Opleiding Informatica/Wiskunde

The Crew:
a Strategy Analysis

Eef Witteveen

Supervisors:
Floske Spieksma (MI) & Rudy van Vliet (LIACS)

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

27/August/2025

Abstract

In this thesis we describe various agents for ‘The Crew’ for both phases of the game (task-taking and playing) and evaluate their performances. The agents follow rules based on intuition of human players and Monte Carlo methods. We conclude that the Monte Carlo Tree Search agent using Upper Confidence bound for Trees performs the best in both playing phases. We also provide an analysis of the winnability of a 2-player, 1-task variant of the game and conclude that 99.72% of games of this type are winnable.

Contents

1 Introduction

In this thesis we will discuss and analyse the game "The Crew". *The Crew* is a cooperative trick-taking game, where the goal is to work together to ensure that certain players will win certain tricks based on task cards given to the players.

Within this thesis we will analyse and solve a 2-player variant of the game on winnability. We will also define agents that either use rule-based algorithms modelled after player-intuition or Monte Carlo methods and evaluate them on their ability to win a 3-player game for various numbers of tasks.

1.1 Related Work

Various research has already been done relating to determining optimal strategies for *The Crew*.
An analysis of the complexity of various (sub-)problems of *The Crew* where players have perfect information has been performed by F. Reiber [Rei21]. Multiple versions of the game are covered, for example, games with only a single value or only a single suit. Algorithms are defined to check whether a solution to such a problem can be found. It is also shown that the general unbounded case of finding a winning solution to a game of *The Crew* is NP-complete.

Moreover, S. Chan [Cha21] analyses various greedy algorithms that pick each player's next played card based on that trick alone, as well as various heuristics for both the trick-taking and task-choosing phases of the game to determine which strategy is optimal. Games of *The Crew* with 3 players with imperfect information and perfect information are considered. In the case of imperfect information, a player is only aware of their own playing cards. However, the algorithms are also dependent on the playing cards of the other players. Therefore, additional heuristics are introduced to estimate the playings cards of the other players. It is concluded that the greedy algorithm that tries to complete the tasks of all the players performs significantly better than the algorithm that tries to complete only the player's own tasks and the algorithm that picks random playable cards.

Furthermore, D. Rebstrock, et al. [RSSB24] use Generative Observation Monte Carlo Tree Search to analyse *The Crew* for a game with 4 players and imperfect information, as well as various other trick-taking card games. By using a generative model to base the decisions made by a player on, the issue of not having perfect information is overcome.

Lastly, A. de Jong [dJ21] experimentally analyses the winnability of *The Crew* by using SAT-solvers for several randomly generated games of *The Crew* including trump cards. It is shown that there are theoretical possibilities for unwinnable games. However, when generating 10,000 random games where players can select their tasks for up to 23 tasks the fraction of winnable games seems to be nearly 1. Additionally, several possible methods of determinization of the game are considered to overcome the challenge of incomplete information within the game. Determinization can be used to determine a random (valid) instance of a game with complete information based on an instance with incomplete information. This allows for applications of algorithms that use complete information on games that do not have complete information. In the case of *The Crew* this means estimating the hands of the other players, based on prior tricks and the hand of a single player. Finally, several Pure Monte Carlo players (one with perfect information, one with incomplete information that attributes random cards to the other players and one who uses determinization to attribute cards to the other players) are compared on what percentage of tasks are succesfully completed for tasks numbers up to 35. The Pure Monte Carlo player with perfect information outperformed the others, but surprisingly, of the players with imperfect information, the one attributing cards randomly outperformed the player using determinization.

1.2 Thesis overview

Section 2 of this thesis includes an overview of the rules of the game. Section 3 analyses the winnability of a two person variation of the game. Section 4 discusses the algorithms that are used in the experiments done. Section 5 highlights the most interesting results of the experiments. Section 6 concludes the thesis by summarising the results and giving directions for further research on analysing *The Crew*. Appendix A gives an overview of the complete results from the experiments.

This bachelor thesis was written as part of a double bachelor program Computer Science and Mathematics at Leiden University, under supervision of Floske Spieksma (MI) and Rudy van Vliet (LIACS).

2 Rules

We begin by outlining the rules of the game as they were used for this research paper. To allow for a simpler analysis we consider a simplified version of the rule-set.

The Crew is a trick-taking card game, like *contract bridge*, *hearts* or the Dutch game *klaverjassen*. In most trick-taking card games, in each round the starting player may play any card and the other player(s) must *follow* the suit (or colour) of the first card. If a subsequent player cannot follow, he may play any card in his hand. The player with the highest value card of the initial suit wins the trick. However, since *The Crew* is a cooperative game, the goal is not to win as many tricks as possible. The players must work together to ensure that the right people win specific tricks. We will now define the rules of *The Crew* in detail.

2.1 Base rules

Set-up
The original game of *The Crew* consists of 3-5 players. There are 36 base cards in total, namely the values 1-9 for each of the suits: blue, green, pink and yellow. These cards are divided equally over all players. If the cards do not divide evenly, a few players are given an extra card. In fact, this is only possible when there are 5 players and only the first player will be given an extra card. Next, the initial starting player (the *commander*) is chosen randomly among the players with the most cards (so any player in a game of 3-4 players and the player with the extra card when there are 5 players). The *task cards* are shuffled and a fixed number of task cards (1-10 according to the standard rules) are laid out (face up). Beginning with the starting player in clockwise order, each player must pick one of the tasks laid out in front of them until no more tasks remain. If there are more tasks than players, the initial player must then pick a second task and so forth. Task cards correspond to each of the 36 regular cards. When a player holds a task card he must obtain the corresponding regular card by winning the trick where it is played. When all tasks have been divided over the players the game begins.

Rounds
The first trick is started by the initial starting player. In all subsequent tricks, the starting player is the player who won the previous trick. During a trick, the first player may play any card in his hand, and the other players (in clockwise order) must follow the suit if possible. If a player has multiple cards of the required suit he may play any of these cards. If he does not have the required suit, the player may play any card in his hand. The player who played the highest card of the required suit wins the trick. An example of a played trick can be seen in Figure 1. The game ends when all tasks have been completed successfully, when a task is failed (the card specified by the task is taken by a wrong player) or when the hand of at least one of the players is empty. The game is won when all tasks have been completed successfully and lost otherwise.

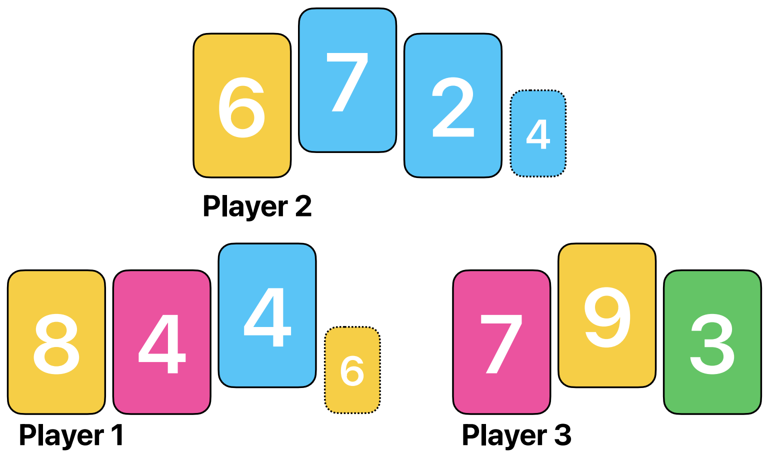


Figure 1: Example of a played trick in *The Crew*. Player 1 has the task associated with the yellow 6 and player 2 has the the task for the blue 4. Player 3 has no tasks. This trick is opened by player 1, who first plays the blue 4. player 2 must then choose between his blue 7 or blue 2 and chooses the 7 since he must win the blue 4. Player 3 has no blue cards in his hand and therefore can freely choose which card to play. Since the yellow 9 might interfere with player 1 winning the yellow 6 in later tricks, he chooses to play the yellow 9 to get rid of it.

2.2 Further rules

In addition to the above mentioned base rules there are several more rules of *The Crew* that can be incorporated to obtain a more complex game, and which are part of the original game. These rules will not be considered within this thesis, but are included for completeness.

Trump/rocket cards
In the original game in addition to the 36 normal cards, 4 black '*trump*/*rocket*' cards with values 1-4 are present in the game. These black cards operate as normal cards in the sense that when the player who starts the trick plays one of the trump cards, other players with trump cards must also play them. However, if a trump card is played when the required suit was different, the trump card wins over the required suit. Note that, like all other suits, it is only possible for a player to play trump cards on a different required suit if the player has no cards of the required suit on hand. When multiple trump cards are played, the highest value trump card wins. When the trump cards are in the game, the player with the rocket 4 card is the *commander* (the initial starting player).

Distress signal
Once per game, between tricks, players may decide unanimously to use the so-called *distress signal*. When this happens, each player passes on a card of his choice to the player on the right. In the original game this counts as a sort of 'penalty' and the win is worth slightly less.

Communication
In the original game (non-task) cards are private and each player may only communicate about his cards once per game. Communication may only happen after the tasks have been divided and only between (so not during) different tricks or before the first trick. The player shows a single card and communicates that this card is the highest, the lowest card or the only card he holds of its respective suit. A player may only communicate the card, if it is in at least one of these three categories and he may only communicate truthful information. Moreover, trump cards may never be communicated.

Task order tokens
Order tokens can be added to some or all of the tasks during the set-up phase. These tokens specify, for example, that a task should be completed first of all the tasks, last of all the tasks or before or after a specific other task.

More complicated/varied tasks
In the second version of the game ("*The Crew: Mission Deepsea*") other tasks are possible than just needing to win a specific card. For example: needing to win three cards with a blue suit, needing to win the first trick, or needing to win no tricks in the entire game.

Missions
In the original game, the number of tasks is determined by the mission-specification. The game includes 50 missions with increasing difficulty. A mission includes a fixed number of tasks (increasing as the missions get higher), a potential order in which these tasks must be completed and sometimes other special requirements, such as not being able to communicate for the duration of the mission.

3 Theoretical analysis

In this section we will analyse a game with two players. Note that in this case information is always complete, hence it is not usually a version that is played in practice. It is, however, not entirely trivial. We show, that not all games of this form are winnable by analysing a game with a single task. We determine exactly how many instances of such a game exist. Moreover, we determine the percentage of winnable games.

3.1 Formal definition of the game

First of all, we define a game of *The Crew* formally, in order to properly prove aspects of the game.

Definition 3.1 (2-player game of The Crew)**.** *An initial configuration of a 2-player The Crew game is a 7-tuple* $(C, S, V, P_1, P_2, T_1, T_2)$*, where*

1. S *the set of suits in the game, finite and non-empty.*
2. $V \subset \mathbb{Z}_{>0}$*, the set of values in the game, finite and non-empty.*
3. $C := S \times V$*, the entire set of cards present in the game; the cartesian product of* S *(the suits) and* V *(the values).*
4. $P_1, P_2 \subset C$*, where* $P_1 \cup P_2 = C$*,* $P_1 \cap P_2 = \emptyset$ *and* $0 \leq |P_1| - |P_2| \leq 1$*, the initial cards for player 1 and player 2 respectively.*
5. $T_1, T_2 \subset C$*, where* $T_1 \cap T_2 = \emptyset$*,* $0 \leq |T_1| - |T_2| \leq 1$ *and* $|T_1| \geq 1$ *the tasks for player 1 and player 2 respectively.*

That is, a game of *The Crew* has sets of suits and values. The cards (which are all unique combinations of suits and values) are used to fill the hands of both players (where player 1 has at most one card more, specifically in the case where the total number of cards is odd). The hands of both players must be disjoint subsets that (together) contain all of the cards. Player 1 and player 2 also both have a set of tasks, which are also subsets of the card-set. In this case, not necessarily all cards have to be used, but the players may not have overlapping tasks. Moreover, since in the game player 1 would be the first to pick a task, player 1 will have an extra task when dealing with an odd number of tasks. Note that in the above definition tasks are given directly to the players. In the actual game, however, there is a task-taking phase, where players choose their tasks. The initial configuration contains a set of tasks that are laid out in front of the players who can then pick their tasks, with every possible division of these tasks being a reachable state.

For the purposes and ease of our analysis, we instead consider each of the reachable states as initial configurations. Also note that when considering games with only one task, there is no difference between the two methods. Finally, in the game, task cards and playing cards are physically different cards, but for our purposes it is more convenient to consider them as subsets of the same set C, since their possible values do not differ.

In the analysis, we consider a standard game (i.e., a game with the 36 cards as described in Section 2) with a single task. This specific case is described in Definition 3.2.

Definition 3.2. *[2-player standard game of The Crew with a single task] An initial configuration of a 2-player standard The Crew game is a 7-tuple* $(C, S, V, P_1, P_2, T_1, T_2)$*, where*

1. $S := \{blue,\ green,\ pink,\ yellow\}$,
2. $V := \{1, 2, ..., 9\}$,
3. $C := S \times V$,
4. $P_1, P_2 \subset C$ *where* $P_1 \cup P_2 = C$, $P_1 \cap P_2 = \emptyset$ *and* $0 \leq |P_1| - |P_2| \leq 1$,
5. $T_1 := \{t\}$ *for some* $t \in C$, $T_2 := \emptyset$.

