that choosing what to discard is sufficiently expressive to encode universal quantification.

We complement these hardness proofs by tractability results in Section C of the appendix. Those positive and negative results yield a complete picture and a full understanding of the algorithmic influence of the number of ranks. Finally, we provide a graphical summary of the complexity landscape in trick-taking card games (Section E of the appendix) before putting forward a few open problems (Section 4).

[2] There are more elaborate point-based variants where tricks might have different values, possibly negative, based on cards comprising them. We focus on the special case where each card has the same positive value.

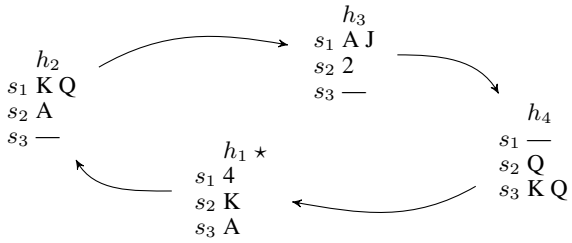

Figure 1 Example of a trick-taking game position with 4 hands, 3 suits, and 1 as lead turn. If team A controls h_1 and h_3 and team B controls h_2 and h_4, then team A can make all three remaining tricks by starting with (s_3, A)

2 Definitions and notation

2.1 Trick-taking game

▶ **Definition 1.** A *card* c is a pair of two integers representing a *suit* (or color) $s(c)$ and a *rank* $r(c)$. A *position* p is defined by a tuple of hands $h = (h_1, \ldots, h_\eta)$, where a *hand* is a set of cards, and a *lead turn* $\tau \in [1, \eta]$. We further assume that all hands in a given position have the same size $\forall i, j \in [1, \eta], |h_i| = |h_j|$ and do not overlap: $i \neq j \Rightarrow h_i \cap h_j = \emptyset$.

An example position with 4 hands and 12 total cards is given in Figure 1. The position is written as a diagram, so for instance, hand h_3 contains 3 cards $\{(s_1,\mathrm{A}), (s_1,\mathrm{J}), (s_2, 2)\}$.

▶ **Definition 2.** Playing a *trick* consists in selecting one card from each hand starting from the lead: $c_\tau \in h_\tau, c_{\tau+1} \in h_{\tau+1}, \ldots, c_n \in h_\eta, c_1 \in h_1, \ldots, c_{\tau-1} \in h_{\tau-1}$. We also require that *suits are followed*, i.e., each played card has the same suit as the first card played by hand τ or the corresponding hand h_i does not have any card in this suit: $s(c_i) = s(c_\tau) \vee \forall c \in h_i, s(c) \neq s(c_\tau)$.

▶ **Definition 3.** The *winner of a trick* is the index corresponding to the card with highest rank among those having the required suit. The position resulting from a trick with cards $C = \{c_\tau, \ldots, c_{\tau-1}\}$ played in a position p can be obtained by removing the selected cards from the hands and setting the new lead to the winner of the trick.

In the example in Figure 1, the lead is to 1. A possible trick would be $(s_3,\mathrm{A}), (s_1,\mathrm{Q}), (s_2, 2), (s_3,\mathrm{Q})$; note that only hand h_4 can follow suit, and that 1 is the winner so remains lead.

▶ **Definition 4.** A *team mapping* σ is a map from $[1 \ldots \eta]$ to $\{A, B\}$ where η is the number of hands. A (perfect information, plain) *trick-taking game* is pair consisting of a position and a team mapping σ.

For simplicity of notation, team mappings will be written as words over the alphabet $\{A, B\}$. For instance, $1 \mapsto A, 2 \mapsto B, 3 \mapsto A, 4 \mapsto B$ is written $ABAB$.

▶ **Definition 5.** A trick is won by team A if its winner is mapped to A with σ. The *value* of a game is the maximum number of tricks that team A can win against team B.

The value of the game presented in Figure 1 is 3 as team A can ensure making all remaining tricks with the following strategy known as *squeeze*. Start with (s_3, A) from h_1 and play $(s_2, 2)$ from h_3, then start the second trick in the suit where h_2 elected to play.

2.2 Decision problem and fragments

The most natural decision problem associated to trick-taking games is to compute whether the value of a game is larger or equal to a given value ν. Put another way, is it possible for some team to ensure capturing more than ν tricks? We will see that the general problem is PSPACE-hard, but there are several dimensions along which one can constrain the problem. This should allow to better understand where the complexity comes from.

Team mappings only allow team mappings belonging to a language $\mathcal{L} \subseteq \{A, B\}^*$.
Number of suits the total number of distinct suits s is bounded by a number $s = S$, or unbounded $s = \_$.
Number of ranks the total number of distinct ranks over all suits r is bounded by a number $r = R$, or unbounded $r = \_$.
Symmetry for each suit, each hand needs to have the same number of cards pertaining to that suit.

The fragments of problems respecting such constraints are denoted by $\mathcal{B}(\mathcal{L}, s, r)$ when symmetry is not assumed. If symmetry is assumed, then we denote the class by $\mathcal{B}^{\mathcal{M}}(\mathcal{L}, s, r)$, whereas we denote by $\mathcal{B}_{\mathcal{A}}(\mathcal{L}, s, r)$ the restriction in which team A has to win all the tricks. The largest class, that is, the set of all problems without any restriction is $\mathcal{B}(\_, \_, \_)$.

The class of double-dummy Bridge problems is exactly $\mathcal{B}(\{ABAB\}, 4, 13)$.

Except for the double-dummy team-mapping $ABAB$, we will not treat separately team-mappings containing the same number of hands in different orders. A simpler notation is thus available: we write $\mathcal{B}(\eta, s, r)$ as short for $\mathcal{B}(\{A, B\}^\eta, s, r)$, *i.e.*, the class of problems with η hands and s suits containing r cards each.

▶ Proposition 1. $\mathcal{B}(\_, \_, \_)$ is in PSPACE.

Proof. The game ends after a polynomial number of moves. It is possible to perform a minimax search of all possible move sequences using polynomial space to determine the maximal number of tricks team A can achieve. ◀

We recall our contributions, this time with the formal fragment notations.

1. Theorem 9 shows that $\mathcal{B}(\_, \_, 2)$ is PSPACE-hard.
2. Theorem 7 goes a step further and establishes that $\mathcal{B}_{\mathcal{A}}(\_, \_, 2)$ is PSPACE-hard.
3. Theorem 10 shows that for any constant N and R, the fragment $\mathcal{B}(N, \_, R)$ is tractable.
4. $\mathcal{B}(\_, \_, 1)$ is tractable (Prop. 2) and so is $\mathcal{B}(\_, S, R)$ for any constant S and R (Prop. 4).

