Karlsruhe Institute of Technology

The Compexity of the Oriented Cycle Game

Bachelor Thesis of

Shabi Shabani

At the Department of Informatics
Institute of Theoretical Informatics

Reviewers:	PD Dr. Torsten Ueckerdt
	T.T.-Prof. Dr. Thomas Bläsius
Advisor:	PD Dr. Torsten Ueckerdt

Time Period: 24th November 2022 – 24th March 2023

Statement of Authorship

Ich versichere wahrheitsgemäß, die Arbeit selbstständig verfasst, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde sowie die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet zu haben.

Karlsruhe, March 24, 2023

Abstract

The *oriented cycle game*, which was introduced by Bollobás and Szabó, is a strategy game played on undirected graphs in which both players, A(voider) and C(reator), assign orientations to undirected edges alternately until there are no more undirected edges. Unlike the original game, where player C tries to create a directed cycle and player A tries to prevent him, we introduce two new variations. First, we introduce the *avoid oriented cycle game*, where both players try to prevent a directed cycle. The first player to create a directed cycle loses. Next, we introduce the *seek oriented cycle game*, where both players try to create a directed cycle. The first player to create a directed cycle wins. Following, we introduce the *oriented edge groups*, which is a relation between oriented edges. Here, a player, when orienting an undirected edge, must orient the other edges that are in relation to the undirected edge accordingly.

First, we look at the decision problem *"Does the current player possess a winning strategy?"* and show that this decision problem is PSPACE-complete under certain properties of the oriented edge groups for our newly presented game variations. After that, we will look at certain properties of the *oriented edge groups*, where the decision problem is solvable in polynomial time.

Finally, we look at some interesting graphs, such as the grid graph, and indicate which player wins on them in the original *oriented cycle game* from Bollobás and Szabó.

Deutsche Zusammenfassung

Das *Oriented Cycle Game*, welches von Bollobás und Szabó eingeführt wurde, ist ein Strategiespiel auf ungerichteten Graphen, bei dem beide Spieler, A(voider) und C(reator), ungerichtete Kanten abwechselnd Orientierungen zuweisen, bis es keine ungerichteten Kanten mehr gibt. Im Gegensatz zum original Spiel, bei dem Spieler C versucht einen gerichteten Zyklus zu erzeugen, und Spieler A versucht ihn daran zu hindern, führen wir zwei neue Varianten ein. Erstens führen wir das Spiel *avoid oriented cycle game* ein, bei dem beide Spieler versuchen einen gerichteten Zyklus zu verhindern. Der Spieler, der einen gerichteten Zyklus zuerst erzeugt, verliert. Als nächstes führen wir das *seek oriented cycle game* ein, bei dem beide Spieler versuchen, einen gerichteten Zyklus zu erzeugen. Der Spieler, der einen gerichteten Zyklus zuerst erzeugt, gewinnt. Als nächstes führen wir die *oriented edge groups* ein, die mehrere Relationen zwischen orientierten Kanten darstellen. Hier muss ein Spieler, wenn er eine ungerichtete Kante richtet, die anderen Kanten, die in einer Relation zur gerichteten Kante steht, auch entsprechend richten.

Zunächst betrachten wir das Entscheidungsproblem *"Besitzt der aktuelle Spieler eine Gewinnstrategie?"* und zeigen, dass dieses Entscheidungsproblem unter bestimmten Eigenschaften der orientierten Kantengruppen PSPACE-vollständig ist für unsere neu vorgestellten Spielvarianten. Danach werden wir bestimmte Eigenschaften der *oriented edge groups* betrachten, bei denen das Entscheidungsproblem in polynomieller Zeit lösbar ist.

Schließlich betrachten wir einige interessante Graphen, wie zum Beispiel den Gittergraphen, und geben an, welcher Spieler in dem ursprünglichen *Oriented Cycle Game* von Bollobás und Szabó gewinnt.

Contents

1. Introduction

The *oriented cycle game*, invented by Béla Bollobás and Tamás Szabó, is a maker-breaker game that falls under the category of positional games [BS98]. Two players, A(voider) and C(reator) participate in the orientation cycle game on an undirected graph G. Edges of G are directed by the players alternately. Player C, who initiates the game, only directs one edge. In response, exactly one previously undirected edge is directed by player A. Finally, we obtain a directed graph G' once the players have directed all of the edges of G. The directed graph G' must have at least one directed cycle for player C to win; otherwise, player A wins.
Let us consider the game played on the triangle graph shown in Fig. 1.1. Player C chooses one of the three edges and orients it. Because of the symmetry of the graph, we can assume that player C has directed the upper edge towards the right (Fig. 1.2). In order for player A to win and prevent the directed cycle, he can now make the node at the top right a sink. For a node to become a sink, it must have no outgoing edges. The top right node becomes sink by orienting the bottom right edge to the right as shown in Fig. 1.3. Now player C has only one edge left to orient and both possibilities to orient this edge lead to no directed cycle and therefore to a defeat for player C. This gives us a winning strategy for player A on the triangle graph.

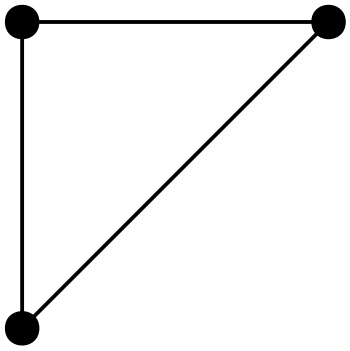

Figure 1.1: An undirected triangle

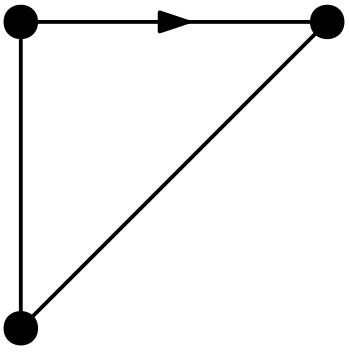

Figure 1.2: A triangle with one directed edge

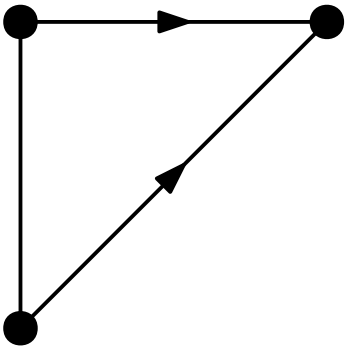

Figure 1.3: Losing configuration for player C

1.1 Variants of the game

This section will focus on specific game variations in more detail. This involves changing the basic rules of the game or the goals of the individual players. It is crucial to keep in mind that these variations of the game can also be combined in a meaningful way. This makes sense when one variant changes the basic rules of the game and the other variant changes the objective of the game.

1.1.1 Biased game

Depending on our graph G, an unfair power distribution may take place. For instance, if G has few edges, A might have a significant advantage over C because it is unlikely that a directed cycle will form. If G has many edges, then there can be more opportunities to form a cycle and C would have a tremendous amount of power. Giving the weaker party more power is one way to maintain a healthy balance of power. As a result, Chvátal and Erdös introduce biased maker-breaker games [CE78]. In our case, we let players direct multiple edges at once.
Let $a, b \in \mathbb{N}$ be positive numbers that we call biases of players A and C. In the $(a : b)$ *biased oriented cycle game*, as before, we let the players take turns directing undirected edges on the graph G, where in each round player C directs a edges and then player A directs b edges. In the very last round, if the number of undirected edges of G is less than the player's bias, the player must direct all remaining edges. Note that the $(1 : 1)$ *biased oriented cycle game* is the original *oriented cycle game*.

1.1.2 Avoid game & seek game

We attempt to combine A and C's interests in the avoid game and the seek game. Therefore, for avoid and seek games, player 1 and player 2 will be used instead of players A and C. In the avoid game, avoiding closing a cycle is the objective. The player who closes a cycle by setting an edge loses. Therefore, the winner is the player who can force their opponent to close a cycle. It is not decisive who has oriented the other edges in the directed cycle, but only the player who closes the oriented cycle loses. The seek game, on the other hand, is the complete opposite. The objective in the seek game is to finish a cycle first. As a result, the winner is the first person to finish a cycle by orienting an edge. In this case, as well, it is not decisive who has oriented the other edges in the directed cycle, but only the player who closes the directed cycle wins. If no cycle forms in either of the two game scenarios, a draw happens. We assume for the sake of simplicity that forcing a draw is not an ideal strategy when forcing a win is also available.

1.1.3 Oriented edge groups

To make our game a little more interesting, we introduce oriented edge groups M. To do this, we give both players a table that contains which oriented edges are in a group. An oriented edge group is a set of oriented edges. If a player decides to orient an undirected edge e, he must look in the table to see which oriented edges are in a group with e. The orientation is decisive here since other-oriented edge groups are possible for both orientations of e. Once the oriented edge group has been found, each edge that is not oriented in this group must be oriented in the way prescribed by the group. If there are several oriented edge groups that contain the oriented edge e, all edges from all oriented edge groups that contain e will be oriented accordingly. If there is no oriented edge group for this oriented edge, then this edge cannot be oriented. This also allows the player to orient several edges simultaneously, but he is not as free to decide which edges to orient as in the biased game. The game ends when one player has won or until there are no more valid oriented edge groups.