We also define the following notations and definitions for ease of writing.

Notation 3.3. *For* $c := (x, y) \in C$*, we write* $c_s := x$ *and* $c_v := y$*, the suit and the value of the card.*

Notation 3.4. *Let* $P_{i,s} = \{c \in P_i \mid c_s = s\}$*, the set of cards player* i *holds of suit* s*.*

Notation 3.5 (Card count). *Let* $k_i = |\{c \in P_1 \mid c_s = i\}|$*, that is* k_i *is the number of cards player 1 holds of a given suit* i*.*

Note that in a standard 2-player game, player 2 then holds $9 - k_i$ cards of suit i.

Definition 3.6 (Critical colour/suit). *The critical colour/suit is equal to* t_s*.*

Definition 3.7 (Critical card). *The critical card is the playing card present in one of the players' hand corresponding to the task card* t*.*

Next, we must define a playout of the initial game-configuration. Both players will play $n \leq 18$ tricks, specifically until the critical card is played. Player 1 is the initial starting player so he will win the first trick if both players play cards of different suits. In subsequent tricks, the player who won the prior trick (and therefore started this one) will win in such cases. We also want to enforce the requirement that the players must follow suit when possible.

Definition 3.8 (Playout). *A playout of a game is a sequence*

$$G = ((x_1, y_1, w_1), (x_2, y_2, w_2), \dots, (x_n, y_n, w_n)),$$

where $x_1, x_2, \dots x_n \in P_1$*,* $y_1, y_2, \dots y_n \in P_2$ *with* $x_i \neq x_j$*,* $y_i \neq y_j$ *for* $i \neq j$ *and (*$x_n = t$ *or* $y_n = t$*). Additionally, let* $w_0 := 1$ *and for* $i \geq 1$

$$w_i = \begin{cases} 1, & \text{if } (x_{i,s} = y_{i,s} \text{ and } x_{i,v} > y_{i,v}) \text{ or } (x_{i,s} \neq y_{i,s} \text{ and } w_{i-1} = 1) \\ 2, & \text{if } (x_{i,s} = y_{i,s} \text{ and } x_{i,v} < y_{i,v}) \text{ or } (x_{i,s} \neq y_{i,s} \text{ and } w_{i-1} = 2). \end{cases}$$

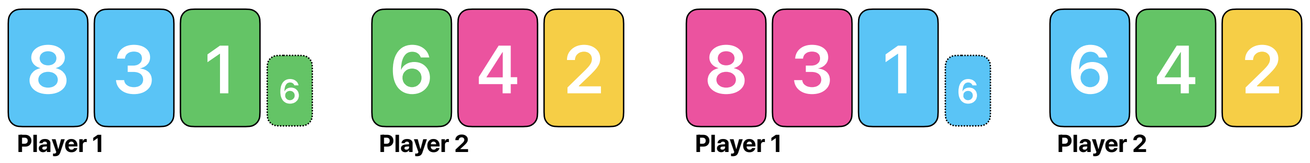
Lastly, for all $i \geq 1$ *either* $w_{i-1} = 1$ *and* $P_{2,x_{i,s}} \setminus \{y_1, ..., y_{i-1}\} = \emptyset$*, or*
$w_{i-1} = 2$ *and* $P_{1,y_{i,s}} \setminus \{x_1, ..., x_{i-1}\} = \emptyset$*, or* $x_{i,s} = y_{i,s}$*.*

Definition 3.9 (Winning/losing playout for a 2-player game with 1 task; winnable/losing game). *Let* n *be the number of tricks played in a given playout of a 2-player The Crew game with one task. The playout is called winning if* $w_n = 1$*, i.e. Player 1 wins the final trick of the game. By definition, this is the trick containing the critical card. Likewise, a losing playout is a playout such that* $w_n = 2$*. A game which has a winning playout is called winnable and a game that does not is called losing.*

Definition 3.10 (Directly winning). *Given two cards* $x, y \in C$*. We call* x *directly winning over* y *if* $x_s = y_s$ *and* $x_v > y_v$*.*

3.2 Total number of games filtered for symmetries

In order to determine how many games are winnable, we must first determine how many games there are in total. Given a game, we can create a different game by swapping, for example, two of the suits. Note that by swapping the same colours of a winning playout of the original game we get a winning playout for the second game. The same holds for a losing playout. Therefore, we do not consider these games 'unique', but rather symmetries of one another. Likewise, we can create symmetries of any given game by considering all $4!$ permutations of suits, see also Figure 2.



(a) Game instance 1

(b) Game instance 2

Figure 2: Two 'nearly-identical'/symmetrical games that can be changed into one another using a simple permutation that swaps blue to pink, pink to green and green to blue, and yellow to yellow (and vice versa), this is commonly written as $(blue, pink, green)$ in group theory. Such swaps are possible for all $4!$ permutations of the 4 suits.

To determine how many unique games there are with unique playouts, we need to filter out these symmetries. Since it is difficult to count the unique games directly, we calculate them based on the number of invariant games under each symmetry using Burnside's Lemma, see also [Rot95, Chapter 3], [KM25].

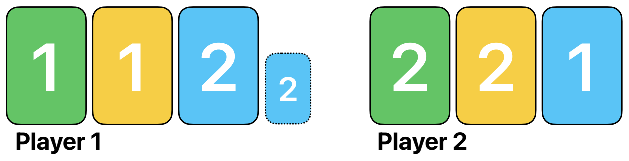
Lemma 3.11 (Burnside's lemma)**.** *Let G be a finite group that acts on a set X. For $g \in G$, let X^g denote the set of elements in X that are left invariant by g, i.e. $X^g = \{x \in X \mid g \cdot x = x\}$. The number of G-orbits of X, written as $|X/G|$, satisfies the following formula:*

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$$

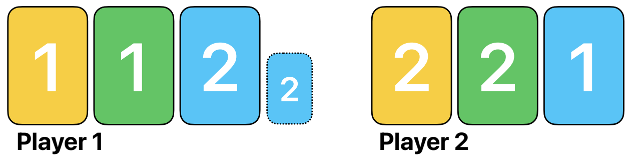
Proof. The proof of this lemma can be found in [Rot95, Chapter 3].

□

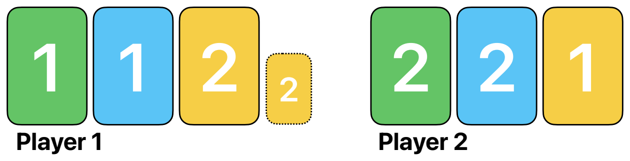
Orbits are the equivalence classes of unique games, G in this case is the group of permutations and X is the set of all possible games (without filtering for symmetries). A game is invariant under a permutation if the division of the (task and playing) cards remains the same after applying the permutation. An example of an invariant game can be found in Figure 3. The key difference between symmetrical games and invariant games is that symmetrical games have equivalent play-outs under a (non-identity) permutation and invariant games are completely identical games.



(a) A (simplified) game instance



(b) The game of 3a is invariant under $(yellow, green)$



(c) The game of 3a is not invariant under $(yellow, blue)$

Figure 3: The game instance shown in 3a is invariant under the operation $(yellow, green)$ (see 3b) as only the order of the cards in hand changes. It is, however, not invariant under $(yellow, blue)$ (see 3c). The task colour is changed and player 1 now holds a yellow 2 instead of a yellow 1 and a blue 1 instead of a blue 2. That is, 3a and 3c are different games within the same orbit. On the other hand, 3a and 3b are the exact same game.

Theorem 3.12 (Total number of 2-player, 1-task games filtering out for all symmetries)**.** *There are* $13{,}696{,}940{,}880$ *unique instances of a 2-player game of The Crew with a single task, when filtering symmetrical games.*

Proof. We prove this theorem by applying Burnside's Lemma. In this case G represents all the symmetries of the game. As mentioned before, these are all the $4! = 24$ permutations of the card suits. We can separate these permutations by the (number of) suits that are swapped.

- 0/1 suits swapped: the only permutation of this type is the identity. All

$$\binom{36}{1} \cdot \binom{36}{18}$$

  games are left invariant under this permutation. In this number, $\binom{36}{1}$ represents choosing a single task card out of 36 possibilities, and $\binom{36}{18}$ represents choosing 18 playing cards for player 1. Choosing cards for player 1 also automatically fixes the cards player 2 has.

- 2 suits swapped: all permutations where two suits are swapped, e.g. $(yellow, blue)$. Since we have 4 suits there are a total of $\binom{4}{2} = 6$ permutations. Games are invariant under these permutations if the task-suit is not changed and player 1 has exactly the same values for both suits (and player 2 as well, but this is fixed by the cards of player 1). The total number of invariant games for each symmetry of this form is

$$\sum_{i=0}^{9} \binom{18}{1} \cdot \binom{9}{i} \cdot \binom{18}{18-2i}.$$

  Here $\binom{18}{1}$ represents choosing a task. Note that this task cannot be of a suit that gets swapped. When choosing cards for player 1 we may pick any number of cards from the two suits that get swapped, as long as player 1 has the same values in both suits. This is represented by term $\binom{9}{i}$. After selecting cards from the suits that get swapped, we have $18 - 2i$ cards left to pick from the non-swapped suits. There are $\binom{18}{18-2i}$ possibilities for this.

- 3 suits swapped: all permutations where three suits are swapped, e.g. $(yellow, blue, green)$ or $(yellow, green, blue)$. Since we have 4 suits and 2 distinct permutations for each group of 3 there are a total of $\binom{4}{3} \cdot 2 = 8$ such permutations. Games are invariant under these permutations if the task-suit is not changed and player 1 has exactly the same values for all three swapped suits (and player 2 as well, but this is fixed by the cards of player 1). The total number of games for each symmetry of this form is

$$\sum_{i=3}^{6} \binom{9}{1} \cdot \binom{9}{i} \cdot \binom{9}{18-3i}.$$

Here, the term $\binom{9}{1}$ represents choosing the task (which must be of the non-permuted suit), $\binom{9}{i}$ represents choosing cards of the 3 permuted suits for player 1 (who must once again have the same values in each suit) and $\binom{9}{18-3i}$ represents choosing $18-3i$ remaining cards from the non-permuted suit. Note that the maximum number of cards player 1 may hold of the fourth (non-permuted) suit is 9. This means that in invariant games of this kind player 1 must hold at least 3 cards of each of the permuted suits. Moreover, he only has 18 cards total. Since he holds an equal number of all the permuted suits, this means that he can hold at most 6 cards of each of the permuted suits.

- 4 suits swapped: Permutations of all 4 suits can be of two different types, i.e. a rotation over all 4 suits $(pink, yellow, green, blue)$ or two swaps of suits $(pink, yellow)(green, blue)$. The number of permutations for the rotation is $3 \cdot 2 = 6$. The first suit is arbitrary as only the order matters, then the second suit has 3 options and the third suit has 2 options, after which the final suit of the rotation is also fixed. For the double swap there are 3 options. Again, the first suit you pick is arbitrary and then there are 3 options for suits to swap it with. Then, based on the first duo picked, the second duo is fixed. The total number of permutations including all 4 suits is therefore 9. There are no invariant games with only a single task under any of these permutations. This is due to the fact that the task suit always changes, leading to a different element within the orbit.

We conclude that we have accounted for all $1 + 6 + 8 + 9 = 24 = 4!$ symmetries. Moreover, the total number of invariant games is equal to

$$\binom{36}{1} \cdot \binom{36}{18} + 6 \cdot \left(\sum_{i=0}^{9} \binom{18}{1} \cdot \binom{9}{i} \cdot \binom{18}{18-2i} \right) + \\ 8 \cdot \left(\sum_{i=3}^{6} \binom{9}{1} \cdot \binom{9}{i} \cdot \binom{9}{18-3i} \right) + 9 \cdot 0 = \\ 326{,}704{,}870{,}800 + 1{,}551{,}835{,}152 + 469{,}875{,}168 = 328{,}726{,}581{,}120$$

Using Burnside's Lemma we conclude that the total number of unique games is equal to

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g| = \frac{1}{24} \cdot 328{,}726{,}581{,}120 = 13{,}696{,}940{,}880\ .$$

□

In Section 3.3, we will consider different types of winnable games. Filtering for all symmetries in that analysis would be complex. We therefore recalculate the total number of games for analysis purposes. In this case we only filter on symmetries in the task card, since this is easy to account for.

Lemma 3.13 (Total number of 2-player, 1-task games filtering for task-suit symmetry)**.** *There are*

$$\binom{9}{1} \cdot \binom{36}{18} = 81{,}676{,}217{,}700$$

unique instances of a 2-player game of The Crew with a single task, when filtering out symmetry in task colour.