We now have a complete understanding of the card rank parameter.

3 $\mathcal{B}_{\mathcal{A}}(\_, \_, 2)$ is PSPACE-complete

We now prove that $\mathcal{B}_{\mathcal{A}}(\_, \_, 2)$ is PSPACE-complete. An approach similar to the one in Theorem 9, and [4, Theorem 2], that is, a reduction from a variant of GG via a mapping from vertices to hands appears to be difficult. Indeed, the *all tricks* constraint introduces an important difference: while in Theorem 9, each player's GG decisions were mapped to decisions on how to lead the next card, this symmetry is not possible in $\mathcal{B}_{\mathcal{A}}(\_, \_, 2)$ because team B does not get to lead any card, except possibly the very first one, until the instance is solved.

Instead, we reduce from a more fundamental tool, the True (or Totally) Quantified Boolean Formula (TQBF) problem. We still need to find a way to provide team B with a means to influence which team A hand wins the next trick, through team B's reply to a trick lead by team A. A *Bridge Squeeze* [15], as in Figure 1 can allow the defending team to choose in which suit a trick is lost.

We draw inspiration from this elaborate domain-specific technique and adapt this insight to the case with more than 4 hands but only two ranks. This results in a gadget where the defending team can suggest which attacking hand leads next by offering a bonus trick. Unfortunately, this is not a strong constraint on the attacking team as it can miss out on the bonus trick and follow its own preference for the next lead. A large part of the difficulty in proving that $\mathcal{B}_{\mathcal{A}}(\_,\_,2)$ is PSPACE-complete is forcing the attacking team to abide by the defending team's suggestions.

3.1 TQBF

The input of TQBF is a fully quantified Boolean formula in prenex normal form with the propositional part in conjunctive normal form with at most literals by clause (CNF). That is, the input can be written as $\exists x_1 \forall x_2 \exists x_3 \ldots \forall x_n \phi(x_1, \ldots, x_n)$ where ϕ is a conjunction of m clauses: $\phi = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{m_i} l_i^j$ and $l_i^j \in \{x_k, 1 \leq k \leq n\} \cup \{\neg x_k, 1 \leq k \leq n\}$. For example,

$$\exists v_1 \forall v_2 \exists v_3 \forall v_4 (v_1 \vee v_2 \vee v_4) \wedge (\neg v_1 \vee \neg v_2 \vee \neg v_3) \wedge (\neg v_4 \vee v_2 \vee v_1)$$

is a valid instance of TQBF. The PSPACE-complete decision problem associated to TQBF is determining whether the input formula is true. In our example, the formula is true indeed, which can be seen from setting $v_1 = \top$ and then for any choice of v_2, setting $v_3 = \bot$. Then no matter what value is assigned to v_4, the propositional part is true.

Our reduction from TQBF to $\mathcal{B}_{\mathcal{A}}(\_,\_,2)$ uses 10 types of gadgets.

A Choice This gadget captures the idea of an existential choice (a decision by Team A).

B Choice This dual gadget captures decisions by team B, it embeds two kinds of internal gadgets: **Unit Choices** and **Magnifiers**.

Conjunction This gadget builds on many B Choice gadgets and uses **Waiting** gadgets to synchronize them, each of the latter is parameterized by a delay.

Disjunction A simple generalization of the A Choice.

Quantifier This gadgets embeds one type of choice gadget which defines whether we have an existential or universal quantifier.

Variable This is another simple gadget linking literals selected via the formula gadgets (Conjunction and Disjunction) back to the quantifier of the corresponding variable.

Conclusion A gadget parameterized by a cost, in which the games *following the semantics* of the TQBF formula are supposed to end.

Each gadget features an internal set of hands and sub-gadgets as well as an internal set of suits. Each gadget also presents two types of communicating suits that make it possible to connect it to other gadgets in the reduction: the gadget provides the A of each *entering suit*, and provides the K of each *exiting suit*. To help the reader, we adopt the convention that those *entering* and *exiting* cards are displayed in bold font. The gadgets are presented in a Table format where each horizontal line corresponds to an internal hand or an embedded gadget, and where suits are different across columns and across horizontal lines splitting padding suits. The first column determines the type of sub-gadget or whose team the hand belongs to.

3.2 Choice gadgets

The first and simplest gadget is the *A-choice gadget* (Table 1). It contains two internal hands, has a single entering suit label s, two exiting suits s_+ and s_-, as well as a number of *padding suits* that ensure every hand in the reduction has the same number of cards. The first internal hand is controlled by team A and can make a single trick in the game, by playing the A in suit s. Team A can then choose to lead in s_+ or in s_-, and the gadget is exited. The second internal hand is controlled by team

Table 1 Team A choice gadget: A−`choice`. The gadget is entered by playing in suit s and can either exit through suit s_+ or suit s_-. Team A chooses whether the gadget is exited via s_+ or via s_-. In both cases, a single trick is spent in the gadget.

A	s s_+ s_-	Padding
A	**A K K**	K…K
B		AAAA…A

Table 2 Unit choice gadget for Team B: `unit`. The gadget is entered by playing in suit s and can either exit through suit s_+ or suit s_-. Team B selects which suit the gadget exits from after 4 tricks are played. However, team A can choose to force the exit suit at the cost of only winning 3 tricks in the gadget.

	s	s_+ s_-	Padding
A	**AK**		KK…K
B	A A		AA…A
A	AK K		K…K
B			AAAA…A
A	AK	**K**	K…K
B			AAAA…A
A	AK	**K**	K…K
B			AAAA…A

B and cannot make a single trick under optimal play by team A. Indeed, as long as the play remains out of the gadget, it is a dominant for team A to discard one of the padding K, and it is dominant for team B to discard an A in suit that team A does not possess.

Creating the B-choice gadget that we need for our reduction is significantly more involved, since team B has no direct influence on which suit is lead. We start by creating a *unit choice gadget* to let team B influence which suit is lead next via a single trick incentive (Table 2). Consider the first team B internal hand in this gadget, say h_2. When the gadget is entered via suit s, h_2 needs to discard a card. Discarding a padding A is suboptimal because team A h_1 could then lead the corresponding K for free. Thus, two possibilities are left for h_2, and each leaves h_3 with a symmetrical dilemma.

- In the first case, h_3 can choose to exit the gadget via suit s_+ after 4 tricks or via suit s_- after 3 tricks.
- In the second case, h_3 can exit via suit s_- after 4 tricks or via suit s_+ after 3 tricks.