1.1.4 Partizan game

In our original game, both players were allowed to direct an edge that was not yet directed, so they have the same moves available. In game theory, such games are called *impartial games* [Dem01]. In partizan games, we do not want to have this and want to give one player certain moves that the other player doesn't have. An example of a partizan game is chess. Only one player can play with the white pieces and the other player is not allowed to move these white pieces. We can also turn our game into a partizan game by coloring the edges of G in black and white. This is done by coloring the edges fairly so that there are 50% white edges and 50% black edges. Thus we allow player C to play only on white edges and player A to play only on black edges. Both players take turns again and orient an edge of their color. Player C tries again to close a directed cycle, the directed cycle may have any color combination. This means that player C also wins if the directed cycle has alternating edge colors at the end of the game. And as usual, player A tries to prevent a directed cycle, regardless of the color of the edges.

1.2 Outline and Example

After looking at definitions and some decision problems in Chapter 2, in Chapter 3 we take a closer look at the *avoid oriented cycle game* with oriented edge groups and analyze it in terms of complexity, how hard it is to find out if the current player can force a win, depending on the oriented edge groups. Chapter 4 deals with the *seek oriented cycle game*, again with oriented edge groups. There we again analyze the game in terms of complexity, and how hard it is to find out if the current player can force a win, depending on the oriented edge groups. Finally, in chapter 5 we look at the normal *oriented cycle game* without oriented edge groups and try to gather some new insights for the game.

Let us consider the game played on the house graph shown in Fig. 1.4. Here, both players have given the following oriented edge groups M:
$M = \{\{(v_3, v_2)\}, \{(v_2, v_4), (v_5, v_4)\}, \{(v_2, v_1), (v_1, v_3)\}, \{(v_5, v_3), (v_4, v_2)\}\{(v_5, v_4), (v_3, v_5)\}\}$.
In this case, we look at the avoid version, that is, the player who creates a directed cycle loses. In the oriented edge groups, oriented edges are given as a group.
For example, suppose player 1 starts the game and orients the edge from node v_2 to node v_1. Now we look into the oriented edge groups to find groups. Only the group $\{(v_2, v_1), (v_1, v_3)\}$ is found that has exactly this edge with the orientation. Since the oriented edge (v_1, v_3) is in this group, it must also be oriented as the group specifies. This means that when orienting the edge from v_2 to v_1, the edge from v_1 to v_3 is oriented at the same time as shown in Fig. 1.5.
Now it is player 2's turn and he would like to orient the edge from v_2 to v_3. After looking at which oriented edges are in a group with his chosen edge, he notices that this edge with this orientation is not in any group. This means that he cannot orientate this edge in this way and must choose another one. We assume that player 2 orients the edge from v_2 to v_4. Now we look again at the oriented edge groups to find groups. The only group that has the selected edge with matching orientation is $\{(v_2, v_4), (v_5, v_4)\}$. Thus, at the same time as orienting the edge from v_2 to v_4, we also orient the edge from v_5 to v_4. We do not orient the edge of v_3 to v_5, even if it is in a group with the edge v_5 to v_4, because we have oriented the edge v_2 to v_4 and only orient the groups that have exactly this edge with the matching orientation as can be seen in Fig. 1.6.
Since there is still no directed cycle in the graph, it is player 1's turn again. We assume that player 1 chooses to orient the edge from node v_3 to v_5. Looking at the oriented edge groups containing the selected edges, we get $\{(v_5, v_4), (v_3, v_5)\}$ as the only group. However, since the other edge in the group is already oriented and there are no other edges besides the selected one, only the edge from v_3 to v_5 is oriented as can be seen in Fig. 1.7.

It is player 2's turn again, as there is still no directed cycle. However, there is only one edge left and, as we have already seen, this edge can not be oriented from v_2 to v_3. This leaves only the orientation of this edge from v_3 to v_2. Since this edge with this orientation alone represents a group, no other edge is oriented with it. However, with this move a directed cycle $v_1 \to v_3 \to v_2$ was created and therefore player 2 loses if the game is played this way.

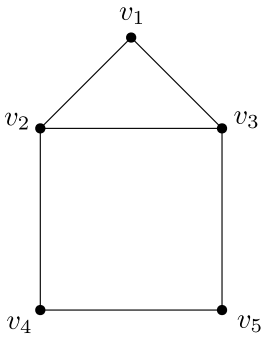

Figure 1.4: The house graph

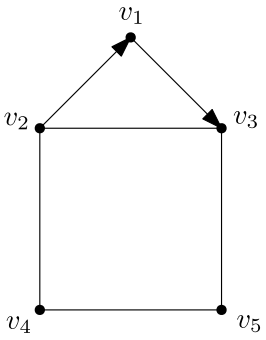

Figure 1.5: The house graph after the 1st move

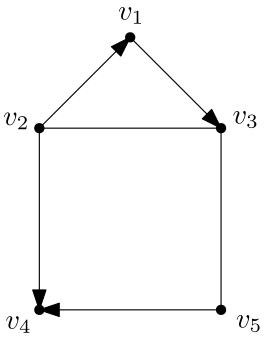

Figure 1.6: The house graph after the 2nd move

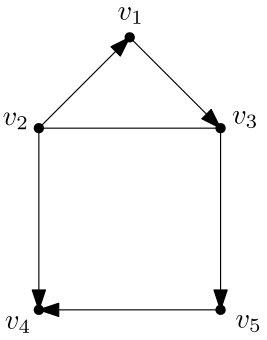

Figure 1.7: The house graph after the 3rd move

1.3 PSPACE

In theoretical computer science, determining the difficulty of a given decision problem is interesting. Different complexity classes have emerged as a result of this. It has already been established for numerous board games that the decision problem of whether a player has a winning strategy belongs to a specific complexity class. The decision problem is oftentimes PSPACE-complete when we narrow our attention to two-player board games. PSPACE is the set of problems that can be solved in polynomial space or to be more precise PSPACE is the complexity class of decision problems L for which a Turing machine M exists for which:

1. M holds for every input
2. M accepts an input $x \in \{0,1\}^*$ if and only if $x \in L$;
3. there exists a polynomial p such that the size of the tape of M $Space_M(n)$ used during calculations is limited by p.

$$Space_M(n) \leq p(n)$$

In order to prove that a decision problem L is PSPACE-complete we must prove that $L \in PSPACE$ and we must show that L is PSPACE-hard. We can provide an algorithm that solves L while using only a polynomial amount of space in order to demonstrate that $L \in PSPACE$.
To show that L is PSPACE-hard, we take another problem Π which is known to be PSPACE-hard, and prove that Π can be reduced to L in polynomial space. This demonstrates that problem L is at least as difficult to solve as problem Π, which is PSPACE-hard, and that solving Π is not computationally more difficult than solving L.
A reduction from a language $\Pi \subseteq \Sigma^*$ to a language $L \subseteq \Sigma^*$ is a function $f : \Sigma^* \to \Sigma^*$ such that:

- f is computable by a polynomial space algorithm
- $w \in \Pi \Leftrightarrow f(w) \in L$

We then say Π is reducible to L and write $\Pi \leq_p L$ [AB09].
For an example of a PSPACE-complete decision problem, one can take *undirected edge geography* by Fraenkel, Scheinerman, and Ullman [FSU93]. In the game *undirected edge geography*, two players take turns moving a token on an undirected graph from one vertex to an adjacent vertex, erasing the edge in between. The loser is the player who cannot make a move legally first. The associated decision problem, which is PSPACE-complete, is: *"Given a token on an undirected graph, is this position a win for the first player?"*

2. Preliminaries

In this chapter, we establish some assumptions and notations in this thesis. We also look at problems known in complexity theory and used in the thesis before delving into the complexity of the *oriented cycle game*.

2.1 Assumptions and Notations

In this thesis, we consider mixed graphs $G = (V, E, A)$ unless specified otherwise. The set of nodes in G is called V, $E \subseteq \binom{V}{2}$ is the set of undirected edges and $A \subseteq V \times V$ is the set of directed edges. Consider the connected nodes $u, v \in V$. A directed edge is an edge with an orientation and is denoted by $\overrightarrow{uv}$ or (u, v). In this instance, the edge points from u to v. Undirected edges lack orientation and are identified by the symbols uv or $[u, v]$. For undirected edges, $[u, v]$ refers to the same edge as $[v, u]$. In addition, there are no multiple edges between two nodes and there is no edge connecting a node to itself.

For a given mixed graph $G = (V, E, A)$ and a node $v \in V$, we define $G' = G - v$ as the resulting mixed graph by removing v from V and all edges incident to v. To be more precise, the following applies to $G' = (V', E', A')$: $V' = V/\{v\}$, $E' = \{[x, y] \in E \mid x \neq v, y \neq v\}$ and $A' = \{(x, y) \in A \mid x \neq v, y \neq v\}$.