Proof. When fixing the task colour we have 9 possible tasks from which we must choose 1. Then, when we pick 18 cards from the 36 available playing cards for player 1, the cards of player 2 are also determined. □

3.3 Winnability of 2-player games with only 1 task

We now analyse the winnability of the game. We do this by splitting the game into two possible situations: one where the critical card is in player 1's hand and one where the critical card is in player 2's hand. We cover these two cases in Theorem 3.15 and Theorem 3.18 respectively.

For the case where the critical card is in the hand of player 1, the task can be completed in two different ways. The first method is by having player 1 play the critical card and player 2 play a card of the critical suit and lower value. The second method is by having player 1 play the critical card and player 2 play a card of a different suit (and player 1 opening the trick). The second method is only possible when player 2 has no (more) cards of the critical suit. To see whether this is possible, we determine how many cards player 2 can discard in a suit.

Lemma 3.14 (Discard opportunities for a given suit using a given suit)**.** *The maximum number of cards of a single other suit that can be discarded by player 2 due to player 1 playing cards of suit i is equal to*

$$T(k_i) = \max\{0, 2k_i - 9\}, \tag{1}$$

where k_i is the number of cards of suit i in player 1's hand.

Proof. If player 1 has k_i cards of suit i, then player 2 has $9 - k_i$ cards. If $9 - k_i > k_i$ (equivalently $k_i < 5$ since $k_i \in \mathbb{Z}$) then player 1 cannot open any tricks in colour i where player 2 does not also play colour i. Therefore, in this case $T(k_i) = 0$. Now assume that $k_i \geq 5$. The difference in cards is $k_i - (9 - k_i) = 2k_i - 9$. In the first tricks both players must play cards of the same suit. Afterwards, if player 1 can open the tricks a total of $2k_i - 9$ tricks can be opened in suit i where player 2 is free to play cards of a chosen different suit.

□

Using this lemma we can determine when games with the critical card in the hand of player 1 are winnable. We split these games in three different cases, which we try to keep maximally disjoint so that we can calculate the exact number of games in each case easier later on.

Theorem 3.15 (Winning 2-player, 1-task games where the task card is in player 1's hand)**.** *A 2-player, 1-task The Crew game, where the critical card is in the hand of player 1 is winnable if and only if (at least) one of the following holds:*

1. *Player 2 has a card that directly loses to the critical card.*
2. *Player 1 has all cards of a suit.*
3. *There is at least one 'directly winning' card $x_w \in P_1$ to a card $y_l \in P_2$ and the cards are distributed according to the formula below (which is equivalent to saying player 2 must get the opportunity to discard all cards he has of the critical suit) for critical colour r and unique suits $a, b, c \in S \setminus \{r\}$.*

$$\begin{cases} 2k_r - 10 + T(k_a) + T(k_b) + T(k_c) \geq 0 \\ \{c \in C \mid c_v \leq t_v \textit{ and } c_s = t_s\} \subseteq P_1 \\ k_r, k_a, k_b, k_c \leq 8 \ . \end{cases} \tag{2}$$

Proof. Note that, within property 3, the rule $\{c \in C \mid c_v \leq t_v \text{ and } c_s = t_s\} \subseteq P_1$ excludes games that satisfy property 1 (since when player 1 is missing a card of the critical suit with a lower value than t property 1. holds) and that $k_r, k_a, k_b, k_c \leq 8$ excludes games that satisfy property 2, to maintain disjointness between the properties. The purpose of these requirements is purely to maintain disjointness, for the ease of later theorems, particularly Theorem 3.17.

We need to show that any game with any of the above properties is winnable and any game without the properties is unwinnable. We begin by proving that each of the cases is winnable.

1. Let y_l be a card in player 2's hand that directly loses to the critical card. The following playout is well-defined and winning: $G := ((t, y_l, 1))$.
2. We separate the cases where player 1 has all cards of the critical suit or all cards of a non-critical suit. If player 1 has all cards of the critical suit the following playout is well-defined and winning: $G := ((t, y, 1))$ for an $y \in P_2$. Now assume that player 1 has all cards of a non-critical suit, say suit $a \neq r$. Let $P_{1,a} = \{x_{a,1}, \ldots, x_{a,9}\}$ be all cards player 1 has of suit a, $P_{2,r} = \{y_{r,1}, \ldots, y_{r,j}\}$ for some j with $1 \leq j \leq 8$ be all the cards player 2 has of suit r and let $y \in P_2 \setminus P_{2,r}$. The following playout is well-defined and winning: $G := ((x_{a,1}, y_{r,1}, 1), \ldots, (x_{a,j}, y_{r,j}, 1)), (t, y, 1))$.
   That is, player 1 first plays as many cards of suit a as player 2 has cards of the critical suit. This gives player 2 the opportunity to discard all his cards of the critical suit (which clearly cannot be strictly more than the 9 cards player 1 has of suit a). Then player 1 is free to play the critical card and win, since player 2 has no more cards of the critical suit.
3. We first show that once the card distribution satisfies Formula (2), player 2 has the opportunity to discard all of his $9 - k_r$ cards of the critical suit.
   By definition, the players can use the suits $a,\ b,$ and c to let player 2 discard $T(k_a) + T(k_b) + T(k_c)$ cards. Playing tricks in the critical suit allows player 2 to discard an additional $k_r - 1$ cards. This is because player 1 is able to play that many tricks in the critical suit without being forced to play the critical card. Thus player 2 also gets to play $k_r - 1$ cards of the critical suit. From Formula (2)

we can directly conclude that the following holds

$$\begin{aligned} & 2k_r - 10 + T(k_a) + T(k_b) + T(k_c) \geq 0 \\ \iff & k_r - 1 + T(k_a) + T(k_b) + T(k_c) \geq 9 - k_r . \end{aligned}$$

We can therefore conclude that player 2 has the opportunity to discard all of his $9 - k_r$ cards of the critical suit.

After all this introductory work, we can finally define a winning playout for the third case, implying that the game is winnable. Let $P'_{1,s} := \{c \in P_1 \mid c_s = s,\ c \neq x_w,\ c \neq t\} = \{x'_{s,1}, \ldots, x'_{s,|P'_{1,s}|}\}$, i.e. all cards player 1 has of suit $s \in S$ excluding the directly winning card and the task card. Similarly, let $P'_{2,s} := \{c \in P_2 \mid c_s = s,\ c \neq y_l\} = \{y'_{s,1}, \ldots, y'_{s,|P'_{2,s}|}\}$ be all card player 2 has of suit $s \in S$ excluding the directly losing card. Let $m_s := \min\{|P'_{1,s}|,\ |P'_{2,s}|\}$, be the minimum of the sizes of these two sets.

Lastly, let $P_1^* = \bigcup_{s \in \{a,b,c\}} \{x'_{s,i} \in P'_{1,s} \mid i > |P'_{2,s}|\} = \{x_1, ..., x_\mu\}$, the 'leftover' cards of the (non-critical) suits for which player 1 has a majority.
Likewise, let $P_2^* = \bigcup_{s \in \{a,b,c\}} \{y'_{s,i} \in P'_{2,s} \mid i > |P'_{1,s}|\} = \{y_1, ..., y_\nu\}$, the cards of non-critical suits player 2 has left. Note that $\mu = T(k_a) + T(k_b) + T(k_c)$ (meaning $\mu + k_r - 1 \geq 9 - k_r$).

The following playout is well-defined and winning:

$$\begin{aligned} G := (& (x'_{a,1}, y'_{a,1}, w_1), \ldots, (x'_{a,m_a}, y'_{a,m_a}, w_i), \\ & (x'_{b,1}, y'_{b,1}, w_{i+1}), \ldots, (x'_{b,m_b}, y'_{b,m_b}, w_j), \\ & (x'_{c,1}, y'_{c,1}, w_{j+1}), \ldots, (x'_{c,m_c}, y'_{c,m_c}, w_k), \\ & (x'_{r,1}, y'_{r,1}, w_{k+1}), \ldots, (x'_{r,m_r}, y'_{r,m_r}, w_l), \\ (x_w, y_l, 1), (x_1, y'_{r,m_r+1}, 1), \ldots, & (x_{|P'_{2,r}|}, y'_{r,|P'_{2,r}|}, 1)), (t, y_1, 1)) . \end{aligned}$$

That is: we first play the number of 'overlapping' cards of each non-critical suit and overlapping cards in the critical suit (excluding the critical card and a directly winning/losing combination of cards). Then we make sure player 1 is the trick opener by playing the directly winning card. From that point on, we know that player 1 will win each trick not started in the critical colour, since player 2 has no more cards in any non-critical suit where player 1 still has cards. Moreover, because the number of tricks played in the critical colour equals $\min(k_r - 1, 9 - k_r)$, player 2 has $\max(10 - 2k_r, 0)$ cards of the critical suit left. Player 1 can then play any card in his hand (except the task) to allow player 2 $\mu = T(k_a) + T(k_b) + T(k_c)$ opportunities to discard his remaining cards of the critical suit. This (as shown before) is sufficient for letting player 2 discard all his cards of the critical suit. Lastly, the task card is played and, since he has no more cards of the critical suit, player 2 may play any card and lose the trick.

To conclude our proof we must show that all games that do not satisfy the three given properties, are losing. Suppose, there is a game where none of the properties hold and that is winnable. Then the final/winning trick of the game must be started by player 1 using the critical card and player 2 must play a card of a different suit, otherwise property 1 would hold. This cannot be on the first turn, since then player 1 would have all cards of the critical suit and property 2 would hold. Therefore, player 2 has discarded all cards of the critical suit before the final trick and it is player 1's turn during the final trick.

This means that player 1 must have won at least one trick using a card that is not the critical card. If the first trick is won by player 1, then this must be using a directly winning card (since, again, property 2 does not hold). If the first trick is won by player 2, then player 1 must win a later trick, otherwise he cannot start the final trick. Winning a trick when the other player opens is only possible using a directly winning card. We can conclude that in both cases player 1 has at least one directly winning card over a card of player 2.
Moreover, player 2 discarding all his $9 - k_r$ cards of the critical suit is only possible, if the cards are distributed in a way that satisfies $9 - k_r \leq T(k_a) + T(k_b) + T(k_c) + k_r - 1$, i.e.,
$2k_r - 10 + T(k_a) + T(k_b) + T(k_c) \geq 0$.
Finally, note that the second and third requirements of Formula (2) are satisfied since property 1 and 2 do not hold. Therefore, in this case, all requirements of property 3 are satisfied.

We conclude that a game is winnable only when at least one of the given properties holds. We can therefore conclude that games without any of the properties must all be losing.

□

Corollary 3.16 (Losing 2-player, 1-task games where the task card is in player 1's hand)**.** *A 2-player, 1-task The Crew game, where the critical card is in the hand of player 1 is losing if and only if all of the following properties hold:*

1. *Player 2 has no cards that lose directly to the critical card.*
2. *Player 1 does not have all cards of any suit.*
3. *There is no directly winning card in the hand of player 1 over player 2 and/or the cards are distributed according to the following formula for critical colour r and unique suits $a, b, c \in S \setminus \{r\}$:*

$$\begin{cases} 2k_r - 10 + T(k_a) + T(k_b) + T(k_c) < 0 \\ \{c \in C \mid c_v \leq t_v \text{ and } c_s = t_s\} \subseteq P_1 \\ k_r, k_a, k_b, k_c \leq 8\ . \end{cases} \tag{3}$$

Proof. This is the complement of Theorem 3.15, with the exception that the two final requirements of Formula (2) and Formula (3) overlap. These requirements correspond with maintaining disjoint properties in Theorem 3.15. In this case they overlap with properties 1 and 2 and are redundant, but included to maintain similarity.

□

Theorem 3.17. *The number of losing 2-player, 1-task The Crew games with the critical card in the hand of player 1 for critical suit r and non-critical unique suits $a, b, c \in S$ is equal to $S_1 + S_2$, where*

$$S_1 = \sum_{t_v=1}^{8} \sum_{k_r=t_v}^{8} \sum_{k_a=0}^{8} \sum_{k_b=0}^{8} \sum_{k_c=0}^{8} 1_{f_1(k_r,k_a,k_b,k_c)} \cdot 1_{k_r+k_a+k_b+k_c=18}, \tag{4}$$

with f_1 referring to satisfying Formula (2), and

$$S_2 = \sum_{t_v=1}^{3} \sum_{k_r=t_v}^{3} \sum_{k_a=5}^{7} \sum_{k_b=5}^{7} \sum_{k_c=5}^{7} \binom{9-t_v}{k_r-t_v} \binom{9}{k_a} \binom{9}{k_b} \binom{9}{k_c} \cdot 1_{f2(k_r,k_a,k_b,k_c)} \cdot 1_{k_r+k_a+k_b+k_c=18}, \qquad (5)$$

with f_2 referring to satisfying Formula (3).

Proof. Properties 1 and 2 of Corollary 3.16 must always hold and 3 can be split into two cases. Either there is no directly winning card in the hand of player 1 or the cards are distributed in accordance with Formula (3). We must ensure that the games that have both of these options for property 3 are only counted once.