As a summary, in this `unit` choice gadget, team B can suggest one of the two exit suits and team A earns a bonus trick if it follows the suggestion. The next step is to increase the incentive associated to following team B's suggestion, so as to make accepting the bonus inescapable for team A. To this end, we create a class magnifying gadgets parameterized by a weight w. They are defined by induction, with 2−`magn` built on `unit` (Table 3) and $w+1$−`magn` built on w-magn and more instances of `unit` (Table 4).

If the two suggestions available to team B in a each `unit` gadget are mapped to left and right as in Table 4, then one can show that in team B strategy is dominated by either the strategy always choosing left or is dominated by the strategy always choosing right. Assuming a w−`magn` gadget, these two team B strategies give rise to dual team A alternatives. In the first case, *i.e.*, team B always select the left suit in the `unit` gadget, team A can exit from any suit s_i with up to $6w-i$ tricks. In the second case, team A can exit from any suit s_i with up to $5w+i$ tricks.

Table 3 Magnifier gadget of weight 2: 2−`magn`. The gadget is entered by playing in suit $s_?$ and can either exit through suit s_0 or suit s_1.

	$s_?$	s_0	s_1	Padding
`unit`	**A**	KK		
A	A**K**A			K…K
B				AAAA…A
A			A**K**A	K…K
B				AAAA…A

Table 4 Magnifier gadget of weight $w+1$, $w+1$−`magn` defined in terms of w−`magn`. Vertical bars have been added to help visualize which suits are shared across gadgets and hands. The gadget is entered by playing in suit s and can exit through suit s_i for $i \in [0, w]$. The internal w−`magn` has exit suits s^1 to s^w. Two strategies are of particular interest for team B: always choose left and always choose right. In the first case, team A can choose to exit from any suit s_i with up to $6w - i$ tricks. In the second case, team A can choose to exit from any suit s_i with up to $5w + i$ tricks.

	s	s^1	s^2	…	s^{w-1}	s^w	Padding
w−`magn`	**A**	K	K	…	K	K	
`unit`		K A K	|		|	|	
`unit`		| |	K A K		|	|	
…		| |	| |		|	|	
…		… …	… …		…	…	
…		| |	| |		|	|	
`unit`		| |	| |	K A K		|	
`unit`		| |	| |	| |	K A K		
A	A**K**A	| |	| |	| |	| |	|	K…K
B		| |	| |	| |	| |	|	AAAA…A
A		A**K**A	| |	| |	| |	|	K…K
B			| |	| |	| |	|	AAAA…A
…			| |	| |	| |	|	
…			………	… …	…	…	…
…					| |	|	
A					A**K**A	|	K…K
B						|	AAAA…A
A						A**K**A	K…K
B							AAAA…A
	s	s_0	s_1	…	s_{w-1}	s_w	Padding

Table 5 Team B choice gadget: B−choice. The gadget is entered by playing in suit s and can either exit through suit s_+ or suit s_-. Team B selects which suit the gadget exits from after 13β tricks are played, with $\beta = n + m$. However, team A can choose to force the exit suit at the cost of only winning 12β tricks at most in the gadget.

	s	s_1	$s_2 \dots$	s_β	$s_{\beta+1}$	$s_{\beta+2} \dots$	$s_{2\beta}$	s_+	s_-	Padding
2β−magn	**A**	K	K … K	K	K	… K				
A		A	A … A					**K**		K…K
B										A…AA…A
A				A	A	… A		**K**		K…K
B										A…AA…A

This construction means that the further team A wants to stray away from the leftmost or rightmost path suggested by team B, the larger the number of bonus tricks team A needs to give up. The fan-out of the magnifier gadget is as large as its weight, but if we group the left half of the output together, and the right half of the output, then we obtain a gadget with only two exits suits where the cost for team A to switch exit suit is proportional to the weight. More precisely, the opportunity cost of not following team B suggestion in the resulting choice gadget is half the weight of the underlying magnifier. For our reduction, an opportunity cost of $\beta = n + m$ is sufficient. Table 5 thus defines the B−choice gadget with a single entering suit s and two exit suits s_+ and s_-.

▶ **Lemma 6.** *When the B−choice gadget is entered, team B chooses one of the following two alternatives:*

1. *the game exits via suit s_+ after 13β tricks or the game exits via suit s_- after 12β tricks.*
2. *the game exits via suit s_- after 13β tricks or the game exits via suit s_+ after 12β tricks.*

In either alternative, team A chooses whether to cash $n+m$ bonus tricks and exit via the suit intended by team B, or not to cash any bonus tricks and to exit via the other suit.

3.3 Formula gadgets

Now that we have defined choice gadgets for both teams, we can encode the CNF formula that lies at the heart of the QBF instance we want to reduce. Conjunctions and disjunctions respectively correspond to team B and team A choices. Our goal is therefore to adapt the team B choice gadget to have more than two arguments. The first step is to observe that a m-argument conjunction $a_1 \wedge a_2 \wedge \cdots \wedge a_{m-1} \wedge a_m$ can be seen as a sequence of nested 2-argument conjunctions $a_1 \wedge (a_2 \wedge (\cdots \wedge (a_{m-1} \wedge a_m) \dots))$. This prompts nesting $m - 1$ instances of a w−choice gadget. Unfortunately, this direct idea would result in different number of tricks won by team A based on which conjunct team B elects, even when team A follows each every suggestion. To prevent the nesting depth of the conjuncts from breaking the symmetry, we can compensate conjuncts that come early in the nesting through additional tricks in a *waiting gadget*.

The waiting gadget is parameterized with a positive integer delay t, and is written t−wait (Table 6). It has a single entering suit and a single exiting suit, and a single joint strategy is optimal for both teams that results in exactly t tricks being won by team A while in this gadget. In other words, a t−wait just gives t tricks to team A.