A mixed *subgraph* $G' = (V', E', A')$ of a mixed graph $G = (V, E, A)$ has as its nodes a subset of the nodes of G, $V' \subseteq V$. The undirected edges E' and the directed edges A' are also subsets of the undirected edges E and the directed edges A, i.e. $E' \subseteq E$ and $A' \subseteq A$. The mixed graph $G[X] = (V'', E'', A'')$ induced by a set of nodes $X \subseteq V$ is the mixed subgraph of $G = (V, E, A)$ with nodes X, undirected edges $E'' = \{[x, y] \in E \mid x, y \in X\}$ and directed edges $A'' = \{(x, y) \in A \mid x, y \in X\}$.

2.2 Laman graphs

A *Laman graph* $G = (V, E, A)$ is a mixed graph such that, for all k, every mixed subgraph $G[X]$ with $|X| = k$ has at most $2k - 3$ edges, and such that G has exactly $2|V| - 3$ edges [Lam70]. The Laman graphs can be obtained by starting from a single undirected edge and making a sequence of operations of the following two types:

1. Henneberg 1 step (Hen_1): Add a new node to the graph, together with edges connecting it to two previously existing nodes.

2. Henneberg 2 step (Hen_2): Subdivide an undirected edge of the graph, and add an edge connecting the newly formed vertex to a third previously existing vertex.

A sequence of these operations that result in a Laman graph is called a *Henneberg construction* [Hen11].

2.3 Oriented Edge Groups

For the oriented edge groups M we first introduce $\overrightarrow{E}$. Let $E \subseteq \{[u,v] \mid u,v \in V\}$ be the set of undirected edges, then $\overrightarrow{E}$ is defined as $\overrightarrow{E} = \{(u,v) \mid [u,v] \in E\} \cup \{(v,u) \mid [v,u] \in E\}$. Thus in $\overrightarrow{E}$ every undirected edge $e \in E$ is exchanged with two directed edges in opposite directions. Now we can define M as $M \subseteq 2^{\overrightarrow{E}}$ with $\forall m \in M : (a,b) \in m \Rightarrow (b,a) \notin m$ for $(a,b) \in \overrightarrow{E}$.
We say $m, m' \in M$ are *in conflict* if $(a,b) \in m$ and $(b,a) \in m'$ for $(a,b) \in \overrightarrow{E}$. We also say that M is *complete*, if $\forall \overrightarrow{e} \in \overrightarrow{E}\ \exists m \in M\ \overrightarrow{e} \in m$. Thus, every oriented edge $\overrightarrow{e}$ in $\overrightarrow{E}$ is in at least one oriented edge group m. If every directed edge $\overrightarrow{e}$ in $\overrightarrow{E}$ is in exactly one oriented edge group m, i.e. $\forall \overrightarrow{e} \in \overrightarrow{E}\ \exists! m \in M\ \overrightarrow{e} \in m$ then we call M *perfectly complete*.
We call M *symmetrical* if $\forall m \in M\ \exists m^{-1} \in M$, where m^{-1} is the oriented edge group that has the same edges as m but in the other orientation.
Finally, we call M *unique* if each directed edge $\overrightarrow{e} \in \overrightarrow{E}$ appears in at most one oriented edge group $m \in M$.

2.4 Quantified SAT

As said above in section 1.3, to show that a problem is PSPACE-hard we need a PSPACE-complete problem to reduce from. Therefore, we present here some simple problems that are already proven to be PSPACE-complete.
First, we introduce the quantified SAT problem (QSAT or QBF), which is a generalization of the satisfiability problem [CKS01]. The satisfiability problem, also called SAT, is the problem where one is given a boolean formula and must determine whether there is an interpretation that satisfies the Boolean formula. In QSAT, we are given two parts: one that only performs the quantification for the variables in X, and the other that includes the boolean formula expressed as Ψ. During quantification, the variable is assigned a true or false value. The entire formula Φ can be written down as follows if there are n boolean variables:

$$\Phi = \exists x_1 \forall x_2 \exists x_3 \ldots Q x_n : \Psi(x_1, x_2, x_3, \ldots, x_n)$$

where $Q \in \{\exists, \forall\}$ based on whether n is even or odd, and $X = \{x_1, x_2, \ldots, x_n\}$ are boolean variables. The question for this problem would be: Given a QSAT instance Φ, is it true? Let us consider Φ_y, where $\Phi_y = \exists x_1 \forall x_2 : x_1 \vee x_2$, as an example of an instance that is in QSAT. Because there is an existential quantifier before the x_1, we can select the truth assignment for x_1. Whenever $x_1 = true$, regardless of x_2, the formula $x_1 \vee x_2$ is satisfied. We can use Φ_n as an instance that is not in QSAT, where $\Phi_n = \exists x_1 \forall x_2 : x_1 \wedge x_2$. The formula must be true for every x_2, but for $x_2 = false$, the formula cannot be satisfied no matter how we choose our x_1.

2.4.1 QSAT as a two-player game

QSAT can be seen as a two-player game as well. Player 1 in this scenario is the existential quantifier, and player 2 is the universal quantifier. In other words, player 1 selects a truth assignment for x_1, player 2 selects a truth assignment for x_2, player 1 selects a truth assignment for x_3, etc. Additionally, player 1 wins if and only if Ψ is true and the question for the two-player game would be: Can player 1 satisfy Ψ no matter what player 2 does?

2.4.2 Variants of Quantified SAT

There are numerous QSAT variations, just like there are with the *oriented cycle game*. When we think of QSAT, we frequently imagine two players. These variants can also be linked in a meaningful way, so impartial seek QSAT makes sense, and a completely new game results from the existing QSAT. However, since we are looking for problems that are PSPACE-complete, the question is: *Which of these combinations are PSPACE-complete?* Thomas Schaefer has provided some evidence that proves which combination is PSPACE-complete [Sch78]. The list is as follows:

1. impartial default game positive 11-CNF SAT
2. impartial default game positive 11-DNF SAT
3. partizan default game CNF SAT
4. impartial/partizan avoid game positive 2-DNF SAT
5. impartial/partizan seek game positive 3-DNF SAT
6. impartial/partizan seek game positive CNF SAT

The positive statement says that only positive variables may be in the boolean formula Ψ and, in addition, the players may no longer select the truth assignment, but must alternately select variables and set them to true.
DNF SAT is an SAT instance where the boolean formula Ψ is in disjunctive normal form, i.e. Ψ consists of several disjunctive clauses $\Psi = C_1 \vee C_2 \vee \cdots \vee C_k$. The clauses $C_1, \ldots, C_k$ consist of several conjunctions of one or more variables $C_i = x_1 \wedge \cdots \wedge x_m$. n-DNF SAT is a DNF SAT instance, where each clause C_i has exactly n variables.
Accordingly, CNF SAT is an SAT instance, where the Boolean formula Ψ is in conjunctive normal form, that is, Ψ consists of several conjunctive clauses $\Psi = C_1 \wedge C_2 \wedge \cdots \wedge C_k$. The clauses $C_1, \ldots, C_k$ consist of several dis-junctions of one or more variables $C_i = x_1 \vee \cdots \vee x_m$. n-CNF SAT is a CNF SAT instance, where each clause C_i has exactly n variables.

2.4.2.1 Impartial

As mentioned in the section 1.1.4, there are games that are impartial, or in which both players have access to the same moves. In the normal QSAT instance, there is a fixed order as to which variable is used. It is also determined which player occupies which variable. In impartial QSAT, there would not be a set order, and the player whose turn it is could take over a variable that has not been claimed yet. The goal of the game, which has been set, is not changed, only the rules of the game, which variables are selected, are changed. The entire formula Φ would then look like this with n boolean variables:

$$\exists m_1 \in \{1, \ldots, n\} \exists x_{m_1} \forall m_2 \in \{1, \ldots, n\} \setminus \{m_1\} \forall x_{m_2} \exists m_3 \in \{1, \ldots, n\} \setminus \{m_1, m_2\} \exists x_{m_3} \\ \ldots \exists m_n \in \{1, \ldots, n\} \setminus \{m_1, \ldots, m_{n-1}\} \exists x_{m_n} : \Psi(x_1, \ldots, x_n) \quad (2.1)$$

where n is odd for simplicity. Thus, player 1 sets a truth assignment for a variable x_i selected by him, player 2 sets a truth assignment for a variable x_j with $x_i \neq x_j$ selected by him, and so on until all variables have been selected.
Let us consider $\Phi = x_1 \wedge (x_2 \vee x_3)$ as an impartial QSAT default game. As a reminder, player 1 here wants to fulfill Φ and player 2 wants to prevent him from doing so. Player 1 starts the game and must specify a truth assignment for x_1, x_2, or x_3. A winning strategy would be for player 1 to set x_1 to true. Now it is player 2's turn and he has to give a truth assignment for x_2 or x_3. However, it does not matter what player 2 decides, because after player 2 has assigned either x_2 or x_3, player 1 can set the remaining variable to true and the boolean formula is satisfied. This gives us a winning strategy for player 1.