To find the total number of games, we will first determine the number of games satisfying properties 1 and 2, where there is no directly winning card in the hand of player 1 over player 2, but excluding initial games that satisfy Formula (3). After that, we will determine the number of games satisfying properties 1 and 2 that do satisfy Formula (3) and where there might also be a directly winning card in the hand of player 1 over player 2. Note that (only) the second case includes games which satisfy both parts of property 3.

Property 2 is enforced by setting the maximum number of cards of each suit to 8. We take property 1 into account by requiring that player 1 has all cards of the critical suit of lower value than the critical card. Otherwise the critical card wins directly over a card in the hand of player 2. We ensure this by setting the lower bound in $\sum_{k_r=t_v}^{3}$ to t_v. In the second case, we then have $k_r - t_v$ other cards of the critical suit and $9 - t_v$ cards left to choose from, leading to the factor $\binom{9-t_v}{k_r-t_v}$. Since the maximum number of cards of a suit is 8, this also means that the task-value cannot be greater than 8 in a losing game.

For the first case (given in Formula (4)), player 1 must always have the cards from the lowest value counting upwards for as many cards of the suit he has in total. After all, if he skips a value then there is a directly losing card in the hand of player 2. Since we are excluding games where the cards are distributed in accordance with Formula (3), we include the requirement that the cards are distributed according to Formula (2) instead.

For the second case (given in Formula (5)), note that if a task has value 4 or higher Formula (3) can never be satisfied. After all, $t_v \geq 4$ also implies that $k_r \geq 4$ due to property 1. When k_r satisfies $k_r \geq 5$ it already holds that $2k_r - 10 \geq 0$. If, on the other hand, $k_r = 4$ it holds that $2k_r - 10 = -2$, but either two of k_a, k_b, k_c will be at least 5 or at least one will be at least 6.
In both cases $2k_r - 10 + T(k_a) + T(k_b) + T(k_c) \geq -2 + 2 = 0$ holds. Because $k_r \leq 3$, it must also hold that $t_v \leq 3$. Due to this we can set the upper bound of the task-value and k_r in $\sum_{t_v=1}^{3}$ and $\sum_{k_r=t_v}^{3}$ to 3.

With this upper bound for the task values and k_r we can also determine upper and lower bounds for the other suits. If player 1 were to have 8 cards of a non-critical suit, say a, we know that $k_r = t_v = 1$, since otherwise $2k_r - 10 + T(k_a) + T(k_b) + T(k_c) \geq -6 + T(8) = 1 \geq 0$ would already hold. However, in this case, either k_b or k_c will have to be at least 5 and $2k_r - 10 + T(k_a) + T(k_b) + T(k_c) \geq -8 + T(8) + T(5) = 0$. So the upper bound for k_a, k_b, k_c becomes 7 to ensure Formula (3) can be satisfied.

If player 1 were to have 4 cards (or fewer) of a non-critical suit, say a, then $T(k_a) = 0$. Thus, $2k_r - 10 + T(k_a) + T(k_b) + T(k_c) = 2k_r - 10 + T(k_b) + T(k_c)$ holds. Since $T(k_i) = \max\{0, 2k_i - 9\}$, the minimum of $T(k_b) + T(k_c)$ is 0 when $k_b + k_c = 8$ (since $T(4) = 0$) and increases by 1 for each extra card up to 10 that needs to be of suit b, c (since $T(5) = 1$) and then increases by 2 for each extra card.
If $k_r = 1$ then $k_b + k_c \geq 13$ meaning $T(k_b) + T(k_c) \geq 8 = -2k_r + 10$.
If $k_r = 2$ then $k_b + k_c \geq 12$ meaning $T(k_b) + T(k_c) \geq 6 = -2k_r + 10$.
If $k_r = 3$ then $k_b + k_c \geq 11$ meaning $T(k_b) + T(k_c) \geq 4 = -2k_r + 10$.
Thus, in any such case, $T(k_b) + T(k_c) \geq -2k_r + 10$ and therefore Formula (3) cannot be satisfied. We conclude that the lower bound for k_a, k_b, k_c is 5.

(Note: Within these bounds any combination of k_r, k_a, k_b, k_c satisfies $2k_r - 10 + T(k_a) + T(k_b) + T(k_c) < 0$. To understand why this is true, we first observe that $k_r + k_a + k_b + k_c \geq 1 + 5 + 5 + 5 = 16$, so there are only 2 cards out of 18 left to be distributed over the suits. For $i \in \{a, b, c\}$, $T(k_i)$ is at least 1 since player 1 has at least 1 card more than player 2 in each non-critical suit. Additionally, $T(k_i)$ scales with 2 for each value above 5. The value of k_r is always at least 1 and the term $2k_r$ scales with a factor of 2 for each increase of k_r. Therefore, the 2 cards out of 18 that are 'unaccounted for' always allow for exactly 2 discard opportunities each. Meaning, $2k_r - 10 + T(k_a) + T(k_b) + T(k_c) = 2 - 10 + 1 + 1 + 1 + 2 \cdot 2 = -1 < 0$. This also means that, when using the given lower and upper bounds for k_r, k_a, k_b, k_c and the factor $\binom{9-t_v}{k_r-t_v}$, all three requirements of Formula (3) are accounted for within Formula (5). Thus, using the indicator function $1_{f_2(k_r,k_a,k_b,k_c)}$ is actually redundant in this case, but it is still included for similarity to S_1.)

We conclude that Formula (4) and Formula (5) are well-defined and their sum is equal to the number of 2-player, 1-task losing games of *The Crew* where the critical card is in the hand of player 1.

□

Theorem 3.18 (Winning 2-player, 1-task games where the task card is in player 2's hand)**.** *A 2-player, 1 task The Crew game, where the critical card is in the hand of player 2 is winnable if and only if and only if (at least) one of the following holds:*

1. *Player 1 has a card that directly wins over the critical card.*
2. *Player 1 has all cards of a (non-critical) suit.*
3. *There is at least one 'directly winning' card $x_w \in P_1$ to a card $y_l \in P_2$ and player 1 has a majority of cards (but not all) in any non-critical suit.*

Proof. It is sufficient to show that any game of the above properties is winnable and any game without the above properties is unwinnable. We begin by proving each of the given cases is winnable.

1. Let $x_w \in P_1$ be a directly winning card over t. The following playout is well-defined and winning: $G := ((x_w, t, 1))$.
2. Let $a \in S$ be a (non-critical) suit such that player 1 holds all cards of suit a. Let $x \in P_1$ such that $x_s = a$. Since by definition $t_s \neq a$ and player 2 holds no cards of suit a the following playout is well-defined and winning: $G := ((x, t, 1))$.

3. Say that player 1 has the majority of cards in suit $a \in S$ and $a \neq t_s$ let $P'_{1,a} := \{c \in P_1 \mid c_s = a,\ c \neq x_w\} = \{x'_{a,1}, \dots, x'_{a,\left|P'_{1,a}\right|}\}$ be the set of all cards player 1 has of this suit, excluding the directly winning card (if it is of suit a).
   Likewise, let $P'_{2,a} := \{c \in P_2 \mid c_s = a,\ c \neq y_l\} = \{y'_{a,1}, \dots, y'_{a,\left|P'_{2,a}\right|}\}$ be the set of all cards player 2 has of this suit, excluding the directly losing card (if it is of suit a).
   Let $m_a := \min\{\left|P'_{1,a}\right|, \left|P'_{2,a}\right|\} = \left|P'_{2,a}\right|$. The following playout is well-defined and winning:

$$G := ((x'_{a,1}, y'_{a,1}, w_1), (x'_{a,2}, y'_{a,2}, w_2), \dots, (x'_{a,m_a}, y'_{a,m_a}, w_{m_a}), (x_w, y_l, 1), (x'_{a,m_a+1}, t, 1))$$

   That is, we first play the overlapping cards of suit a (excluding x_w/y_l). Then we play x_w/y_l such that player 1 can open the trick. Player 1 can open the trick in suit a and since player 2 has no more cards of this suit, he is free to play the critical card and win the game.

To conclude our proof we must show that all games that do not satisfy the 3 given properties are losing. Now imagine a game where none of the properties hold and say this game is winnable. The final/winning trick of a winning playout of the game must be started by player 1 in a different suit (otherwise property 1 would hold) and player 2 must play the critical card. If it is the first trick and player 2 can play a card of a different suit then that would mean that player 1 has all cards of the suit, but that would imply property 2 holds. Therefore, we assume it is a later trick. But if this is the case and player 1 does not have all the cards of a non-critical suit, then this would imply that player 1 has at least one directly winning card over player 2. Moreover, player 1 must have strictly more cards of a different suit (since player 1 does not have all cards of any suit, but can get a winning trick where player 2 plays a different suit), which would imply that property 3 holds. In other words, any winning game must satisfy at least one of the given properties.

□

Corollary 3.19 (Losing 2-player, 1-task games where the task card is in player 2's hand)**.** *A 2-player, 1-task The Crew game, where the critical card is in the hand of player 2 is losing if and only if all of the following properties hold:*

1. *Player 1 has no cards that directly win over the critical card.*
2. *Player 1 does not have all cards of any (non-critical) suit.*
3. *There is no directly winning card in the hand of player 1 over player 2 and/or player 1 does not have a majority of cards in a non-critical suit.*

Proof. This is the complement of Theorem 3.18. □

Theorem 3.20. *The number of losing 2-player, 1-task The Crew games with the critical card in the hand of player 2 for critical suit r and non-critical unique suits $a, b, c \in S$ is equal to $S_3 + S_4$, where*

$$S_3 = \sum_{t_v=1}^{9} \sum_{k_r=0}^{t_v-1} \sum_{k_a=0}^{8} \sum_{k_b=0}^{8} \sum_{k_c=0}^{8} 1_{k_r+k_a+k_b+k_c=18} \cdot 1_{k_a \geq 5 \,\vee\, k_b \geq 5 \,\vee\, k_c \geq 5} \tag{6}$$

and

$$S_4 = \sum_{t_v=7}^{9} \sum_{k_r=6}^{t_v-1} \sum_{k_a=2}^{4} \sum_{k_b=2}^{4} \sum_{k_c=2}^{4} \binom{t_v-1}{k_r} \cdot \binom{9}{k_a} \cdot \binom{9}{k_b} \cdot \binom{9}{k_c} \cdot 1_{k_r+k_a+k_b+k_c=18} \cdot \tag{7}$$

Proof. Properties 1 and 2 of Corollary 3.19 must always hold and 3 can be split into two cases. Either player 1 has no majority in any non-critical suit or he has no directly winning card over player 2. We must ensure that the game that have both of the options for property 3 are only counted once.

To find the total number of games, we will first determine the number of games satisfying properties 1 and 2, where there is no directly winning card in the hand of player 1 over player 2, but excluding initial games in which player 1 does not have a majority of a non-critical suit. After that, we will determine the number of games satisfying properties 1 and 2 in which player 1 does not have a majority of a non-critical suit and where there might also be a directly winning card in the hand of player 1 over player 2. Note that (only) the second case includes games which satisfy both parts of property 3.

We will first determine the games where properties 1 and 2 do not hold and player 1 has no majority of non-critical cards (including games where there is also no directly winning card in player 1's hand over a card in player 2's hand).

For the first case (given in Formula (6)) player 1 must only hold cards that directly lose to all cards of the same suit in player 2's hand. Hence, if player 1 has k_s cards of suit s, he should have the lowest k_s cards of that suit. Obviously, he cannot hold any cards that directly win over the task card (otherwise property 1 would not be satisfied) and he holds at most 8 cards in any suit (otherwise property 2 would not be satisfied). Moreover, to exclude all cases where there is both not a directly winning card in player 1's hand and no majority of cards, we add the requirement that at least one of the non-critical suits has at least 5 cards.

For the second case (given in Formula (7)) note that player 1 must have no more than 4 cards of any non-critical suit, since 5 is a majority of cards. This also means that he must hold at least 2 cards of each of the other suits. After all, player 1 holds at most 8 cards of the critical suit, which leaves at least 10 cards to be divided over the three non-critical suits. If he has less than 2 cards of any of the non-critical suits, one of the others would have at least 5 cards. This also means that player 1 must always have at least 6 cards of the critical suit (since having 13 cards of the non-critical suits means at least one non-critical suit with 5 cards). On the other hand, player 1 holds at most $t_v - 1$ cards of the critical suit (i.e., only cards lower than the task value). Otherwise, he would have a directly winning card over the task card. This implies that $t_v \geq 7$. Otherwise, there are no requirements for which cards are chosen, since this time it is no problem if player 1 holds one or more directly winning cards.

□

We now have all the necessary formulae to calculate the total number and percentage of losing games.