Combining waiting gadgets with appropriate delay and nested choices for team B are all the ingredients needed to build a conjunction gadget (Table 7). If there are m conjuncts, then conj has one entering suit s and m exiting suit, d_i for $i \in [1, m]$. Team B selects which suit the gadget exits

Table 6 Waiting gadget with delay t: t-`wait`. The gadget is entered by playing in suit s_0 exits through suit s_t. Once the gadget is entered via s_0, the dominating strategy for team A is to lead each one of its As, before leading s_t. Team A wins exactly t tricks when the game goes through this gadget.

	s_0	$t-1$ times	s_t	Padding
A	**A**	A … A	**K**	K…K
B		K … K		AAA…A

Table 7 Conjunction gadget with m conjuncts: `conj`. The gadget is entered by playing in suit q_{n+1} and can either exit through suit d_i for any $i \in [1, m]$. Team B selects which suit the gadget exits from after $13\beta(m-1)$ tricks are played. However, team A can choose to force the exit suit at the cost of winning no more than $13\beta(m-2) + 12\beta$ tricks in the gadget.

	q_{n+1}	d_1	d_2	d_3	…	d_{m-2}	$d_{m-1}d_m$
B-`choice`	**A** K	K					
B-`choice`	|	A K	K				
B-`choice`	|	|	A K	K			
…	…	…	…	………			
B-`choice`	|	|	|		A K	K	
B-`choice`	|	|	|		|	A **K**	**K**
$13\beta(m-2)$-`wait`	A **K**	|	|		|		
$13\beta(m-3)$-`wait`		A **K**	|		|		
$13\beta(m-4)$-`wait`			A **K**		|		
…				…	…		
13β-`wait`					A **K**		
	q_{n+1}	d_1	d_2	d_3	…	d_{m-2}	$d_{m-1}d_m$

from after $13(m-1)\beta$ tricks are played. However, team A can choose to sacrifice β tricks and force the exit suit, thereby winning no more than $13(m-1)\beta - \beta$ tricks in the gadget.

The *disjunction gadget* is a comparatively much simpler construction and one can see it is simply a slightly more general form of the A-`choice` gadget. For each disjunct $\bigvee_{j=1}^{m_i} l_i^j$, we create a gadget entered via suit d_i and exited after a single trick in any suit corresponding to a literal in the disjunct (Table 8).

3.4 Quantifier gadgets

The first part of the QBF instance we want to reduce to $\mathcal{B}_{\mathcal{A}}(\_, \_, 2)$ is a sequence of alternating quantifiers on the formula variables. We show in Table 9 how to encode each quantifier. If a

Table 8 Disjunction gadget with m_i disjuncts: `disj`. The gadget is entered by playing in suit d_i and can exit through either suit l_i^j for any $j \in [1, m_i]$. Team A selects which suit the gadget exits from after a single trick is played.

	d_i	l_i^1	l_i^2	…	$l_i^{m_i}$	Padding
A	**A**	**K**	**K**	…	**K**	K…K
B				…		AAA…AA…A

Table 9 Quantifier gadget corresponding to variable v_i: i-quant. The gadget embeds a choice gadget and is connected to the rest of the reduction via 5 suits. The type of choice gadget embedded within it is determined by the type of quantifier: if we need an existential quantifier, then we embed a Team A choice gadget, whereas if we need a universal quantifier, then a Team B choice gadget is embedded. The gadget can be entered from 3 distinct suits $q_i, v^i_\bot, v^i_\top$ and can either exit via suit q_{i+1} or suit c_i to conclude the game.

	q_i	$v^i_\bot$ $v^i_\top$		c_i q_{i+1}	Padding
choice	**A**KK				
A	A	**A**	K		K…K
B					AAAA…A
A	A	**A**	K		K…K
B					AAAA…A
A			AA	**K** **K**	K…K
B					AAAAA…A

Table 10 Variable gadgets for v_i, written i-var and $\neg i$-var.

			$v^i_\top$	$v^i_\bot$	Padding
i-var	A	**A** … **A** **K**			K … K
	B				A … A A … A
$\neg i$-var	A		**A** … **A** **K**		K … K
	B				A … A A … A

A-choice gadget is embedded, then we have an existential quantifier, whereas if a B-choice gadget is embedded, then we have a universal quantifier.

We have seen in Section 3.2 how to build a choice gadget that lets team B express a decision preference. In that gadget, Team A can choose to sacrifice potential tricks in order to keep control of that decision. We now show how we can ensure that it is losing for Team A not to follow Team B's preference.

To do so, we define the *Conclusion gadget of cost* c, written c-ccl (Table 11). This gadget has one entry suit per quantifier and the game is virtually over as soon as it is entered. The conclusion gadget has two internal hands, one for each team. Beside the A in the entry suit, the Team A hand has a fixed number c of losing cards and is filled with winners otherwise. If c tricks have been played before the gadget is entered, then Team A will have had enough time to discard of the losers and would win all the remaining tricks. Otherwise, Team A is bound to lose at least one trick via a loser left.

By ensuring that Team A always brings the game to a c-ccl gadget and tuning the cost c, *i.e.*,

Table 11 Conclusion gadget of cost c: c-ccl. The gadget is entered by playing in any suit c_i for $i \in [1, n]$ and concludes the game. Team A wins if and only if such this conclusion gadget is entered after at least c tricks have been played in the game.

	c_1 c_2 … c_n	n times	c times	Padding
A	**A** **A** … **A**		K … K	A … A
B		A … A	A … A	K … K

Table 12 Example reduction for $\exists v_1 \forall v_2 \exists v_3 \forall v_4 (v_1 \vee v_2 \vee v_4) \wedge (\neg v_1 \vee \neg v_2 \vee \neg v_3) \wedge (\neg v_4 \vee v_2 \vee v_1)$.

	$q_1 v_\top^1 v_\bot^1 c_1 q_2 v_\top^2 v_\bot^2 c_2 q_3 v_\top^3 v_\bot^3 c_3 q_4 v_\top^4 v_\bot^4 c_4 q_5 d_1 d_2 d_3 l_1^1 l_1^2 l_1^3 l_2^1 l_2^2 l_2^3 l_3^1 l_3^2 l_3^3$
1-quant	A A A K K
2-quant	| | | A A A K K
3-quant	| | | | | | A A A K K
4-quant	| | | | | | | | | A A A K K
conj	| | | | | | | | | | | | A K K K
disj	| | | | | | | | | | | A | | K K K
disj	| | | | | | | | | | | | A | | | K K K
disj	| | | | | | | | | | | | | A | | | | | K K K
1-var	K | | | | | | | | | | | | | A | | | | | A
¬1-var	| K | | | | | | | | | | | | | A | | | | |
2-var	| | | K | | | | | | | | | | | A | | | | A
¬2-var	| | | | K | | | | | | | | | | | A | | |
¬3-var	| | | | | | K | | | | | | | | | | A | |
4-var	| | | | | | | | K | | | | | | | | | A |
¬4-var	| | | | | | | | | K | | | | | | | | | A
c-ccl	| A | | A | | A | | A | | | | | | | | | |
	$q_1 v_\top^1 v_\bot^1 c_1 q_2 v_\top^2 v_\bot^2 c_2 q_3 v_\top^3 v_\bot^3 c_3 q_4 v_\top^4 v_\bot^4 c_4 q_5 d_1 d_2 d_3 l_1^1 l_1^2 l_1^3 l_2^1 l_2^2 l_2^3 l_3^1 l_3^2 l_3^3$

setting it high enough, we can ensure that Team A cannot sustain missing out on any bonus trick in Team B's choice gadgets. This gives team B full choice of the exit suit in these gadgets.