2.4.2.2 Partizan

Both players have different options for making moves in partizan games. For this purpose, coloring is typically added. In the partizan QSAT, the set of all variables X is split into two disjoint and equal-sized subsets X_1 and X_2 with $X = X_1 \dot\cup X_2$ and $|X_1| = |X_2|$. The only variables that player 1 and player 2 may then assign are those from X_1 and X_2, respectively. This division into two equally sized disjoint subsets can be thought of as coloring each variable either *blue* or *red*. The number of *blue* variables is equal to the number of *red* variables. Now player 1 may only assign true or false values to the *red* variables without any restriction of generality, while player 2 does the same with the *blue* ones. Additionally, there is no requirement that the players choose the variables in a certain order in this version. The game objective, which was previously defined, remains unaffected by this modification, only the rules of the game, who may occupy which variable and when, have been changed.
The formula Φ would then looks like this with $X_1 = \{x_1, \ldots, x_n\}$ and $X_2 = \{x_{n+1}, \ldots, x_{2n}\}$:

$$\exists m_1 \in \{1, \ldots, n\} \exists x_{m_1} \forall m_2 \in \{n+1, \ldots, 2n\} \forall x_{m_2} \exists m_3 \in \{1, \ldots, n\} \setminus \{m_1\} \exists x_{m_3} \\ \ldots \forall m_n \in \{1, \ldots, n\} \setminus \{m_n, \ldots, m_{2n-1}\} \forall x_{m_n} : \Psi(x_1, \ldots, x_n) \quad (2.2)$$

Player 1 picks a variable x_i from his set X_1 and then assigns this variable either true or false, then it is player 2's turn and picks a variable x_j from his set X_2 and again assigns it to either true or false. Player 1 now chooses another variable x_h from X_1, where $x_i \neq x_h$ and the game continues until all variables have been assigned.
Let us consider $\Phi = (x_1 \wedge (x_3 \vee \overline{x_4})) \wedge (x_2 \wedge (\overline{x_3} \vee x_4))$ as a partizan QSAT default game, where $x_1 \cup x_2 = X_1$ and $x_3 \cup x_4 = X_2$. This means that x_1, x_2 are colored *red* and x_3,x_4 are colored *blue*. Since it is the normal QSAT game, we have 2 players, player 1 tries to fulfill Φ, and player 2 tries to prevent player 1 from doing so. Player 1 starts the game and has to give a truth assignment for x_1 or x_2. We specify a winning strategy for player 1 which would be to set x_1 to true. Now player 2 must specify a truth assignment for x_3 or x_4. However, player 2 can not set x_3 to true or x_4 to false, since this would satisfy the boolean formula. However, if player 2 sets x_3 to false or x_4 to true, then player 1 can set x_2 to true, and thus the boolean formula is satisfied and player 1 wins the game.

2.4.3 Default game

QSAT can be thought of as a two-player game, as mentioned above. We consider player 1 to be an existential quantifier and player 2 to be a universal quantifier, meaning that each variable player 1 assigns is one to which an existential quantifier is prefixed, while each variable player 2 assigns is one to which a universal quantifier is prefixed. Once each variable has been assigned, the game is over. Player 1 wins the game if and only if the formula is satisfied (at the end).

2.4.3.1 Seek game

In the seek game we also have a QSAT instance, but we change the goal that both players pursue. In this case, both players want to satisfy the formula, but they both want to do it first. The first player to complete the formula wins, regardless of who has completed the other variables that helped complete the formula. The variables that have not yet been assigned are evaluated as *false*. This means that for every occurrence of a negated unassigned variable $\overline{x_i}$, it is interpreted as true, and for every occurrence of a non-negated unassigned variable x_j, it is interpreted as false. The game is considered a draw if, after the final variable has been assigned by a player, the boolean formula is still not fulfilled.

Let us consider $\Phi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 (x_1 \wedge x_3) \vee (x_2 \wedge x_4)$ as a seek QSAT game. Player 1 starts the game and chooses a truth constraint for x_1. If player 1 sets x_1 to true, then he can win the game, because player 2 cannot yet satisfy the boolean formula with his move. That means no matter how player 2 sets x_2, the boolean formula remains unsatisfied. Player 1 can fulfill the boolean formula by setting x_3 to true. Thus we have found a winning strategy for player 1.

2.4.3.2 Avoid game

In the avoid game, it is the other way around as in the seek game, since here both players avoid the formula from being satisfied. Thus, the player who satisfies the formula first loses. Here again, it is unimportant how the variables were previously assigned. Only the player who first fulfills the formula with the move he makes loses. The variables that have not yet been assigned are evaluated as *false*. This means that for every occurrence of a negated unassigned variable $\overline{x_i}$, it is interpreted as true, and for every occurrence of a non-negated unassigned variable x_j, it is interpreted as false. If after the last variable is assigned by a player the boolean formula is still not fulfilled, then the game is considered a draw.
Let us consider $\Phi = \exists x_1 \forall x_2 : (x_2 \vee \overline{x_2}) \wedge x_1$ as an avoid QSAT game. Player 1 starts the game and chooses a truth assignment for x_1. If player 1 sets x_1 to true, then he wins the game. This is because no matter how player 2 assigns x_2, $x_2 \vee \overline{x_2}$ evaluates to true. Thus, we have found a winning strategy for player 1.

3. Avoid Oriented Cycle Game

This chapter will focus on the *avoid oriented cycle game* in more detail. We shall always use oriented edge groups to look at the game as a whole. The question is whether the current player has a winning strategy with given oriented edge groups M and a mixed graph G. The game is won by the player who first forces the opponent to create a directed cycle or to put it in other words, the player who first creates a directed cycle loses. Whoever orients the other edges in this directed cycle is irrelevant. To put it another way, if player 1 simply orients the directed cycle's last edge, he also loses. This chapter will start by investigating an upper bound on the true difficulty of the *avoid oriented cycle game* independent of M. After that, we shall make assumptions about M and determine if the game becomes simpler or remains unchanged under these assumptions. We are curious as to whether the game can actually be made simpler or even trivial by the features of M.

3.1 Avoid Oriented Cycle Game in PSPACE

We take the standard *avoid oriented cycle game*, in which the first person to close a directed cycle loses, and provide an algorithm that is independent of the properties of the oriented edge groups M, in order to demonstrate that the question of whether the current player has a winning strategy in the *avoid oriented cycle game* lies in PSPACE. With the help of this proof, we can be certain that every instance of the *avoid oriented cycle game* can be solved in polynomial space independently of the oriented edge groups M. Due to the fact that we only have an upper bound, we still do not know whether the game is solvable in polynomial time.

Theorem 3.1. *Avoid oriented cycle game lies in PSPACE.*

Proof. We need to find an algorithm that determines in polynomial space whether the current player has a winning strategy on a graph G with oriented edge groups M in order to demonstrate that the *avoid oriented cycle game* is in PSPACE. We define an algorithm that does just that to accomplish this. To achieve this, we will only compute the input graph by directing its edges. We can avoid allocating additional memory in this manner. We must now think of every scenario in which the game might be played. We analyze the game tree and identify a path where the moves of the opponent are irrelevant for all moves of the opponent. To test every conceivable game variant, we systematically work our way through the game tree. We first check an undirected edge e with its corresponding oriented

Algorithm 3.1: canCurrentPlayerAvoid

Input: Mixed Graph $G = (V, E, A)$, $M \subseteq 2^{\overrightarrow{E}}$
Output: Whether the current Player can avoid creating a directed cycle

```
1  forall e ∈ E do
       // direct edge e randomly with the corresponding oriented edge
          group, if there is no orientation for edge e skip iteration
2      if directEdge (e, G, M) then
3          continue with next iteration
4      if ∃ directed cycle or canCurrentPlayerAvoid (G, M) then
           // direct edge e and the corresponding oriented cycle group
              the other way round, if the other orientation is invalid
              continue with next iteration
5          if reverseEdge (e, G, M) then
6              continue with next iteration
7          if ∃ directed cycle or canCurrentPlayerAvoid (G, M) then
8              resetEdge (e, G, M) // undirect edge e and the corresponding
                  oriented cycle group
9              continue with next iteration
10     resetEdge (e, G, M)
11     return true
12 return false
```

edge group m in the graph G, to see if it does not create a directed cycle and can force a victory. To put it another way, we orient this undirected edge e with its corresponding oriented edge group m and make sure we win against every edge that our opponent plays. To achieve this, we simulate every possible move of our opponent and check again if we have a winning strategy. It is crucial that we only work on the local graph, and if we find a solution in a subtree, we clean up the graph and undirect the edges that were directed during this move. We can use Algorithm 3.1 that satisfies our needs for this. This algorithm makes the *avoid oriented cycle game* in PSPACE possible.

□

3.2 Avoid Oriented Cycle Game is PSPACE-hard

This section will focus on PSPACE-hardness proofs. We always look at the *avoid oriented cycle game* together with the oriented edge groups M and give certain properties to M to show that the *avoid oriented cycle game* with these properties of M is PSPACE-hard. Finally, we will use Theorem 3.1 to show that these game variations are PSPACE-complete.

For our first PSPACE-hardness proof, we use a special version of our game. We start with the standard *avoid oriented cycle game*, in which the player who closes a directed cycle first loses, but we demand special properties from our oriented cycle groups M. For these we demand the following properties: M is *perfectly complete*, *symmetrical* and *max* $M \coloneqq max\{|m| \colon m \in M\} = 3$.