Corollary 3.21 (The number/percentage of winning/losing 2-player, 1-task *The Crew* games)**.** *The number of losing games for 2-player The Crew with a single task is equal to*

$$\begin{aligned}&S_1 + S_2 + S_3 + S_4 = \\ &2037 + 112{,}402{,}080 + 2037 + 112{,}402{,}080 = 224{,}808{,}234\ .\end{aligned}$$

This means that the total losing percentage of the game is

$$\frac{224{,}808{,}234}{81{,}676{,}217{,}700} \approx 0.28\%\ .$$

We conclude that the number of winnable games of this version of The Crew is equal to $81{,}451{,}409{,}466$ *or* 99.72% *of all games.*

Proof. This can be easily verified using a brute-force calculation of Theorem 3.17 and Theorem 3.20 and combining the result with Lemma 3.13. □

3.4 The relation between the two types of 2-player, 1-task games

A surprising result is that the two different types of games described in of Section 3.3 have the exact same number of losing games. This equality leads us to the question if there is a similarity between the two sets. In this section, we give an explanation of the relation between the two subsets of these game types that we counted as S_2 and S_4 in Theorem 3.17 and Theorem 3.20 respectively. The sums S_2 and S_4 can be rewritten into each other using the symmetry of binomial coefficients, but this does not give an intuition of the underlying relation between the sets. In this section, to provide an intuition behind the relation between these two subsets, we define a bijection between the two sets. Since both sets are finite, this once again proves they are of the same size. At the end of this section, we provide an argument as to why any possible relation between the subsets counted in S_1 and S_3 is more difficult to define.

To define the bijection we first define the complement of a card.

Definition 3.22 (The complement of cards)**.** *Given a card* $x := (s, v)$*, we define its complement as* $x^c := (s, 9 - v + 1)$*.*

In other words, a complement of a card is a card of the same suit but the 'opposite' value. For example, the complement of a green 8 is a green 2 (and vice versa).

Theorem 3.23 (Bijection between two subsets of losing 2-player, 1-task games)**.** *The mapping that changes a game by*

1. *Taking the complement of each card of the critical suit and the task card,*
2. *Swapping the playing cards of both players,*

is a bijection that maps the set of losing games as described in Corollary 3.16 that satisfy Formula (3) to the set of losing games as described in Corollary 3.19 where player 1 does not have a majority of cards in any non-critical suit.

Proof. It is easy to see that the mapping is its own inverse mapping and that the images of the mapping are valid game instances. Furthermore, the images of the mapping are different if the inputs are different, i.e. the mapping is injective. All that remains to be proven is that the given domain and codomain are correct. That is, let L_1 be the losing games as described in Corollary 3.16 that satisfy Formula (3) and let L_2 be the losing games as described in Corollary 3.19 where player 1 does not have a majority of cards in any non-critical suit, we want to show that the given mapping maps L_1 to L_2 (and vice versa).

We begin by showing that all games of L_1 get mapped to games of L_2, by showing that the images of the games must satisfy properties 1 and 2 as described in Corollary 3.19 and satisfy property 3 by player 1 not having any majority of cards in non-critical suits.
Take $l \in L_1$ and let l' denote the image of game l under the given mapping. We know that, in game l,

player 2 has no cards that directly lose to the critical card. We also know that player 2 must have at least one card in the critical suit. Note that for the cards x, y where x is directly winning over y, x^c is directly losing to y^c. Likewise, when taking the complement over all of the cards of the critical suit in a game, player 2 now has no cards that directly win over the critical card. After swapping hands, player 1 has no cards that directly win over the critical card, thus the first property of Corollary 3.19 is satisfied in l'.
From (the proof of) Theorem 3.17 we know that in l, player 1 cannot have 0 cards in any suit. Equivalently, player 2 cannot have all cards of any suit in l. This means that after applying the mapping, player 1 does not have all cards in any suit, thus the second property of Corollary 3.19 is satisfied in l'.
Lastly, in the proof of Theorem 3.17 we showed that player 1 must have at least 5 cards of each non-critical suit. Therefore, after applying the mapping, player 1 has at most 4 cards of each non-critical suit. Thus, in l' player 1 has no majority in any critical suit and the required part of the third property of Corollary 3.19 is also satisfied.
We conclude that l' is a game in set L_2.

We now show that all games of L_2 get mapped to games of L_1, by showing that the images of the games must satisfy properties 1 and 2 as described in Corollary 3.16 and satisfy property 3 by satisfying Formula (3).
Take $l \in L_2$ and let l' denote the image of game l under the given mapping. We know that, in game l, player 1 has no cards that directly win over the critical card. We also know that player 1 must have cards in the critical suit, otherwise (as shown in the proof of Theorem 3.20) player 1 cannot have a minority of cards in all non-critical suits. When taking the complement over all of the cards of the critical suit in a game, all of the cards of the critical suit of player 1 now win over the critical card. After swapping hands, player 2 has no cards that directly lose to the critical card, thus the first property of Corollary 3.16 is satisfied in l'.
From (the proof of) Theorem 3.20 we know that in l, player 1 cannot have less than 2 cards in any suit. Equivalently, player 2 has no more than 7 cards in any given suit. Therefore player 2 cannot have all cards of any suit in l. This means that after applying the mapping, player 1 does not have all cards in any suit, thus the second property of Corollary 3.16 is satisfied in l'.
Lastly, in the proof of Theorem 3.20 we showed that player 1 must have between 6 and 8 cards of the critical suit and between 2 and 4 cards of each non-critical suit. After applying our mapping, player 1 therefore ends ups with between 1 and 3 cards the critical suit and between 5 and 7 cards for each non-critical suit. As noted in our proof of Theorem 3.17, these bounds correspond with games that satisfy the requirements of Formula (3). Therefore the required part of third property of Corollary 3.16 is also satisfied.
We conclude that l' is a game in set L_1.

In conclusion, the described map is a bijection that maps the set of losing games as described in Corollary 3.16 that satisfy Formula (3) to the set of losing games as described in Corollary 3.19 where player 1 does not have a majority of cards in any non-critical suit.

$\square$

It is unclear whether such a relation also exists between the smaller sets. If there is one, it is not as obvious. A difficulty with this particular case is that the smaller set of Corollary 3.19 contains games where player 1 has no cards of the critical suit. In all games described in Corollary 3.16 both players must always have cards of the critical suit. Additionally, in both cases, player 2 may have all cards of a non-critical suit and player 1 may not. A mapping that includes swapping the players' hands, as was done in Theorem 3.23, would not be possible. Furthermore, games of the smaller set of Corollary 3.19 exist with all possible task values, whereas games in the smaller set of Corollary 3.16 cannot have tasks with value 9. These differences make it very unintuitive to find a mapping between the two sets.

4 Algorithms

In this section we will compare several strategies (agents) for playing a game of *The Crew*. We separate the agents based on the two phases (task-taking and playing) of the game. Moreover, we make the distinction between simple 'rule-based' agents and more complex agents based on Monte Carlo methods. The agents from both phases can be combined into players for the full game. We combine and evaluate the agents on their ability to win 3-player games in Section 5.

4.1 Rule-based agents

To help identify suitable strategies for *The Crew* we first define four rule-based algorithms, whose performance we will compare later. We define two algorithms for the task-taking phase and two for the playing phase of the game. Within these algorithms each card c is a pair $c := (x, y)$, where $c_s := x$ is the suit of the card and $c_v := y$ is the value of the card (as was the case in Section 3). We begin by defining the two task-taking agents, see Algorithm 1 and Algorithm 2.

Algorithm 1 The Random task-taking Player

```
$T$ unchosen tasks
$t$ task to-be-taken

$t \leftarrow$ random card from $T$
Take($t$)
```

Algorithm 2 The Tactical task-taking Player

```
$T = \{t_1, \ldots t_n\}$ unchosen tasks
$P$ cards of current player
$t$ task to-be-taken

for $t_i \in T$ do
    calculate $x_i := Score(t_i, P)$
end for
$M_T := \{t_i \in T \mid x_i \geq x_j \; \forall j \in \{1, 2, ..., n\}$ and $i \neq j\}$
$t \leftarrow$ random task from $M_T$
Take($t$)
```

In Algorithm 2,

$$Score(x, P) = \begin{cases} (x_v - 5) \cdot 2 + |P_{x_s}| & x \in P \\ |P_{>x}| & x \notin P \text{ and } |P_{>x}| > 0 \\ -|P_{x_s}| & x \notin P \text{ and } |P_{>x_v}| = 0, \end{cases} \tag{8}$$

where we define

$$P_{>x} = \{c \in P \mid c_s = x_s \text{ and } c_v > x_v\} \tag{9}$$

$$P_{x_s} = \{c \in P \mid c_s = x_s\}. \tag{10}$$

The first case of the scoring formula represents the situation where the player picking the task has the associated playing card in hand. First, cards are evaluated on their value. It is intuitive that a player wants to take tasks associated with high value cards in his hands and avoid tasks associated with low value cards in his hand. For example, say a player has both the green 9 task card and the green 9 playing card. He is guaranteed to have the ability to complete this task if there is any trick opened with a green card. On the other hand, it is difficult for a player to complete a green 1 task if he has the green 1 playing card on hand. After all, this is only possible, if he can start a trick when no other players have any green cards. The scoring is scaled over all cases in such a manner that a task for a card with value 1 in hand is the lowest possible base score (-8). Likewise, a task for a card with value 9 in hand gives the highest possible base score (8). Note that if a given player has more cards in a given suit, it becomes easier for all other players to discard their cards of that suit. This is due to the facts that 1. the other players will have fewer cards of the suit to discard and 2. more tricks can safely be opened in that suit with the purpose of others discarding more cards of that suit without the player with the task being forced to play the critical card. When a player has both a task and the corresponding critical card, allowing the other players to discard all their directly winning cards is crucial. We capture this intuition using the term $+|P_{x_s}|$. This gives a bonus for each card you have in the given colour (note that this is always at least (+1)).

The second case represents the case where a player does not have the critical card in hand, but does have at least one directly winning card over the critical card. Intuitively, the more cards a player has that directly win over the critical card, the easier it becomes to complete the task. Therefore the score of this task is equal to the number of directly winning cards the player has.

The third case represents the situation where a player does not have the critical card in hand and cannot win the critical card directly. Intuitively, completing this task is rather difficult. In this case, the player must win the card by opening a trick in a different suit and the player with the critical card in hand must discard it during this trick (i.e. he must not have any cards of the suit in which the trick was opened). This case requires you to have more cards of the non-critical suits, so it is disadvantageous to have more cards matching the suit of the task-card. Therefore the score of this task is equal to minus the number of cards the player has of the critical suit.

Note that this scoring-function is not intended to be optimal (and most likely is not), but simply an improvement over picking tasks randomly. After all, picking tasks randomly appears to be a big factor in limiting the winnability of a game of *The Crew* [dJ21].
Improvements might still be made, for example, by making the scoring function non-linear (winning a task associated with a card in hand with value 9 might be exponentially easier than with a 6/7/8). Alternatively, in the second case you might also want to change the scaling value or take into account how much higher the values are compared to the task (a card with value 1 is much easier when you have a 9 of the same suit than when you only have a 2 of that suit). In the third case you might also want to consider how many cards you have in the different suits. For example, having all cards of a suit as the initial starting player allows you to win any card you do not have in hand. In this case, when you start a trick in this suit you are guaranteed to win the trick and all other players may play any of their hand cards.

Next, we define agents for the playing phase of the game, see Algorithm 3 and Algorithm 4.

Algorithm 3 The Random Player

```
$P$ cards of current player
$p$ card to-be-played
$s$ required suit

if $s$ is not-defined then                              ▷ First card to be played
    $p \leftarrow$ random card from $P$
    $s \leftarrow p_s$
else                                                    ▷ Play card of requested suit if possible
    $P_s := \{ c \in P \mid c_s = s \}$
    if $P_s \neq \emptyset$ then
        $p \leftarrow$ random card from $P_s$
    else
        $p \leftarrow$ random card from $P$
    end if
end if
Play($p$)
```

For a simple improvement upon the random player, we can ensure that the player will play a card (of the required suit) with a high value if there is a card in play he has a task card for (and he has cards of the required suit). Likewise, we can ensure that a player will play a card with low value if there is a card in play that a different player needs to win (and he has cards of the required suit). Random(P) in Algorithm 4 refers to applying Algorithm 3. $\max_v/\min_v$ is taking the card with maximum/minimum value.