3.5 Assembling the gadgets together

The gadgets can now be assembled together to create the $\mathcal{B}_\mathcal{A}(\_, \_, 2)$ instance corresponding to the input TQBF. For example, the reduction for formula $\exists v_1 \forall v_2 \exists v_3 \forall v_4 (v_1 \vee v_2 \vee v_4) \wedge (\neg v_1 \vee \neg v_2 \vee \neg v_3) \wedge (\neg v_4 \vee v_2 \vee v_1)$ gives Table 12. We set $c := 7n/2 + 13\beta(m + n/2 - 1) + 3$.

▶ **Theorem 7.** *The $\mathcal{B}_\mathcal{A}(\_, \_, 2)$ fragment is* PSPACE*-hard.*

4 Conclusions and perspectives

A convenient medium to represent and compare the complexity results on trick-taking card games is to draw a graph between different fragments of $\mathcal{B}(\eta, s, r)$. In this graph, an arrow between two fragments indicates that one is a subclass of the other. Adding the results obtained in the paper to the ones obtained by [17, 18] and [4] gives the *complexity landscape for trick-taking card games* shown in Figure 4, in the appendix.

We can see in the landscape that $\mathcal{B}_\mathcal{A}(\_, \_, 2)$ is PSPACE-complete but adding any further constraint on the fragment would immediately result in the problem being polynomially solvable.

Many actual trick-taking card games also feature a *trump suit* and potentially different values for tricks based on which cards were involved. Such a setting can be seen as a direct generalization of ours, but remains in PSPACE. As a result, any hardness result in our setting trivially carries over to point-based trick-taking card games involving trumps. Conversely, tractability results are not directly guaranteed to generalize, but the specific results we obtain in Section C are also true for settings with points and trumps.

In terms of future work, the most pressing issue is to determine whether bounding the number of suits would bring the problem to polynomial-time solvable. Although the proof for Theorem 7 is much more involved than that of Theorem 9, it appears that the all-trick constraints does not influence the hardness of trick-taking card games along the ranks dimension. The problem remains open for the other two dimensions. In particular, a promising line of inquiry is to settle the complexities of

$\mathcal{B}_{\mathcal{A}}(6,\_,\_)$ and $\mathcal{B}(2,\_,\_)$ so has to contrast them, respectively, with the hardness of $\mathcal{B}(6,\_,\_)$ and the tractability of $\mathcal{B}_{\mathcal{A}}(2,\_,\_)$.

References

1 Jean-Francois Baffier, Man-Kwun Chiu, Yago Diez, Matias Korman, Valia Mitsou, André van Renssen, Marcel Roeloffzen, and Yushi Uno. Hanabi is NP-complete, even for cheaters who look at their cards. In *8th International Conference on Fun with Algorithms (FUN)*, La Maddalena, Italy, June 2016. arXiv:1603.01911.

2 Édouard Bonnet. The complexity of playing Durak. In *25th International Joint Conference on Artificial Intelligence (IJCAI)*, New York, USA, July 2016.

3 Édouard Bonnet, Florian Jamain, and Abdallah Saffidine. Havannah and Twixt are PSPACE-complete. In H. van den Herik, Hiroyuki Iida, and Aske Plaat, editors, *8th International Conference on Computers and Games (CG)*, volume 8427 of *Lecture Notes in Computer Science*, pages 174–185. Springer, Yokohama, Japan, August 2013. arXiv:1403.6518.

4 Édouard Bonnet, Florian Jamain, and Abdallah Saffidine. On the complexity of trick-taking card games. In Francesca Rossi, editor, *23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 482–488, Beijing, China, August 2013. AAAI Press.

5 Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, 2015.

6 Michael Buro, Jeffrey R. Long, Timothy Furtak, and Nathan Sturtevant. Improving state evaluation, inference, and search in trick-based card games. In *21st International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

7 Peter I Cowling, Edward J Powley, and Daniel Whitehouse. Information set Monte Carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games (TCIAIG)*, 4(2):120–143, 2012.

8 Erik Demaine, Martin Demaine, Ryuhei Uehara, Takeaki Uno, and Yushi Uno. Uno is hard, even for a single player. In *Fun with Algorithms*, pages 133–144. Springer, 2010.

9 Ian Frank and David Basin. A theoretical and empirical investigation of search in imperfect information games. *Theoretical Computer Science*, 252(1):217–256, 2001.

10 Matthew L. Ginsberg. GIB: Imperfect information in a computationally challenging game. *Journal of Artificial Intelligence Research (JAIR)*, 14:303–358, 2001.

11 David N.L. Levy. The million pound Bridge program. In *Heuristic Programming in Artificial Intelligence: The First Computer Olympiad*, pages 95–103. Ellis Horwood, 1989.

12 David Lichtenstein and Michael Sipser. Go is polynomial-space hard. *Journal of the ACM (JACM)*, 27(2):393–401, 1980.

13 Jeffrey Long, Nathan R. Sturtevant, Michael Buro, and Timothy Furtak. Understanding the success of perfect information Monte Carlo sampling in game tree search. In *24th AAAI Conference on Artificial Intelligence (AAAI)*, pages 134–140, 2010.

14 Thomas J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16(2):185–225, 1978.

15 Leon Sterling and Yossi Nygate. Python: an expert squeezer. *The Journal of Logic Programming*, 8(1-2):21–39, 1990.

16 Nathan R. Sturtevant and Adam M. White. Feature construction for reinforcement learning in Hearts. In H.Jaap Herik, Paolo Ciancarini, and H.H.L.M.(Jeroen) Donkers, editors, *Computers and Games*, volume 4630 of *Lecture Notes in Computer Science*, pages 122–134. Springer, 2006.

17 Johan Wästlund. A solution of two-person single-suit Whist. *The Electronic Journal of Combinatorics*, 12(1):R43, 2005.

18 Johan Wästlund. Two-person symmetric Whist. *The Electronic Journal of Combinatorics*, 12(1):R44, 2005.

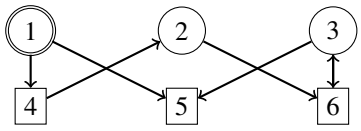

Figure 2 An instance of GG with 1 as the starting vertex.