Theorem 3.2. *Avoid oriented cycle game is PSPACE-hard, even if M is* perfectly complete*,* symmetrical *and max* $M \coloneqq max\{|m| \colon m \in M\} = 3$.

Proof. We provide a reduction to demonstrate that our game is PSPACE-hard:

$$\textit{Impartial avoid positive 2-DNF SAT} \leq_p \textit{avoid oriented cycle game}$$

We are given a Φ instance of the *Impartial avoid positive 2-DNF SAT*. We call the set of variables used in Φ X. The formula used in Φ is a dis-junction of clauses $C_1 \vee \cdots \vee C_k$, where each clause C_j is a conjunction of two positive variables, i.e. $C_k = x_i \wedge x_j$. In addition, each player must choose a variable that has not been chosen before and set it to true. In the individual clauses, variables can only occur not negated. The goal of player A and player B respectively is to avoid satisfying the formula first.
Next, we create a mixed graph $G = (V, E, A)$ with a set of the oriented edge groups M that has a winning strategy for player 1 for (G, M) if and only if player A has a winning strategy for Φ. The steps for the construction of G, M are as follows:

1. Create four layers in which the nodes of G come in.
2. Make a start and end node; from the end node to the start node, there is a directed edge. Put the start node in layer 1 and the end node in layer 4.
3. For each variable $x_i \in X$ we introduce five new nodes $\{w_i, w_i', t_i, y_i, y_i'\}$. Here, we create three undirected edges $[w_i, w_i'], [w_i, t_i]$, and $[y_i, y_i']$. In addition, we create one directed edge (t_i, w_i'). Put the nodes $\{w_i, w_i', t_i\}$ in layer 2 and $\{y_i, y_i'\}$ in layer 3.
4. For each clause C_k with $C_k = x_i \wedge x_j$ there are three directed edges. These connect the start node to w_i, w_i' to y_j, and y_j' to the end node.
5. For the set of oriented edge groups M, we put the three undirected edges $[w_i, w_i'], [w_i, t_i]$, and $[y_i, y_i']$ into one symmetrical edge group. To do this, we will first define two subsets M_T and M_F, with T for *true*, F for *false* and $M_T \dot{\cup} M_F = M$. We add $m_i = \{(w_i, w_i'), (t_i, w_i), (y_i, y_i')\}$ to M_T and $m_i' = \{(w_i', w_i), (w_i, t_i), (y_i', y_i)\}$ to M_F.

The construction of a mixed graph G with the oriented edge set M can be seen in Fig. 3.1 as an example.

Since each variable and clause requires a constant number of nodes and edges, the transformation takes place in polynomial space. Notice that by construction our oriented edge group M has the desired properties. The first thing we notice is that orienting an undirected edge is the same as selecting a particular oriented edge group because when orienting an undirected edge, the entire group is oriented and since the oriented edge groups are perfectly complete, this means that each oriented edge is in exactly one oriented edge group. Now, however, we can see that if a player selects an oriented edge group $m_i' \in M_F$, he loses directly. This is because by choosing the oriented edge group m_i' the player closes the directed cycle $t_i \to w_i' \to w_i$. Since each player can always choose an oriented edge group from M_T, we say, without limiting the generality, that if the player has to choose an oriented edge group, he will always choose one from M_T.
It is still necessary to demonstrate that each instance of *impartial avoid positive 2-DNF SAT* can be converted into an equivalent *avoid oriented cycle game* instance by constructing the graph G and the oriented edge groups M as previously mentioned where an optimal strategy for player A of the *impartial avoid positive 2-DNF SAT* game is equivalent to an optimal strategy for player 1 of the *avoid oriented cycle game*, and an optimal strategy for player B of the *impartial avoid positive 2-DNF SAT* game is equivalent to an optimal strategy for player 2 of the *avoid oriented cycle game*. We will demonstrate this by showing:

$$\text{choosing } m_i \in M_T \Leftrightarrow \text{ choosing } x_i \in X$$

First, we demonstrate that choosing $m_i \in M_T \Leftarrow$ choosing $x_i \in X$. We assume that a player has chosen the variable $x_i \in X$ in the SAT game. In the *avoid oriented cycle game*,

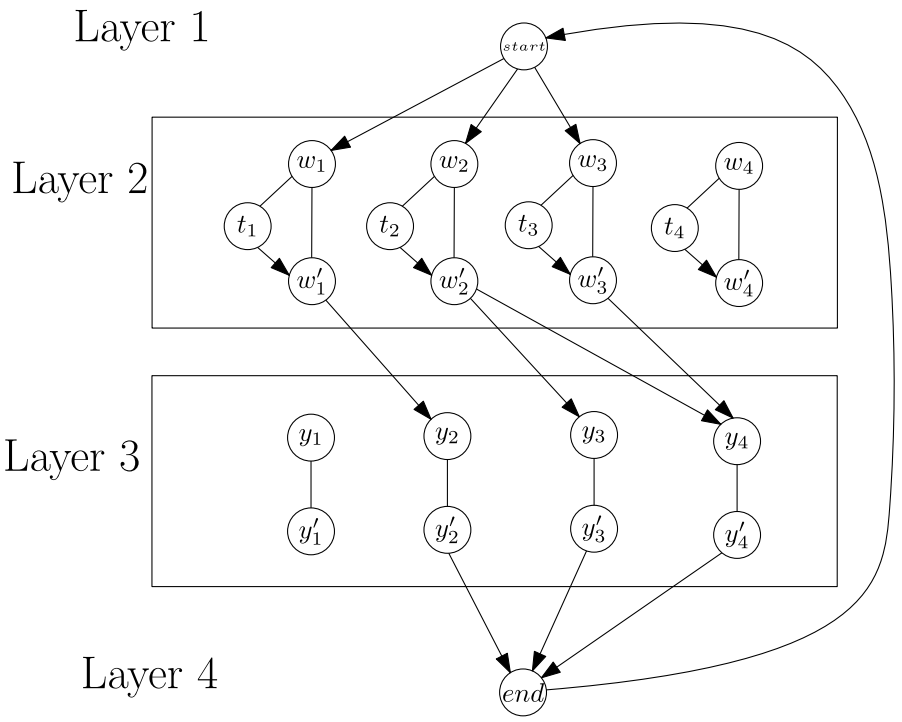

Figure 3.1: Resulting graph G, for the transformation of an *Impartial avoid positive 2-DNF SAT* instance with the boolean formula: $(x_1 \wedge x_2) \vee (x_2 \wedge x_3) \vee (x_2 \wedge x_4) \vee (x_3 \wedge x_4)$

choosing the oriented edge group $m_i \in M_T$ would be the appropriate equivalent move. As a result of choosing m_i, there are three oriented edges (w_i, w'_i), (y_i, y'_i), and (t_i, w_i). Selecting x_i results in Φ being either fulfilled or not. It is enough for us to look at what happens when choosing x_i has led to Φ being fulfilled. For Φ to be satisfied, a clause must be fulfilled. Let C_a be this clause. Let $X' \subseteq X$ also be the set of all variables that have been selected so far. Now we have to show that there is a directed cycle in G with the oriented edge groups M' corresponding to X'. Since Φ was not fulfilled until x_i was selected, x_i must be included in the clause C_a. Thus C_a has the form $C_a = x_i \wedge x_j$ with $x_j \in X'$. That is, x_i and x_j were selected in the course of the game. The equivalent moves for the *avoid oriented cycle game* would be the moves m_i and m_j. By selecting the oriented edge groups m_i and m_j, there are directed edges $(w_i, w'_i), (t_i, w_i), (y_i, y'_i), (w_j, w'_j), (t_j, w_j)$, and (y_j, y'_j) in the graph. Since C_a is a clause, the directed edges $(startnode, w_i), (w'_i, y_j)$, and $(y'_j, endnode)$ exist in the *avoid oriented cycle game*. Together with the directed edge $(endnode, startnode)$, the previously mentioned edges form a directed cycle. The directed cycle must therefore have the following form:

$$startnode \to w_i \to w'_i \to y_j \to y'_j \to endnode$$

We now show choosing $m_i \in M_T \Rightarrow$ choosing $x_i \in X$.
We assume that a player in the *avoid oriented cycle game* chooses an oriented edge group $m_i \in M_T$. In the SAT game, choosing x_i is the appropriate equivalent move. As a result of choosing m_i, there are three oriented edges (w_i, w'_i), (y_i, y'_i), and (t_i, w_i). There are also two possible results of selecting m_i, either G has a directed cycle by the selection or not. It is enough for us to take a look at what happens when the choice of an oriented edge group m_i has led to a directed cycle in G. To accomplish this, we will identify which directed cycles can exist in the graph G.
Let $M' \subseteq M_T$ be the set of all oriented edge groups that have already been selected. Now we have to show that Φ is fulfilled by X' corresponding to M'. Notice that graph G with the various layers has both directed and undirected edges within the layers, and that only