Algorithm 4 The RandomSmart Player

```
$T_i$ tasks of player $i$
$x$ current player
$P$ cards of current player
$R$ cards in play in current trick
$p$ card to-be-played
$s$ requested suit

if $R$ is empty then                          ▷ No cards played yet: play random card
    $p \leftarrow \text{Random}(P)$
    $s \leftarrow p_s$
else if $R \cap T_x \neq \emptyset$ then      ▷ Card in play this player wants: play highest value card of suit
    $P_s := \{c \in P \mid c_s = s\}$
    if $P_s \neq \emptyset$ then
        $p \leftarrow \max_v(P_s)$
    else
        $p \leftarrow \text{Random}(P)$
    end if
else if $R \cap T_i \neq \emptyset$ for some $i \neq x$ then   ▷ Card another player wants: play lowest value card of suit
    $P_s := \{c \in P \mid c_s = s\}$
    if $P_s \neq \emptyset$ then
        $p \leftarrow \min_v\{P_s\}$
    else
        $p \leftarrow \text{Random}(P)$
    end if
else                                          ▷ Base case: play random card
    $p \leftarrow \text{Random}(P)$
end if
Play($p$)
```

As said, the player in Algorithm 4 aims to play his highest card (of the required suit) if a card is in play he wants, and his lowest card if another player wants a card in play. It is possible to have both cases at once, at which point the game is guaranteed to be lost. The player will still opt to play the highest card of the required suit if possible rather than the lowest card of the required suit, but in practice it does not matter, since the game is already lost. Note, that the highest card might not be high enough or the lowest card might not be low enough. In these cases the game is also already lost and the player can do nothing to prevent it.

4.2 Monte Carlo based agents

To consider more advanced strategies for playing *The Crew*, we define Monte Carlo based agents. We begin by defining Monte Carlo Tree Search and a method often used in combination with Monte Carlo Tree Search, called the Upper Confidence bound applied to Trees. After that, we will detail two implementations of Monte Carlo agents, which can be used for both phases of the game.

4.2.1 Monte Carlo Tree Search

In Monte Carlo Tree Search (MCTS) the search-space is expanded by several play-outs. Within MCTS we consider a search tree, where each of the nodes represents a game-state. The children of each node are the game-states that can be reached by doing a valid move from the original node. The root R of the search tree is the current game-state. The leaf nodes of the tree are any nodes from which no play-out has been initiated. Each round of MCTS consists of four steps (see [CWH+08], [CF10]):

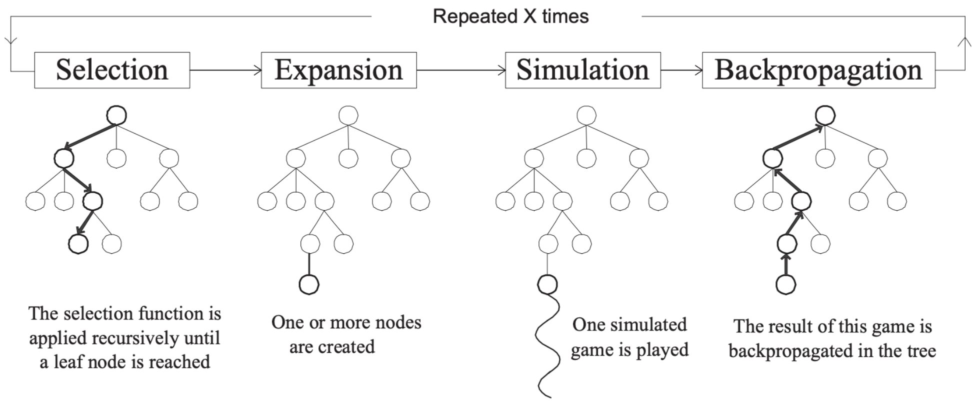
- *Selection*: Starting from the root node, children are repeatedly chosen based on a selection criterion, until a leaf node L is reached. This leaf node will be expanded in the following steps. Typically, the selection criteria prioritises nodes with either high uncertainty (i.e. there have been relatively few play-outs from a node, so not a lot is known about it) and/or high valuation (i.e. the node seems relatively promising so far).

- *Expansion*: This step is optional. Expand the tree (from leaf-node L) by adding one or more child (leaf-)nodes to the tree. Typically, at least the first move of the play-out of the *Simulation* step is included. We will elaborate on our expansion method in Section 4.2.3.

- *Simulation*: Complete one or more play-outs from node C, one of the children of L (or L itself if no children were added). Commonly, Monte Carlo play-outs use random moves, but it is also possible to do play-outs based on heuristics.

- *Backpropagation*: The result of the play-out is used to update all nodes on the path from C back to the root node.

Rounds are repeated until a certain threshold is reached. Typically, this is either a maximum number of rounds being completed or a fixed amount of time running out. Afterwards, a move is chosen based on the information of the direct children of the root node. We will refer to this phase of MCTS as the *conclusion* phase.



Repeated X times
Selection
Expansion
Simulation
Backpropagation
The selection function is applied recursively until a leaf node is reached
One or more nodes are created
One simulated game is played
The result of this game is backpropagated in the tree

Figure 4: Example of the four MCTS phases, taken from [CWH+08].

4.2.2 Upper Confidence bound applied to Trees

A common method for node selection in the *Selection* phase (and one we will use for one of the Monte Carlo agents) is Upper Confidence bound applied to Trees (UCT) [KS06]. This method balances prioritising nodes with high uncertainty and high valuation as follows (notation is based on [CF10]):

$$U_i = \frac{v_i}{n_i} + c\sqrt{\frac{\ln(N)}{n_i}}, \quad \text{with}$$

$$\begin{aligned} v_i &: \text{the value/score of child node } i, \\ c &: \text{a constant that values expanding nodes with high valuation if low, and high uncertainty if high,} \\ N &: \text{The number of times the parent node has been visited,} \\ n_i &: \text{The number of times the child node } i \text{ has been visited.} \end{aligned}$$

The child with the highest U_i value will be chosen until a leaf node is reached. In our implementation we will consider the value of U_i to be ∞ whenever $n_i = 0$.
c is a value that can be fine-tuned to the specific use-case, but is often set to the theoretical value $\sqrt{2}$ (see [SM22]). This is also the value we will use.

4.2.3 Applied implementations

We implemented two distinct versions of MCTS: one that selects randomly in the selection phase and one that selects based on UCT. Moreover, we have two different scoring functions that can be applied to both of the methods. In both cases the Monte Carlo agents make use of complete information of the game state, that is: information is used about the hands of all players.

At first only a basic MCTS function that evaluates winning games with $+1$ was considered. However, this method seemed insufficient, when considering high numbers of tasks. Based on prior research done by [dJ21] there is reason to believe that games with a higher number of tasks should also be winnable in many cases. This discrepancy might be due to it being difficult to find any winning games during the first (possibly most critical) moves. This is caused by increasingly larger search spaces and increasingly lower fraction of winning playouts. Note, that even when a playout is not won, we might still consider a playout that has more tasks completed to be 'closer' to a win than a game where fewer tasks were completed. To account for this intuition we will incorporate the number of tasks won in our update function as follows. Let t be the number of tasks completed in a random playout of node i and let $S(i)$ be the score that will be added to the value of i (v_i) and the values of its ancestors.

$$S(i) = \begin{cases} 100, & \text{random playout won} \\ t, & \text{random playout not won.} \end{cases}$$

The value of 100 was chosen since we still want to weigh a proper win much heavier than a game where only a subset of the tasks was won and the maximum number of tasks we will consider in Section 5 is 20.

We have implemented two different Monte Carlo agents for playing *The Crew*, as specified in Algorithm 5 and Algorithm 6. There are two variants for both agents, one for each of the scoring functions described above.
Both defined Monte Carlo agents can be used in both phases. When the agents are used only in the task phase, the playouts will still continue into the playing phase and nodes for the playing phase will also be added to the search tree. The full playout is needed in order to score the different possible moves from the root node properly.

Algorithm 5 The Pure Monte Carlo Player ('PMC', can be applied to both task and playing phases)

```
This is a Monte Carlo Tree Search agent with the following properties.
Selection: Repeatedly choose a random child until a leaf is reached.
Expansion: All children are added to the tree.
Simulation: Play random moves until the game reaches a conclusion (win/loss).
Backpropagation: Standard scoring: update the score of the leaf and each of its ancestors with $+1$ on
a win. After a loss do nothing.
Smart scoring: update the score of the leaf and each of its ancestors with $+100$ on a win. Update the
score of the leaf and each of its ancestors with $+t$, where $t$ is the number of tasks that were completed
after a loss.
Conclusion: the eventual 'best move' is a (random) child of the root-node that has the highest score.
```

In literature, Pure Monte Carlo is either described as a method where a fixed (equal) number of random playouts are done for each possible move (e.g. [dJ21]) or as a method where each move of the playout is done randomly (e.g. [MC08]). In the used implementation defined by Algorithm 5, the moves are chosen randomly each time, since both the children chosen in the *Selection* phase and the moves done in the *Simulation* phase are random. Since the children of the root node (and all subsequent children) are chosen randomly, the total number of playouts of each direct child will be roughly equal. Therefore, the implementations are practically equivalent regardless.

Algorithm 6 The Monte Carlo Tree Search using Upper Confidence bound applied to Trees Player (‘MCTS-UCT’, can be applied to both task and playing phases)

```
This is a Monte Carlo Tree Search agent with the following properties.
**Selection:** Repeatedly choose the child with highest UCT score.
**Expansion:** All children are added to the tree.
**Simulation:** Play random moves until the game reaches a conclusion (win/loss).
**Backpropagation:** *Standard scoring*: update the score of the leaf and each of its ancestors with $+1$ for a win. After a loss do nothing.
*Smart scoring*: update the score of the leaf and each of its ancestors with $+100$ on a win. Update the score of the leaf and each of its ancestors with $+t$, where $t$ is the number of tasks that were completed after a loss.
When calculating the UCT for node $i$ the value of node $\frac{v_i}{n_i}$ is multiplied by a factor of $\frac{1}{100}$ to correct for the higher score values.

**Conclusion:** the eventual ‘best move’ is a (random) child of the root-node which has the highest score ($v_i$).
```

Note that there are many more variations of MCTS that can be applied to this problem. An example would be evaluating the score of each of the direct children of the root node by also taking into account the number of visits to that node (i.e. divide the score by the number of visits) in the *conclusion* phase. Since this particular implementation did not lead to better performance in trial runs of the algorithms compared to the regular scoring function, this variation will not be considered in this thesis.
Sometimes MCTS implementations will also evaluate losses with a penalty (in the form of a -1 update to the scores in the backpropogation phase for example). However, this is again not logical in this implementation due to the cooperative nature of the game. There will be no opposing player trying to move towards a losing state. Moreover, the number of losing playouts is much larger than the number of winning playouts in most cases. This means that implementing this scoring method would simply lead to devaluing nodes that are visited more often.

5 Experiments

We have tested the various agents for both the task-phase and the playing phase of the game described in Section 4. Specifically, we have simulated every possible combination of the task and playing phase agents for games with task numbers ranging from 1-20. For each task number 1000 random seeds where generated to seed each of the games. This ensures that the initial state of each trial is the same for each agent. Moreover, the random seed was set again after the task-phase, ensuring that the playing-phase agent played the same, regardless of the choices of the task-phase agent (e.g. the Random-player will always do the same moves no matter how the tasks were distributed). The full results of these experiments can be found in Appendix A. In this appendix, two views on the data can be found: one based on the task-phase used and one based on the playing-phase used.

We now highlight a few interesting results.
First of all, the MCTS-UCT agent showed the best results overall, followed by the PMC agent. However, the difference between the two is only truly apparent, when the 'smart' scoring function is used, as can be seen in Figure 5. An explanation for this result would be that, when no winning solution has been found, both agents end up doing random playouts for random direct children of the root node. The only difference is, that PMC selects children in a uniform random manner and MCTS-UCT selects based on the UCT score. When no solution has been found, however, MCTS-UCT selects purely based on high uncertainty. This would also lead to each child being selected roughly the same number of times, just like selecting in a uniform random manner.
On the contrary, when the smart scoring function is used, MCTS-UCT is able to distinguish between children for which no winning playout has been found yet. Children with playouts that end in a higher number of tasks completed are more likely to have winning playouts, and UCT ensures that more visits will be done to these children. This leads to MCTS-UCT being more likely to find a winning playout even during the first few moves when the search space is very large and, thereby, being able to select the moves that are better than the moves PMC selects.