A Generalized Geography

Generalized Geography (GG) is a zero-sum two-player game over a directed graph with one vertex token. Players take turn moving the token to an adjacent vertex and thereby removing the origin vertex. The player who cannot play anymore loses. Figure 2 presents an instance of GG on a bipartite graph.

Deciding the winner of a GG instance is PSPACE-complete [14], and GG was used to prove PSPACE-hardness for numerous games including GO [12], UNO [8], and more recently, HAVANNAH [3]. Lichtenstein and Sipser have shown that GG remains PSPACE-hard even if the graph is assumed to be planar, bipartite, and of degree 3 [12]. GG was also used to prove both hardness results in the foundation paper on the complexity of trick-taking card games [4]. Contrary to the previous reduction [4, Theorem 2], the next Section takes advantage of the possibility of bounding the degree of the graph while retaining hardness.

B $\mathcal{B}(\_,\_,2)$ is PSPACE-complete

We present a polynomial reduction ϕ from bipartite GG on graphs of degree 3 to $\mathcal{B}(\_,\_,2)$.

An instance of GG on a bipartite graph is given by $(G = (V_A \cup V_B, E_{AB} \cup E_{BA}), v_1)$ where $v_1 \in V_A$ denotes the initial location of the token. Let $m = m_{AB} + m_{BA} = |E_{AB}| + |E_{BA}|$ the number of edges and $n = n_A + n_B = |V_A| + |V_B|$ the number of vertices. We construct an instance of $\mathcal{B}(\_,\_,2)$ using $m + n(m+1)$ suits, and $2n$ hands with $m+1$ cards each as follows.

Each vertex $v \in V_A$ (resp. $\in V_B$) is encoded by a hand h_v owned by team A (resp. B). We add n additional dummy hands, hand h_{A_1} up to hand $h_{A_{n_B}}$ for team A and h_{B_1} up to hand $h_{B_{n_A}}$ for team B.

Each edge $(s,t) \in E_{AB}$ (resp. E_{BA}) is encoded by a suit $s_{s,t}$ of length 2, for instance $\{\mathrm{AK}\}$. The cards in suit $s_{s,t}$ are dealt such that hand h_s receives K, hand h_t receives A. We add $n(m+1)$ additional dummy suits $s_{A_{i,j}}$ for all $i \in \{1,\dots,n_B\}$ and $j \in \{1,\dots,m+1\}$ and $s_{B_{i,j}}$ for all $i \in \{1,\dots,n_A\}$ and $j \in \{1,\dots,m+1\}$.

For all $i \in \{1,\dots,n_B$ (resp. $n_A)\}$, hand h_{A_i} (resp. h_{B_i}) receives A in all suits $s_{A_{i,j}}$ for $j \in \{1,\dots,m+1\}$, while hand h_i (resp. h_{n_A+i}) receives K in all suits $s_{A_{i,j}}$ for $j \in \{1,\dots,m-\deg(v_i)\}$. Recall that $1 \leqslant \deg(v_i) \leqslant 3$ and note that the suits $s_{A_{i,j}}$ with $m-2 \leqslant j$ might only feature A (with no K).

The goal of team A is to make at least $m_{BA}+1$ tricks. Intuitively, for team A (resp. B) playing in a suit $s_{B_{i,j}}$ (resp. $s_{A_{i,j}}$) makes them lose all the remaining tricks (provided, of course, that hand h_{B_i} (resp. h_{A_i}) has not discarded its corresponding A) so it cannot be good. The interesting and difficult part of this bridge game would only occur in playing accurately the suits $s_{s,t}$ between hands h_i for $i,s,t \in \{1,\dots,n\}$, that is the non dummy suits and the non dummy hands. Remark that this part simulates GG on the instance (G, v_1).

▶ **Lemma 8.** *Player 1 has a winning strategy in (G, v_1), then team A can make $m_{BA}+1$ tricks in $\phi(G, v_1)$.*

h_1		h_2		h_3	
$s_{B_{1,1}}$	K	$s_{B_{2,1}}$	K	$s_{B_{3,1}}$	K
⋮		⋮		⋮	
$s_{B_{1,6}}$	K	$s_{B_{2,6}}$	K	$s_{B_{3,5}}$	K
$s_{1,4}$	K	$s_{2,6}$	K	$s_{3,5}$	K
$s_{1,5}$	K	$s_{4,2}$	A	$s_{3,6}$	K
				$s_{6,3}$	A

h_4		h_5		h_6	
$s_{A_{1,1}}$	K	$s_{A_{2,1}}$	K	$s_{A_{3,1}}$	K
⋮		⋮		⋮	
$s_{A_{1,6}}$	K	$s_{B_{2,6}}$	K	$s_{A_{3,5}}$	K
$s_{1,4}$	A	$s_{1,5}$	A	$s_{2,6}$	A
$s_{4,2}$	K	$s_{3,5}$	A	$s_{3,6}$	A
				$s_{6,3}$	K

h_{A_1}		h_{A_2}		h_{A_3}	
$s_{A_{1,1}}$	A	$s_{A_{2,1}}$	A	$s_{A_{3,1}}$	A
$s_{A_{1,2}}$	A	$s_{A_{2,2}}$	A	$s_{A_{3,2}}$	A
⋮		⋮		⋮	
$s_{A_{1,8}}$	A	$s_{A_{2,8}}$	A	$s_{A_{3,8}}$	A

h_{B_1}		h_{B_2}		h_{B_3}	
$s_{B_{1,1}}$	A	$s_{B_{2,1}}$	A	$s_{B_{3,1}}$	A
$s_{B_{1,2}}$	A	$s_{B_{2,2}}$	A	$s_{B_{3,2}}$	A
⋮		⋮		⋮	
$s_{B_{1,8}}$	A	$s_{B_{2,8}}$	A	$s_{B_{3,8}}$	A

Figure 3 Reduction from Figure 2.

Proof. Let ψ be the winning strategy of player 1, mapping a path $v_1 \dots v$ ending in V_A to a vertex v' in V_B. We define the following winning strategy for team A in $\phi(G, v_1)$. When in a hand h_{A_i} for some i, cash all the remaining A (all the remaining tricks). When in a hand h_i for some i, cash all A (they are in suits $s_{s,t}$). Then ψ tells you which of the K in a non dummy suit (suits of the form $s_{s,t}$) to play. Keeping track of which hands have taken the lead so far (without counting several times a hand which cashes some A) $h_1 h_{k_2} h_{k_3} \dots h_i$, play the K in $s_{i,t}$ where $\psi(v_1 v_{k_2} v_{k_3} \dots v_i)$ is the t-ieth vertex.