directed edges lead from a lower layer to a higher layer, with the exception of the directed edge from the end node to the start node. There can be no directed cycle within a layer because layers 1 and 4 have only one node and no loop, layer two only consists of several triangles, and can only form a directed cycle if a move $m'_i \in M_F$ is played, and layer three consists of nodes that have exactly one undirected edge. Thus, a directed cycle can only be created if it has nodes from each layer, and since layers 1 and 4 only have one node, they are included in the directed cycle. So, the directed cycle must have the following shape:

$$startnode \to \cdots \to endnode$$

Since there can only be edges from the starting node to the nodes $\{w_1, \ldots, w_n\}$, a $w_j \in \{w_1, \ldots, w_n\}$ must also be contained in the directed cycle. The only way to go up another layer is through node w'_j, as this can have the only directed edge to a higher layer. Thus the oriented edge group m_j must have been chosen and the directed edge (w_j, w'_j) is in the mixed graph G. Thus w'_j must also be contained in the directed cycle. So the directed cycle must look like this:

$$startnode \to w_j \to w'_j \to \cdots \to endnode$$

There must also be a node $y_l \in \{y_1, \ldots, y_n\}$ in the directed cycle since the node w'_i can only have directed edges to the nodes $\{y_1, \ldots, y_n\}$. Since y_l only has one undirected edge $[y_j, y'_j]$, the oriented edge group m_l must be chosen. There is only one additional edge that the y'_l can have, and it is a directed edge that connects the y'_l to the end node. This leaves us with the directed cycle:

$$startnode \to w_j \to w'_j \to y_l \to y'_l \to endnode$$

In order for this directed cycle to be created, all three directed edges $(startnode, w_j)$, (w'_j, y_l) and $(y'_l, endnode)$ must first be present. This can only be the case if the clause $C = x_j \wedge x_l$ exists in the SAT game. At the same time, the undirected edges $[w_j, w'_j]$ and $[y_l, y'_l]$ must have been oriented from w_j and y_l to w'_j and y'_l. This can only be the case if $m_j, m_l \in M'$. However, this orientation corresponds to selecting and assigning the variables x_i and x_j to true.
Since the move, m_i created a directed cycle, $m_i \in \{m_j, m_l\}$ and the other move $\{m_j, m_l\} \backslash m_i$ necessary for the directed cycle were made by a player before. However, since this directed cycle was created, the equivalent move x_i would satisfy clause C and therefore satisfy Φ. Thus x_i satisfies Φ if m_i creates a directed cycle in G. □

Now that we have seen our first PSPACE-hardness proof, we can ask ourselves if we can lower our requirements for the oriented edge groups even further. Since our M was already *perfectly complete* and *symmetric*, these properties were already the best possible. However, we could still do something about the size of the respective groups. We perform another PSPACE-hard proof, but this time we change the requirements for the oriented edge groups so that we have the smallest possible individual groups. To do this, we again take the standard *avoid oriented cycle game* and this time we demand from the oriented edge groups M: M is *unique* and $max\ M \coloneqq max\{|m| \colon m \in M\} = 2$. So with the following proof we manage to lower the respective oriented edge group size to 2, but we lose the properties that M is *symmetric* and *perfectly complete*.

Theorem 3.3. *Avoid oriented cycle game is PSPACE-hard, even if M is* unique *and $max\ M \coloneqq max\{|m| \colon m \in M\} = 2$.*

Proof. We provide a reduction to demonstrate that the *avoid oriented cycle game* with these special properties is PSPACE-hard:

$$\textit{Impartial avoid positive 2-DNF SAT} \leq_p \textit{avoid oriented cycle game}$$

We are given a Φ instance of the *Impartial avoid positive 2-DNF SAT*. We define X as the set of variables used in Φ. The formula used in Φ is a dis-junction of clauses $C_1 \vee \cdots \vee C_k$, where each clause C_j is a conjunction of two positive variables, i.e. $C_k = x_i \wedge x_j$. In addition, each player must choose a variable that has not been chosen before and set it to true. In the individual clauses, variables can only occur not negated. The goal of player A and player B respectively is to avoid satisfying the formula first.
In the following step, we construct a mixed graph $G = (V, E, A)$ with a set of oriented edge groups M that has a winning strategy for player 1 for (G, M) if and only if player A has a winning strategy for. Here we will construct the set of oriented edge groups M such that M is *unique* and *max* $M = 2$. The steps for the construction of G, M are as follows:

1. Create four layers in which the nodes of G come in.
2. Make a start and end node; from the end node to the start node, there is a directed edge. Put the start node in layer 1 and the end node in layer 4.
3. For each variable $x_i \in X$ we introduce four new nodes $\{w_i, w_i', y_i, y_i'\}$. Here, we create two undirected edges $[w_i, w_i']$, and $[y_i, y_i']$. Put the nodes $\{w_i, w_i'\}$ in layer 2 and $\{y_i, y_i'\}$ in layer 3.
4. For each clause C_k with $C_k = x_i \wedge x_j$ there are three directed edges. These connect the start node to w_i, w_i' to y_j, and y_j' to the end node.
5. For the set of oriented edge groups M, we put the two undirected edges $[w_i, w_i']$ and $[y_i, y_i']$ into an edge group. We must put only one orientation into oriented edge groups, namely, we add $m_i = \{(w_i, w_i'), (y_i, y_i')\}$ into the oriented edge groups M. Thus we have prevented an orientation and the undirected edges can only be oriented from w_i, y_i to w_i', y_i'.

The construction of a mixed graph G with the oriented edge set M can be seen in Fig. 3.2 as an example.

Since each variable and clause requires a constant number of nodes and edges, the transformation takes place in polynomial space. Notice that by construction our oriented edge group M has the desired properties. The first thing we notice is that orienting an undirected edge is the same as selecting a particular oriented edge group because when orienting an undirected edge, the entire group is oriented and since the oriented edge groups are unique, this means that each oriented edge is in one or non-oriented edge group. To complete the reduction, we need to show that any instance of *impartial avoid positive 2-DNF SAT* can be converted to an equivalent *avoid oriented cycle game* instance by building the mixed graph G and the oriented edge groups M as mentioned. The optimal strategy for player A on Φ of the *impartial avoid positive 2-DNF SAT* must be equivalent to an optimal strategy for player 1 on (G, M) of the *avoid oriented cycle game*. The first thing we notice is that M and the M_T from Theorem 3.2 are almost equal to each other. The only difference is that M_T contains one edge (t_i, w_i) more in each oriented edge group. However, in the proof that $m_i \in M_T \Leftrightarrow x_i \in X$, this edge (t_i, w_i) was never discussed and is therefore irrelevant. Since this one more edge was never used and otherwise $M = M_T$ is true, the proof that $m_i \in M \Leftrightarrow x_i \in X$ is the same as in Theorem 3.2. □

With this proof, we have seen that the *avoid oriented cycle game* is also PSPACE-hard for smaller groups. We have now seen two versions of our oriented edge groups that are

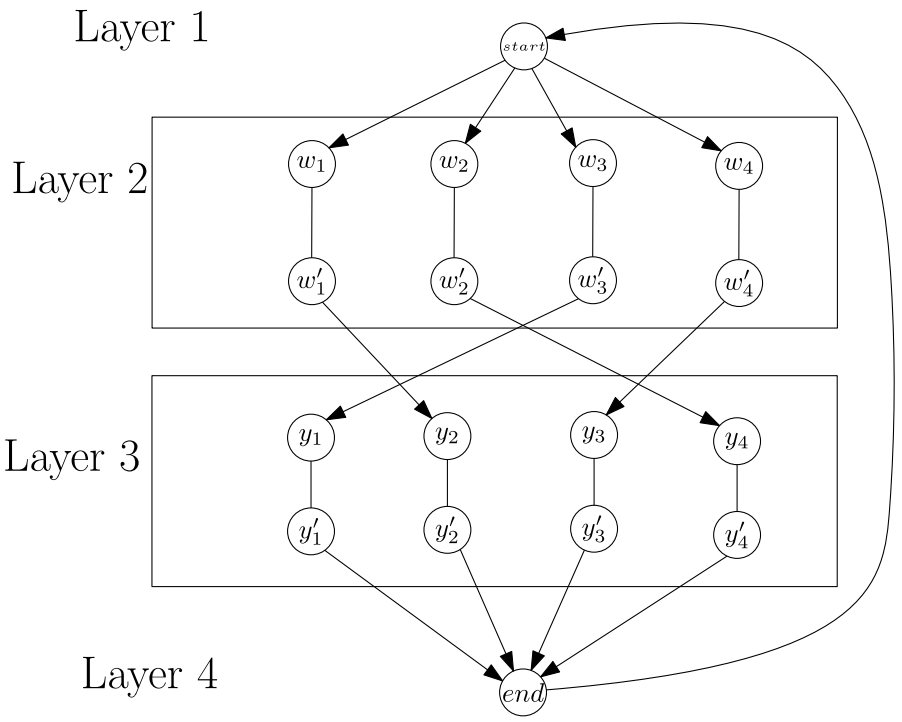

Figure 3.2: Resulting graph G, for the transformation of an *Impartial avoid positive 2-DNF SAT* instance with the boolean formula: $(x_1 \wedge x_2) \vee (x_3 \wedge x_1) \vee (x_2 \wedge x_4) \vee (x_4 \wedge x_3)$

PSPACE-hard together with the *avoid oriented cycle game*, one when M is *symmetric* and *perfectly complete*, but the individual groups each have cardinality 3, and the other when M is *unique*, but the individual groups each have cardinality 2. As already described in the introduction 1.3, we can use the two proofs of PSPACE-hardness 3.3 and 3.2 and the proof that the *avoid oriented cycle game* lies in PSPACE 3.1 to show that the *avoid oriented cycle* game with the oriented edge groups is PSPACE-complete.