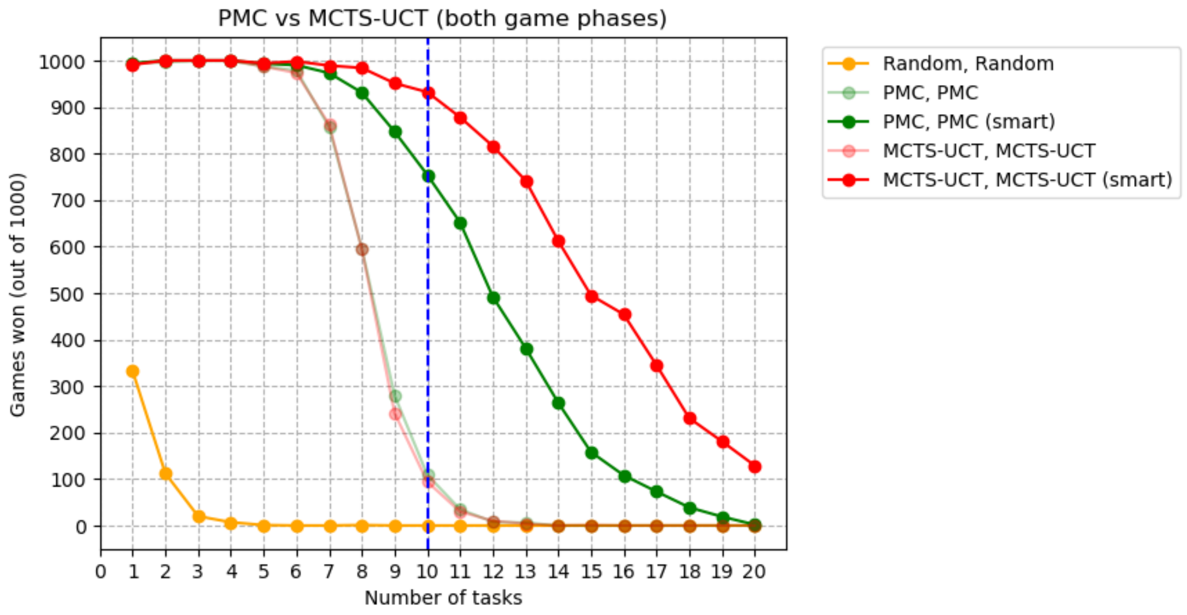


Figure 5: Comparison of the performance of PMC (‘Pure’ Monte Carlo player) and MCTS-UCT (Monte Carlo Tree Search using the Upper-Confidence Bound for Trees), when both are applied to both game-phases. The pure Random player (both phases of the games played by random agents) is also visible as a baseline. With the smart-scoring MCTS function both agent types perform better than with the regular scoring function. Moreover, a clear difference in performance between the PMC and MCTS-UCT players can be seen when applying the smart scoring function.

Secondly, distributing tasks randomly severely limits the potential of all agents, which is in line with the results found in [dJ21] using SAT-solvers. However, selecting tasks based on even just the simple evaluation function used by the Tactical task-taker improves the winnability of the game significantly. This can be seen in Figure 6.

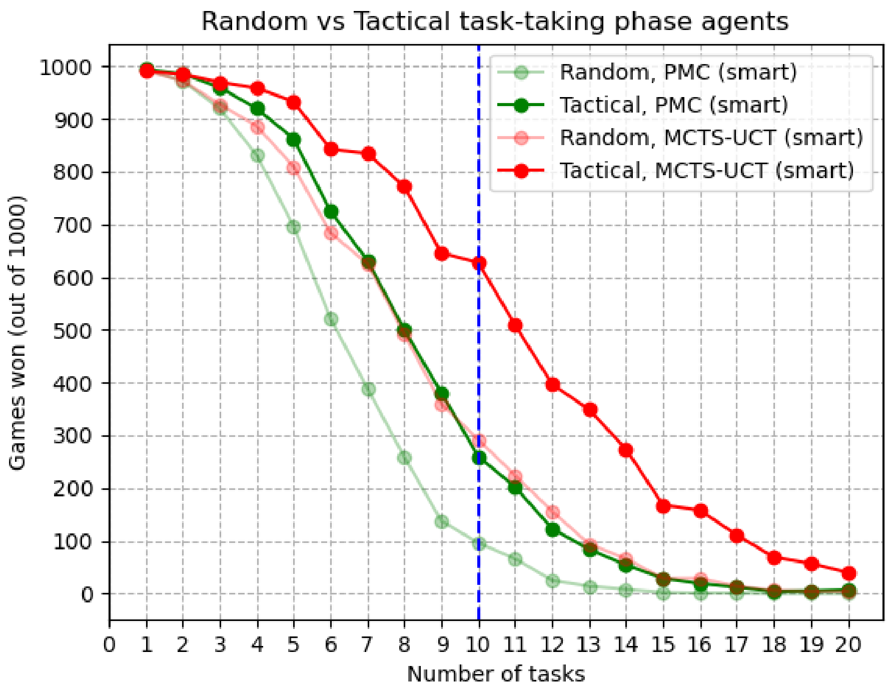


Figure 6: Comparison between the Random and Tactical task-taking agents (using smart-scoring PMC and MCTS-UCT players for the playing phase). In both cases using the Tactical task-taking agent improves the performance significantly for both Monte Carlo agents, especially for the 'standard' task numbers (≤ 10). This shows the importance of the Task-taking phase as well as that even a simple intuitive function to pick tasks significantly improves the chance of winning over just picking at random.

Thirdly, the RandomSmart player shows clear improvement over the true Random player, as can be seen in Figure 7. The number of games won by both agents is still very limited when compared to the number of games won by Monte Carlo agents.

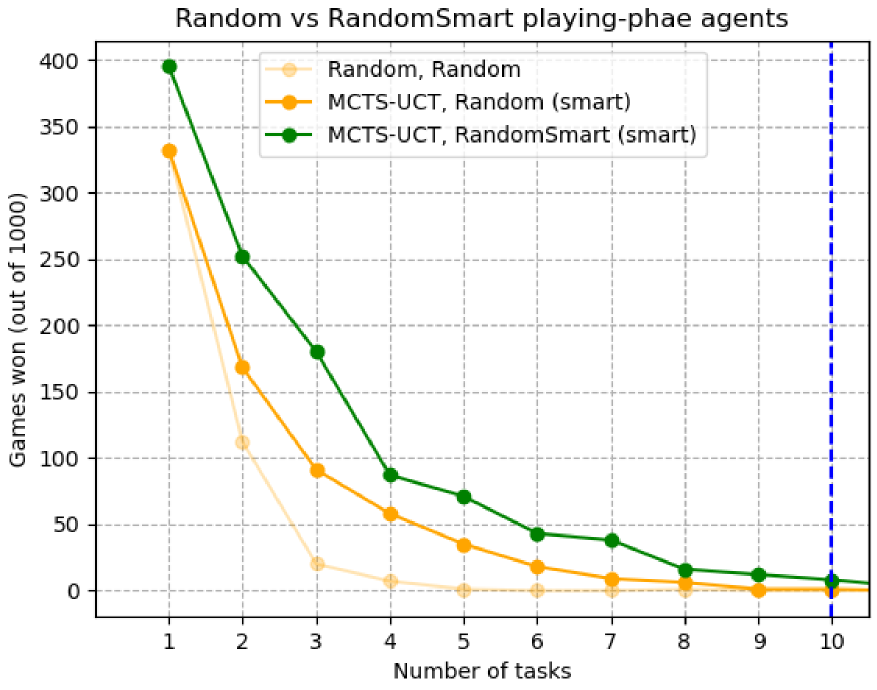


Figure 7: Comparison between the Random and RandomSmart playing-phase agents (using MCTS-UCT with the smart scoring function for optimal task-selection), the pure random player is also visible as a baseline. The RandomSmart player clearly outperforms the Random player. Both players can only win a significant number of games for small numbers of tasks.

Lastly, the maximum number of tasks specified by the *The Crew* rulebook is 10 and the pure MCTS-UCT player is capable of winning $> 90\%$ of these games. To test the limits of our Monte Carlo agents we increased the maximum number of tasks considered in our experiments to 20. At 20 tasks very few games can be won even by the Monte Carlo agents. It is interesting to note that for more than 10 tasks the number of games won by our strongest player begins to drop quickly. Whether intentional or not, *The Crew* has picked the maximum number of tasks described by its rules in a manner that matches well with the idea of difficulty/achievability given by our agents. However, [dJ21] has shown that even these games are expected to be winnable most of the time, so perhaps with more adjustments an agent can be created that is able to win nearly all games for up to 20 (or more) tasks, too.

6 Conclusions and Further Research

In this thesis we have formally defined a 2-player version of *The Crew*. We calculated the total number of games of this form with a single task, accounting for symmetries over colourings of all cards and for symmetries relating to the task card specifically. We have defined when such a game is winnable. Moreover, we calculated the total number of (non-)winning games accounting for symmetries over the task-card, as well as the fraction of the total number of games. Particularly, the percentage of winnable games of this form is 99.72%. It is difficult to prove this formally for unbounded numbers of players/-tasks (as is shown in [Rei21]). However, these results are in line with SAT-solver experiments done for 4-player games with up to 23 tasks (see [dJ21]). Therefore, it seems a high percentage of winnability is not exclusive to 2-player 1-task games. We have also found that, when splitting losing games into the categories based on whether or not the critical card is in the hand of player 1 and player 2, the amount of losing games is the same in each category. We have defined a bijective mapping over a large subset of these games.

Additionally, we have defined and evaluated several agents for both the task-taking and playing phases of the game *The Crew*. We implemented these agents on a simplified 3-player game of *The Crew*. We have established a baseline for a player agent taking purely random tasks and playing purely random moves. We have improved on this baseline slightly in the playing phase by introducing the RandomSmart player for the playing phase. We have improved upon the baseline in the task-taking phase by introducing the Tactical player. Furthermore, we have considered two implementations of Monte Carlo players: a Pure Monte Carlo player (PMC) and a Monte Carlo Tree Search player using Upper-confidence Bound for Trees selection (MCTS-UCT). We can conclude that Monte Carlo methods using complete information can be used to play the game effectively, especially when combining it with a ‘smarter’ scoring function and Upper Confidence Bound for Trees. Using the Monte Carlo agents we were able to conclude that the simple heuristic used for the Tactical task-taking player is a great improvement over selecting tasks randomly.

There are many options for further research on *The Crew*. Firstly, more rules of the game can be incorporated into the simulation and the agents. These rules include adding trump cards and incorporating communication or distress signals into the agents’ strategies. Moreover, the Monte Carlo agents we have defined, make use of complete information, whereas this is not the case in the actual game. Using determinizations (for example those described in [dJ21]) to determine possible game states would allow Monte Carlo agents to be used with incomplete information. This would asses their ability to win *The Crew* more accurately, as information is incomplete in the actual game.

Furthermore, the scoring functions for the Tactical task-taking agent players can be fine-tuned to a much higher degree. The same holds for the scoring and the selection functions of the Monte Carlo agents. In this thesis, the goal for these functions was simply to make an improvement over the random task-taker and the ‘standard’ scoring function, so there are most likely better optimized alternatives for further improvements.

Moreover, the rule-based algorithmic agents can be improved upon by taking into account more details of the game. These details could include taking note of which players do not have cards in a certain colour or what cards they have communicated (if communication is added to the simulation).

Lastly, the theoretical analysis can also be expanded upon. Particularly to games with more players, more tasks and/or with trump cards. It might also be interesting to analyse to what extend adding the 'distress signal' rule (which allows players to swap playing cards once) will improve winnability.

References

Paolo Ciancarini and Gian Piero Favini. Monte Carlo tree search in Kriegspiel. *Artificial Intelligence*, 174(11):670–684, 2010. URL: https://www.sciencedirect.com/science/article/pii/S0004370210000536, doi:10.1016/j.artint.2010.04.017.

Sam Chan. Performance of a multi-agent greedy algorithm in a cooperative game with imperfect information. Bachelor thesis, Utrecht University, 2021. URL: https://studenttheses.uu.nl/handle/20.500.12932/1209.

Guillaume Chaslot, Mark Winands, H. van den Herik, Jos Uiterwijk, and Bruno Bouzy. Progressive Strategies for Monte-Carlo Tree Search. *New Mathematics and Natural Computation*, 04:343–357, 11 2008. doi:10.1142/S1793005708001094.

Aron de Jong. An Analysis of the Cooperative Card Game The Crew. Bachelor thesis, Leiden University, 2021. URL: https://theses.liacs.nl/pdf/2020-2021-JongAde.pdf.

Joachim Kock and Thomas Jan Mikhail. Free loop spaces and the Cauchy–Frobenius Lemma, 2025. URL: https://arxiv.org/abs/2507.01637, arXiv:2507.01637.

Levente Kocsis and Csaba Szepesvári. Bandit Based Monte-Carlo Planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer Berlin Heidelberg, 2006.

Jean Méhat and Tristan Cazenave. Monte-Carlo Tree Search for General Game Playing. 2008. URL: https://api.semanticscholar.org/CorpusID:32291693.

Frederick Reiber. The Crew: The Quest for Planet Nine is NP-complete. 2021. URL: https://arxiv.org/abs/2110.11758, arXiv:2110.11758.

Joseph J. Rotman. *An Introduction to the Theory of Groups.* Graduate Texts in Mathematics, 148. Springer New York, 4th edition, 1995.

Douglas Rebstock, Christopher Solinas, Nathan R. Sturtevant, and Michael Buro. Transformer Based Planning in the Observation Space with Applications to Trick Taking Card Games, 2024. URL: https://arxiv.org/abs/2404.13150, arXiv:2404.13150.