As for discarding, hands h_i for $i \in \{1, \dots, n_A\}$ can throw away any of the dummy K in suits s_{B_j} in any order. Hands h_{A_i} for $i \in \{1, \dots, n_B\}$ have to be more careful. They can start by discarding the A in suits $s_{A_{i,m-\deg(v_i)+1}}$ up to $s_{A_{i,m+1}}$. Then, they can discard in the same suit hand h_{n_A+i} has discarded its K at some previous trick. Indeed, hand h_{n_A+i} does not discard at most $\deg(v_i)$ times in the part of the game in non dummy hands.

From a hand h_j, team B cannot play towards a non dummy hand h_i of team A which has already taken the lead, since by construction, h_i has cashed all the A in non dummy suits, and in particular the one in $s_{i,j}$, so the K owns by team B has gone. Team B, had therefore two losing options: follow an actual GG game simulation where he will eventually lose, or play in a dummy suit and lose all the remaining tricks. All in all, team B cannot cash more than its number of A in non dummy suits which is equal to m_{AB}. So, team A will make at least the complement $m_{BA} + 1$. ◀

The same result also applies to team B. Therefore team A has a winning strategy in $\phi(G, v_1)$ if and only if the first player has a winning strategy in the instance (G, v_1) of GG. The reduction is thus complete, leading to PSPACE-hardness.

▶ **Theorem 9.** *$\mathcal{B}(\_, \_, 2)$ is* PSPACE*-complete.*

C Tractability results

We now show that bounding the number of hands when bounding the number of ranks per suit results in a tractable class.

▶ **Theorem 10.** *$\mathcal{B}(N, \_, R)$ is in* P.

Proof. We will show that any position with N hands, s suits, and R ranks can be solved by minimax with transposition table using $O(R(s+1)^{(N+1)^R+1})$ recursive calls to minimax. If N and R are held constant, then this is polynomial time in terms of s. This will prove that $\mathcal{B}(N, \_, R)$ is polynomial.

Assume that we have a bounded number N of hands and a bounded number R of ranks.

For each suit i, each card in i is either distributed to a hand indexed in $\{1, N\}$ or played in a previous trick. This means that there are exactly $(N+1)^R$ ways of distributing all the cards in i among the hands and the deck of previous tricks. Put another way, each suit i can be mapped into a number $d_i \in [1, (N+1)^R]$ without loss of information.

For P a position and $d \in [1, (N+1)^R]$ a card distribution, let σ_d^P be the number of suits having this distribution in the input position. Let τ^P be the lead turn and ν^P be the number of tricks won by team A so far. Keeping in mind that the name of the suit is irrelevant for the question of determining if team A can make ν tricks, and observing that suits with the exact same distribution can play a symmetrical role, we can represent P as a vector $\chi(P) = \langle \sigma_1^P, \sigma_2^P, ..., \sigma_{(N+1)^R}^P, \tau^P, \nu^P \rangle$. This representation does not incur any loss of relevant information. Indeed, any two intermediate positions P_1 and P_2 having the same card distribution up to renaming, $\chi(P_1) = \chi(P_2)$, lead to the same optimal number of tricks won by team A.

When we restrict ourselves to positions P having at most s suits, N hands, and R ranks, σ_d^P takes no more than $s+1$ values, for any d, τ^P takes no more than N values, and ν^P takes no more than $\frac{sR}{N}$ distinct values. Therefore, the co-domain of χ has cardinality bounded by $(s+1)(s+1)\dots(s+1)N\frac{RS}{N} = Rs(s+1)^{(N+1)^R}$. As a result, χ can be used to derive a hashing function from any positions with N hands, at most s suits and R ranks into a transposition table with at most $Rs(s+1)^{(N+1)^R}$ entries. Although this hashing function admits collisions, two position having the same hash value are guaranteed to have the same optimal number of tricks for team A.

To solve an input position P with s suits, N hands, and R ranks, we can run a minimax search with transposition tables using χ for hashing. If N and R are held constant, we can prove that the search runs in time polynomial in s. Indeed, an internal node will only make recursive calls to minimax if the hash of the corresponding position is not in the transposition table yet. After the recursive calls return and the value of the node is computed, this value is added to the transposition table for the first and unique time. Since the maximum size of the transposition table is polynomial in s, only a polynomial number of internal nodes will make a recursive call. ◀

We list here the tractability of a couple straightforward classes, so as to better see the frontier between known easy classes, known hard classes, and open problems. The first observation is that assuming a single rank per suit makes it impossible for the opponent team to win any trick. This result in all $\mathcal{B}(\_, \_, 1)$ instance being true.

▶ Proposition 2. $\mathcal{B}(\_, \_, 1)$ is in PTIME.

The second observation is that a simple criterion tested on each suit independently suffices to determine if team A can win all tricks in instances with a single team A hand. Indeed, if team A has a single hand, then it does not have any discarding opportunity and all its cards will be lead.

▶ Proposition 3. $\mathcal{B}_{\mathcal{A}}(2, \_, \_)$ is in PTIME.

Proof. Let s be a suit. Let r be the smallest rank of team A in suit s, and let r_i be the highest rank of team B hand h_i in s. Let w_i be the number of cards of team A of rank larger than r_i. Then team A can make all tricks in suit s if and only if $\forall i, r > r_i \vee w_i > |h_i(s)|$ ◀

▶ Proposition 4. $\mathcal{B}(\_, S, R)$ is in PTIME.

Proof. There is a constant number of instances, so they can be pre-computed. ◀

Table 13 The remaining cards in the quantifier gadget i-quant, when the corresponding team decides to *set the variable x_i to true*.

	q_i	$v^i_\bot v^i_\top$		$c_i q_{i+1}$	Padding
choice	AKK				
A	A	A	K		K…K
B					AAAA…A
A	A	A	K		K…K
B					AAAA…A
A			AA	K K	K…K
B					AAAAA…A

Table 14 The remaining cards in the quantifier gadget i-quant, when the corresponding team decides to *set the variable x_i to false*.

	q_i	$v^i_\bot v^i_\top$		$c_i q_{i+1}$	Padding
choice	AKK				
A	A	A	K		K…K
B					AAAA…A
A	A	A	K		K…K
B					AAAA…A
A			AA	K K	K…K
B					AAAAA…A

D Proof of Theorem 7

Proof. We show that the polynomial-time reduction $\mathcal{R}$ that we presented from the PSPACE-hard problem TQBF is sound.