Theorem 3.4. *Avoid oriented cycle game with oriented edge groups is PSPACE-complete.*

Proof. The *avoid oriented cycle game* is in PSPACE, as we have demonstrated in Theorem 3.1. The oriented cycle groups M did not have any restrictions placed on them in the proof so the properties of M are insignificant for the proof and M can be arbitrary. Our *avoid oriented cycle game* is PSPACE-hard for specific requirements on M, as demonstrated in Theorem 3.3 or in Theorem 3.2. A problem is considered to be PSPACE-hard if it is at least as challenging to solve as all other problems in the class. For this, it must be true that it lies in PSPACE and it must also be PSPACE-hard. The *avoid oriented cycle game* is PSPACE-complete, even if M is *perfectly complete*, *symmetrical* and $max\ M \coloneqq max\{|m| \colon m \in M\} = 3$ or M is *unique* and $max\ M \coloneqq max\{|m| \colon m \in M\} = 2$ as shown by the theorems taken together. This makes it one of the most difficult problems of the class PSPACE. □

3.3 Avoid Oriented Cycle Game in P

Now that we have spent a lot of time finding out when the *avoid oriented cycle game* is particularly difficult to solve, we will attempt to find out when the *avoid oriented cycle game* is easy to solve. In this case, we look at the normal *avoid oriented cycle game* without oriented edge groups and ask ourselves how difficult it is to solve. That is, we look at the case where the two players take turns orienting exactly one undirected edge, and the

player who first creates a directed cycle loses. With Theorem 3.5, we will show that if both players play optimally, it will always end in a draw and that it is possible to find the best possible move for the current player in polynomial time.

Theorem 3.5. *Avoid oriented cycle game* $\in P$.

Proof. We prove that the *avoid orient cycle game* lies in P by specifying an algorithm that terminates in polynomial time. For the algorithm, we first have to prove that there cannot be a mixed graph G where a player is forced to make a move that directly closes an orientated cycle and thus loses the game. We assume that player 1 is forced to orient an undirected edge in the mixed graph G, which directly closes a directed cycle. Thus, the mixed graph G on which the players orient the undirected edges must consist of directed cycles, all of which have exactly one undirected edge, so that player 1 orients this undirected edge and completes the directed cycle. Let e be the undirected edge that player 1 orients (from b_1 to a_1) to complete the directed cycle C_1 as shown in Fig. 3.3.

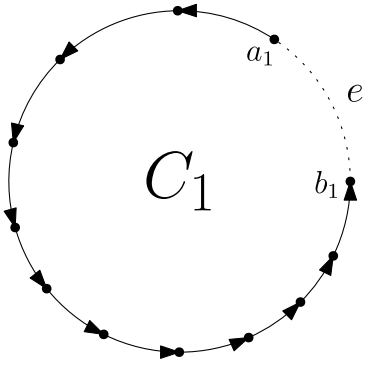

Figure 3.3: One directed cycles with 1 undirected edge each

Now the question arises here, why did player 1 not orient the edge e the other way around (from a_1 to b_1)? This orientation would prevent the directed cycle C_1. Since our assumption is that player 1 is forced to orient an undirected edge that loses, there must be another directed cycle C_2 that includes e and closes the directed cycle with the orientation of e (from a_1 to b_1) as shown in Fig. 3.4.

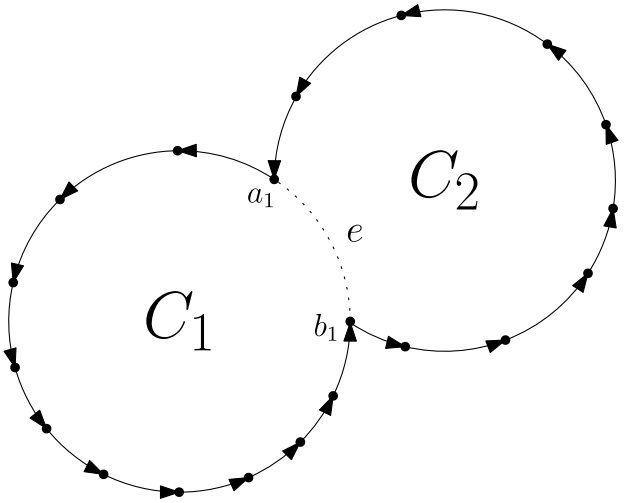

Figure 3.4: Two directed cycles with 1 undirected edge each and a directed cycle

However, we can see that C_1 together with C_2 already close a directed cycle. This is against our assumption that the game is not yet over and that player 1 is forced to make a move

that directly loses.
Thus our assumption is wrong and we have seen that there can be no mixed graph G in which a player has no possible move that does not lose directly since trying to create such a mixed graph G leads to a graph that already has a directed cycle.
This implies that a mixed graph G in which a player loses at some point is not possible, because if there were such a mixed graph G, the players would eventually reach a mixed graph G' in the game where one of them would have to lose directly. However, since we have already shown that a G' like this one does not exist, G also does not exist.
Since both players cannot force the other to close a directed cycle, it is sufficient to always make a move that does not close a directed cycle. This is exactly how the algorithm works, which is supposed to find the optimal move in polynomial time. The algorithm works as follows:

1. Find an orientation of an undirected edge so that no directed cycle is closed in $\mathcal{O}(|E| * (|V| + |E|))$.

□

This concludes the chapter *avoid oriented cycle game*, where we have learned which variations of the game are PSPACE-complete and also looked at which variation of the game is solvable in polynomial time.

4. Seek Oriented Cycle Game

The *seek oriented cycle game* game will be discussed in more detail in this chapter. We shall always use oriented edge groups to look at the game as a whole. The question is whether the current player, given oriented edge groups M and a mixed graph G, has a winning strategy. The player who makes a directed cycle first wins the game. Whoever orients the other edges in this directed cycle is irrelevant. To put it another way, player 1 also wins if he simply orients the directed cycle's final edge. This chapter will start by investigating an upper bound on the true difficulty of the *seek oriented cycle game* independent of M. Following that, we'll make assumptions about M and see if the game gets easier or stays the same under them. We want to know if the features of M can actually make the game easier or even trivial.

4.1 Seek Oriented Cycle Game in PSPACE

We consider our normal *seek oriented cycle game*, where the first player to close a directed cycle wins, and give an algorithm that is independent of the properties of the oriented edges groups M so that we show that the question of whether the current player has a winning strategy in the *seek oriented cycle game* lies in PSPACE. With the help of this algorithm, we can be sure that every instance of the *seek oriented cycle game* is solvable with polynomial space, also independent of the properties of our oriented edge groups M. However, it may still be that there is a faster algorithm that may even terminate in polynomial time. This algorithm thus acts as an upper bound and we know that polynomial space is sufficient to solve the *seek oriented cycle game*.

Theorem 4.1. *Seek oriented cycle game lies in PSPACE.*

Proof. To show that the *avoid oriented cycle game* is in PSPACE, we need to find an algorithm that determines in polynomial space whether the current player has a winning strategy on a graph G with oriented edge groups M. To achieve this, we define an algorithm that does just that. We will only compute the input graph by directing its edges in order to accomplish this. In this way, we can avoid allocating more memory. Now we must consider every scenario that could occur during the game. We analyze the game tree and locate a path where no opponent move is relevant for any opponent move. We systematically traverse the game tree, testing every conceivable game variant. We first determine whether an undirected edge e with its corresponding oriented cycle groups m in the graph G

Algorithm 4.1: canCurrentPlayerSeek

Input: Mixed Graph $G = (V, E, A)$, $M \subseteq 2^{\overrightarrow{E}}$
Output: Whether the current Player can create a directed cycle first

```
1  forall e ∈ E do
       // direct edge e randomly with the corresponding oriented edge
          group, if there is no orientation for edge e skip iteration
2      if directEdge (e, G, M) then
3          continue with next iteration
4      if ∃ directed cycle then
           // undirect edge e and the corresponding oriented cycle group
5          resetEdge (e, G, M)
6          return true
7      if canCurrentPlayerSeek (G, M) then
           // direct edge e and the corresponding oriented cycle group
              the other way round, if the other orientation is invalid
              continue with next iteration
8          if reverseEdge (e, G, M) then
9              continue with next iteration
10         if canCurrentPlayerSeek (G, M) then
11             resetEdge (e, G, M)
12             continue with next iteration
13     resetEdge (e, G, M)
14     return true
15 return false
```

can force a victory. To put it another way, we orient this undirected edge e with its corresponding oriented edge group m and make sure we win against every edge that our opponent plays. To do this, we simulate every move that could be made by our opponent and then check again whether we have a winning plan. It is crucial that we only focus on the local graph. If we discover a solution in a sub-tree, we should clean up the graph and undirect any edges that were directed as a result of this move. We can use Algorithm 4.1 that satisfies our needs for this. This algorithm makes the *avoid oriented cycle game* in PSPACE possible. □

4.2 Seek Oriented Cycle Game is PSPACE-hard

This section will focus on PSPACE-hardness proofs for the *seek oriented cycle game* with oriented edge groups. That is, in this section we always assume that we have the normal *seek oriented cycle game* but we give special properties to the oriented edge groups M and show that despite these special properties of M, the game is PSPACE-hard. Finally, we will show with the help of Theorem 4.1 that these game variations are also PSPACE-complete.