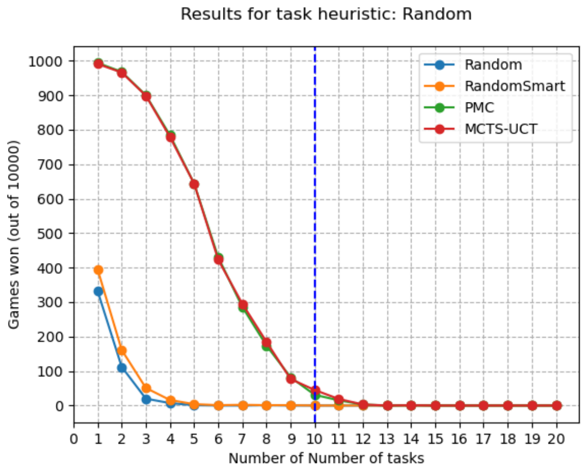
Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte Carlo Tree Search: A Review of Recent Modifications and Applications. *Artificial Intelligence Review*, 56(3):2497–2562, July 2022. URL: http://dx.doi.org/10.1007/s10462-022-10228-y, doi:10.1007/s10462-022-10228-y.

Appendices

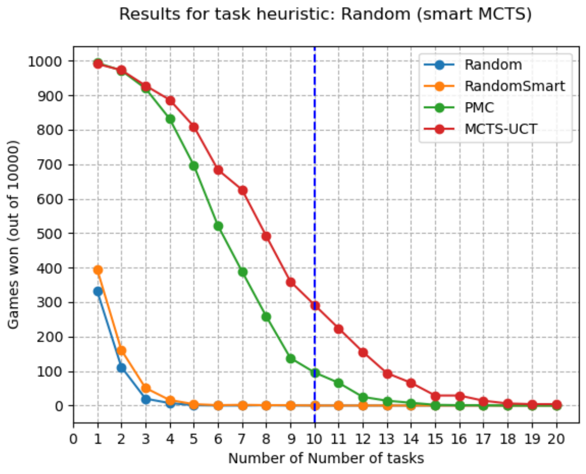
A Full experimental results

All results of the experiments can be found here. The results are given in two different views (one based on task-phase and one based on playing phase) that both represent the same data.

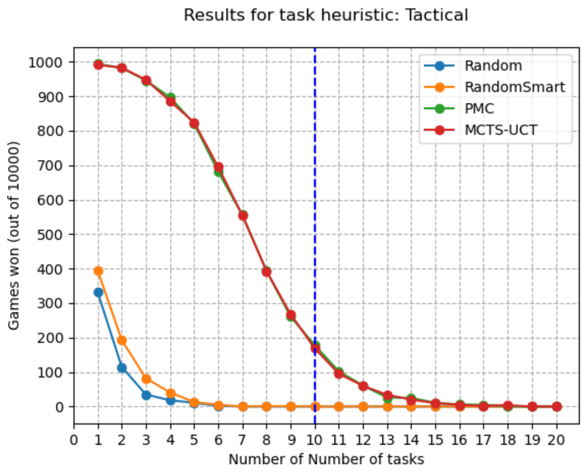
A.1 Results by task-phase



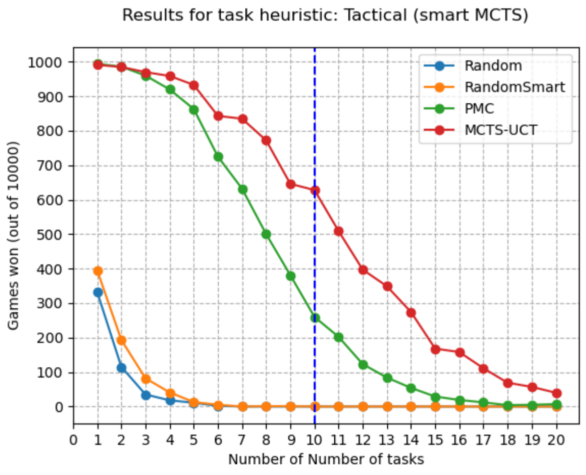
(a) Results for the Random task-taking agent.



(b) Results for the Random task-taking agent, with smart MCTS scoring for Monte Carlo agents.

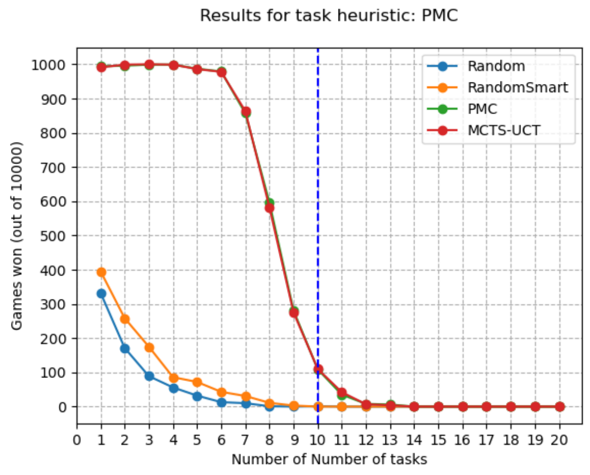


(c) Results for the Tactical task-taking agent.

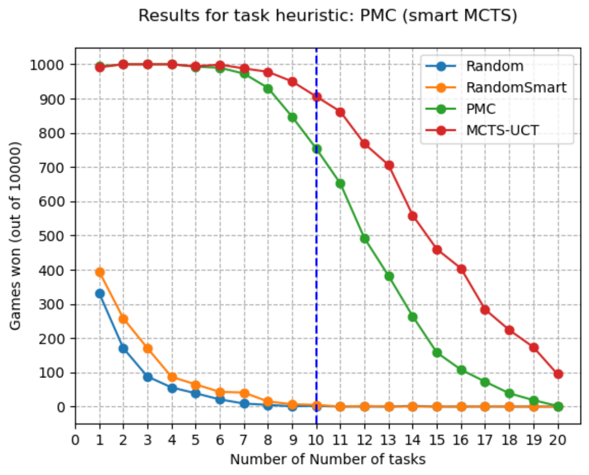


(d) Results for the Tactical task-taking agent, with smart MCTS scoring for Monte Carlo agents.

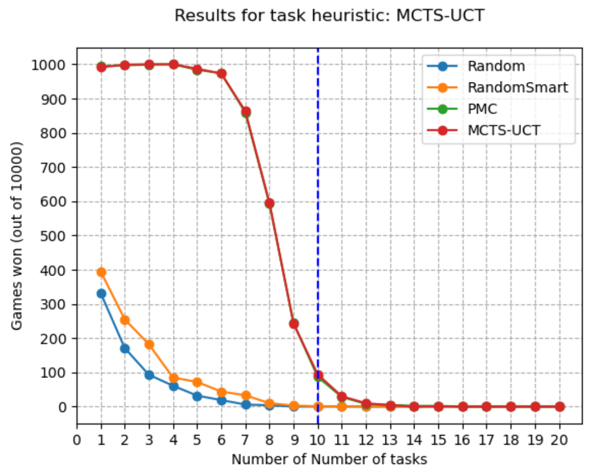
Figure 8: Experiment graphs based on task-taking-phase for the Random and Tactical agents. Tasks won (out of 1000) with the given task-taking-phase agent and various playing-phase agents.



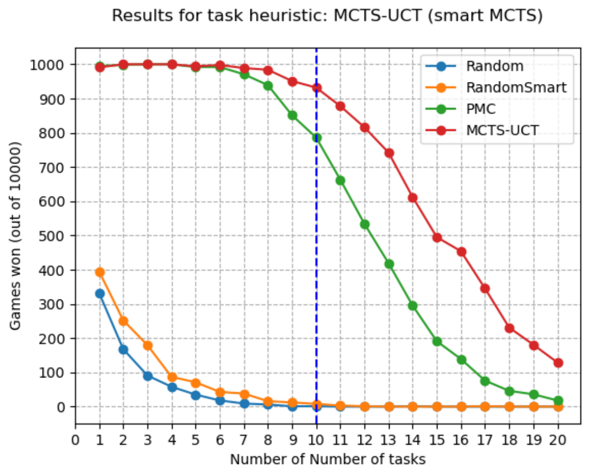
(a) Results for the PMC task-taking agent.



(b) Results for the PMC task-taking agent, with smart MCTS scoring for Monte Carlo agents.



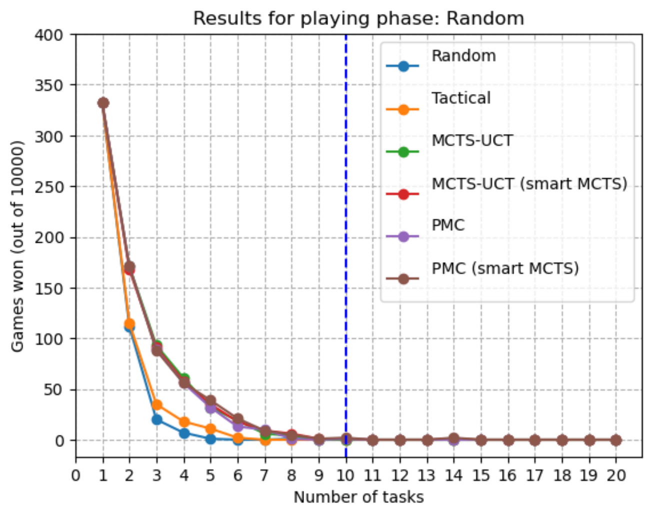
(c) Results for the MCTS-UCT task-taking agent.



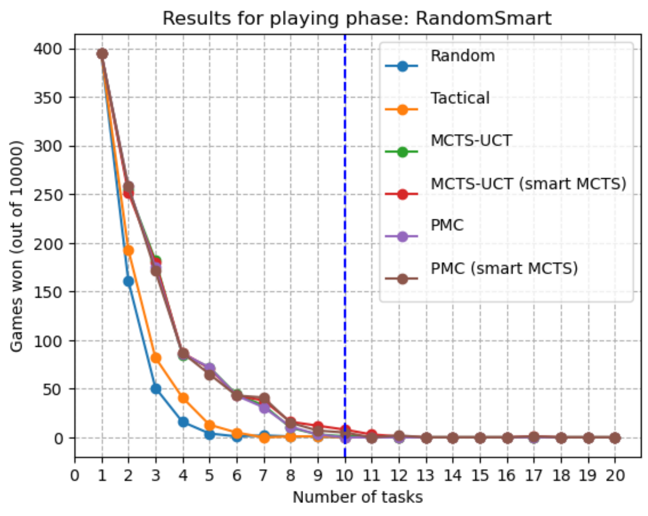
(d) Results for the MCTS-UCT task-taking agent, with smart MCTS scoring for Monte Carlo agents.

Figure 9: Experiment graphs based on task-taking-phase for the Random and Tactical agents. Tasks won (out of 1000) with the given task-taking-phase agent and various playing-phase agents. The 'smart' MCTS scoring function is used (or not used) in both the task-taking and the playing phase in all cases (when applicable).

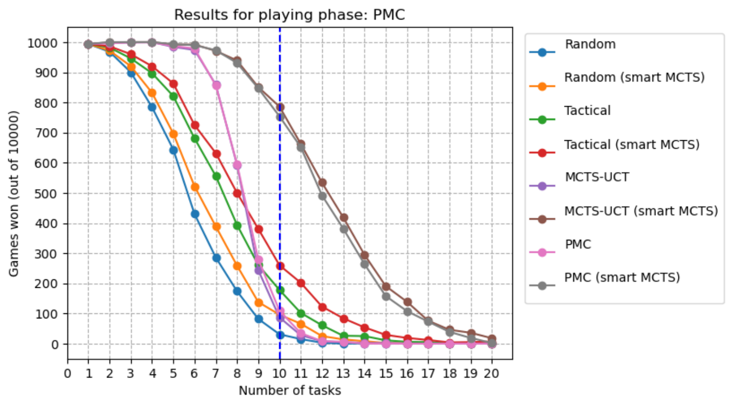
A.2 Results by playing-phase



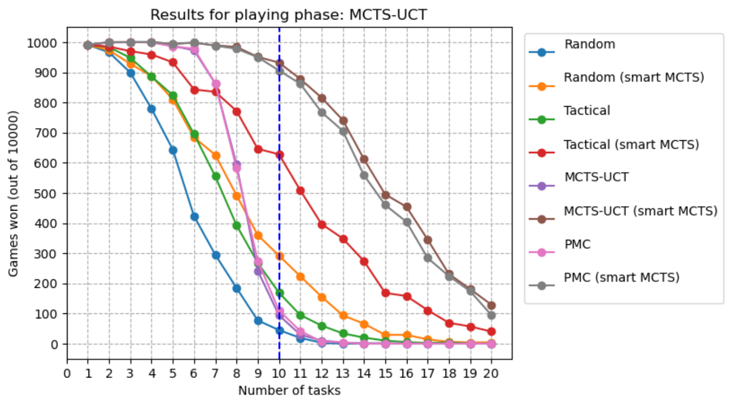
(a) Results for the Random playing-phase agent.



(b) Results for the RandomSmart playing-phase agent.



(c) Results for the PMC playing-phase agent.



(d) Results for the MCTS-UCT playing-phase agent.

Figure 10: Experiment graphs based on playing-phase. Tasks won (out of 1000) with the given playing-phase agent and various task-taking agents. The ‘smart’ MCTS scoring function is used (or not used) in both the task-taking and the playing phase in all cases (when applicable).