ψ is true $\Rightarrow$ team A wins. Let us assume that formula $\psi := \exists x_1 \forall x_2 \exists x_3 \ldots \forall x_n \phi(x_1, \ldots, x_n)$ is true. We consider the problem of deciding whether or not ψ is true as a two-player game between an existential player and a universal player. Let χ be a winning strategy tree for the existential player. We describe a strategy for team A to make all the tricks in $\mathcal{R}(\psi)$. We recall that the hand q_1 of the 1-quant gadget leads the first trick. Team A cashes (q_1, A) and discards any K in the padding of each other hand. In general, this will be the way team A discards.

Existential quantifier. After team A makes a trick with (q_i, A) for an odd $i \in [1, n]$ (i.e., entered an *existential quantifier* gadget), team A plays the K towards the A of the hand that also owns $(v^i_\bot, A)$ (resp. $(v^i_\top, A)$) if χ advocates to set the variable x_i to true (resp. to false), cashes this ace, and finally exits the gadget by playing (q_{i+1}, K). It is important that team A does not cash the A that is underlined in Table 13 and 14.

Universal quantifier. After team A makes a trick with (q_i, A) for an even $i \in [1, n]$ (i.e., entered a *universal quantifier* gadget), team A makes 13β tricks in the internal B-choice gadget, and ends up exiting either via suit s_+ or via suit s_- of Table 5, accordingly to the discards of team B. This, in turn, decides if the remaining cards in i-quant correspond to Table 13 or 14. We *move* in the tree χ to the corresponding child node: $x_i \leftarrow \top$ in the former case and $x_i \leftarrow \bot$ in the latter.

Up to this point, team A has made $2n + (13\beta + 3)n/2$ tricks and enters the conj gadget. Thanks to χ, the joint assignment performed by team A and team B, defined by which of $(v^i_\bot, A)$ or $(v^i_\top, A)$

is remaining, satisfies the 3CNF formula ϕ.

Conjunction gadget (aka clause-selector) and Disjunction gadget (aka literal-selector). Team A accepts to exit `conj` with (d_i, K), where $i \in [m]$ is *chosen* by team B, and makes $13\beta(m-1)$ tricks in the gadget. There is at least one literal l_i^j with $j \in [1, m_i]$ that has been set to true by the joint assignment. Team A plays the (l_i^j, K) for the (l_i^j, A). From this hand, team A plays the (v_q^k, K) with $q = \bot$ if $l_i^j = \neg x_k$ and $q = \top$ if $l_i^j = x_k$ for the (v_q^k, A). Thereby, team A reenters the k`-quant` gadget, plays the K towards the underlined A (that is why it was important *not* to cash this A earlier), and exits in the c`-ccl` gadget. At this point, team A has cashed $2n + (13\beta+3)n/2 + 13\beta(m-1) + 3 = 7n/2 + 13\beta(m + n/2 - 1) + 3 = c$ tricks. Therefore, all the c Ks in the padding of the Conclusion gadget has already been discarded. Thus, there are only aces left in the hand; which means team A wins all the tricks.

ψ is false $\Rightarrow$ team B wins. We now describe a strategy for team B to prevent team A from doing all the tricks, assuming ψ is false. Let χ be a winning strategy tree for the universal player. First, the discards of team B obey the following rule: discard the A in suits where the K is gone (also, discarding Ks is safe). This rule can be applied in every gadget but the Unit choice gadget. Also, whenever a hand h (not leading the next trick) of team A contains only Ks, it is not necessary anymore for team B to keep the corresponding As, since h can no longer get the lead.

Team A can get at most $3n + (13\beta+3)n/2$ tricks from the first run through the quantifier gadgets. This corresponds to the $2n + (13\beta+3)n/2$ tricks as described in the winning strategy for team A (when ψ is true) plus at most n *underlined As* (see Table 13 and 14). When team A *assigns a value* to a variable x_i with an odd $i \in [1, n]$ by keeping either the card $(v_\bot^i, A)$ or the card $(v_\top^i, A)$, one can move to the corresponding node in the tree χ. When entering a universal quantifier i`-quant`, team B discards in the B`-choice` gadget (always the rightmost A or always the leftmost A) in order to either make team A eventually cash $(v_\bot^i, A)$ (resp. $(v_\top^i, A)$) if χ advocates to set x_i to true (resp. to false) or to cost β tricks to team A. When entering the `conj` gadget, there is necessarily a clause $\bigvee_{j=1}^{m_i} l_i^j$ of ϕ which is not satified by the *joint assignment* of team A and team B. Therefore, team B discards in the successive B`-choice` gadgets to reach suit d_i of Table 7, or to cost at least β tricks to team A. At this point, we should observe that neither at this stage nor in the B`-choice` gadget of universal quantifiers, can team A contradict the choices of the discards of team B. Indeed, before finally entering the Conclusion gadget, team A can only compensate with n *underlined* As and at most $m-1$ As in the variable gadget. Those at most $n + m - 1 < \beta$ extra tricks are not sufficient for team A to reach the c tricks necessary to win. Thus, all the $(v_\top^k, A)$ (resp. $(v_\bot^k, A)$) such that x^k (resp. $\neg x^k$) is a literal of clause $\bigvee_{j=1}^{m_i} l_i^j$ have already been cashed and team A cannot reach the Conclusion gadget with enough tricks.

◀

E Complexity landscape

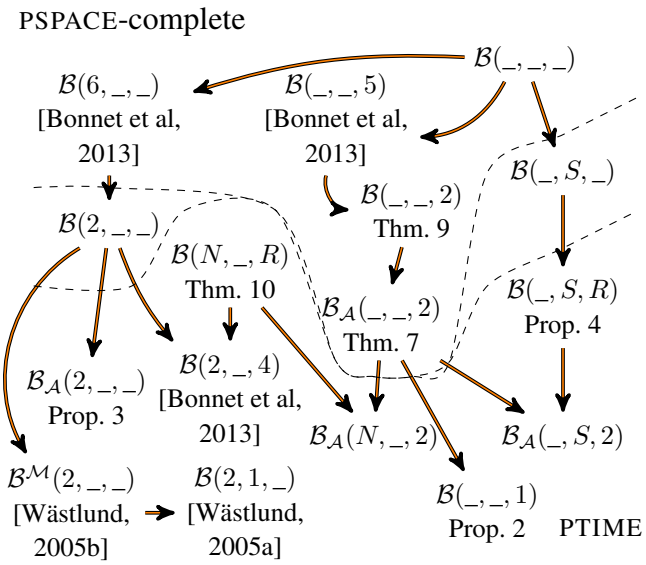

Figure 4 Summary of the hardness and tractability results known for the fragments of $\mathcal{B}(\eta, s, r)$.