For our first PSPACE-hardness proof, we take a special version of our game. We start with the standard *seek oriented cycle game*, where the player who closes a directed cycle first wins, but we expect special properties from the oriented edge groups M. For the oriented edge groups M we demand, that M is *perfectly complete* and we define $max\ M \coloneqq max\{|m| \colon m \in M\}$.

For a 3-DNF SAT formula Φ with the clauses $\{C_1, \ldots, C_n\}$ we define $t(\Phi)$, where $t(\Phi) \in \mathbb{N}$ is the number of times a variable can be at most in the middle of the clauses $\{C_1, \ldots, C_n\}$.

Since the clauses are conjunctions of three literals, the order of the literals can be reversed within the clauses. We look at the permutation where $t(\Phi)$ is minimal.

Theorem 4.2. *Seek oriented cycle game is PSPACE-hard even if M is* perfectly complete. *Moreover, if impartial seek positive 3-DNF SAT is PSPACE-hard with $t(\Phi) = t^*$, $t^* \in \mathbb{N}$, then seek oriented cycle game is PSPACE-hard for max $M \leq t^* + 3$*

Proof. To prove that the game is PSPACE-hard, we provide a reduction:

Impartial seek positive 3-DNF SAT $\leq_p$ seek oriented cycle game.

We are given an instance Φ of the *Impartial seek positive 3-DNF SAT*. We call the set of variables used in Φ X. The formula used in Φ is a dis-junction of clauses $C_1 \vee \cdots \vee C_n$, where each clause is a conjunction of three positive literals, i.e. $C_k = x_i \wedge x_j \wedge x_l$. The number $t(\Phi) \in \mathbb{N}$ is the number of times a variable can be at most in the middle of the clauses $\{C_1, \ldots, C_k\}$. In addition, no clause has a negated literal and both players take turns choosing a literal and setting it to true. The aim of both players A and B is to be the first to fulfill the boolean formula.
Next, we create the mixed graph G together with the oriented edge groups M depending on Φ, where player 1 wins the *seek oriented cycle game* on (G, M) if and only if player A of the *impartial seek positive 3-DNF SAT* wins on Φ. The steps to create G, M are as follows:

1. Create six layers in which the nodes of G come in.
2. Make a start and end node; from the end node to the start node, there is a directed edge. Put the start node in layer 1 and the end node in layer 5.
3. For each variable $x_i \in X$ we introduce seven new nodes $\{w_i, w_i', z_i, z_i', t_i, t_i', t_i''\}$. Here, we create six undirected edges $[w_i, w_i']$, $[z_i, z_i']$, $[t_i, t_i']$, $[t_i, t_i'']$, and $[t_i', t_i'']$. Put the nodes $\{w_i, w_i'\}$ in layer 2, $\{z_i, z_i'\}$ in layer 4 and $\{t_i, t_i', t_i''\}$ in layer 6.
4. For each clause C_k with $C_k = x_i \wedge x_j \wedge x_l$ there are two nodes $\{y_j^k, y_j^{k\prime}\}$ in layer 3. There are also four directed edges. We add $(start, w_i)$, (w_i', y_j^k), $(y_j^{k\prime}, z_l)$, and (z_l', end). We also add $[y_j^k, y_j^{k\prime}]$ to G.
5. For the set of oriented edge groups M, we will first define two subsets M_T and M_F, with T for *true*, F for *false* and $M_T \dot\cup M_F = M$.
   We add $m_i^1 = \{(w_i, w_i')\} \cup \{(y_j^k, y_j^{k\prime}) | x_i \text{ middle variable of clause } C_k\} \cup \{(z_i, z_i')\}$ to M_T and we add $m_i^2 = \{(w_i', w_i)\} \cup \{(y_j^{k\prime}, y_j^k) | x_i \text{ middle variable of clause } C_k\} \cup \{(z_i', z_i), (t_i', t_i)\}$, $m_i^3 = \{(t_i, t_i''), (t_i'', t_i')\}$, $m_i^4 = \{(t_i, t_i')\}$, and $m_i^5 = \{(t_i', t_i''), (t_i'', t_i)\}$ to M_F.

The construction of a mixed graph G with the oriented edge set M can be seen in Fig. 4.1 as an example.

Since each variable and clause requires a constant number of nodes and edges, the transformation takes place in polynomial space.
Notice that by construction our oriented edge group M has the desired properties because each oriented edge is in exactly one oriented edge group and because m_i^2 contains the most oriented edges and has three oriented edges from layers 2,4, and 6 and also one oriented edge for each occurrence in the middle of the clause. This makes it $t(\Phi) + 3$ edges.
The first thing we notice is that orienting an undirected edge is the same as selecting a particular oriented edge group because when orienting an undirected edge, the entire group is oriented and since the oriented edge groups are perfectly complete, this means that each oriented edge is in exactly one oriented edge group. Now, however, we can see that if a

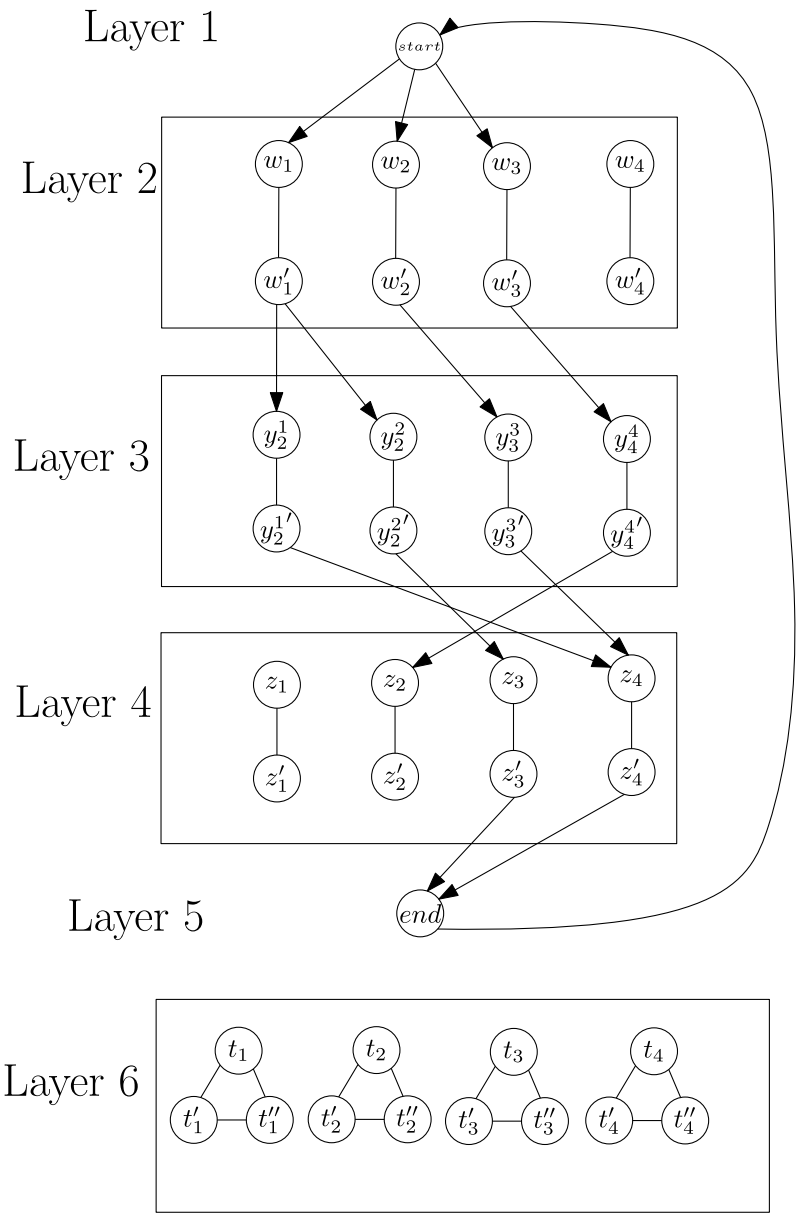

Figure 4.1: The resulting graph, where the boolean formula used in Φ is $(x_1 \wedge x_2 \wedge x_4) \vee (x_1 \wedge x_2 \wedge x_3) \vee (x_2 \wedge x_3 \wedge x_4) \vee (x_3 \wedge x_4 \wedge x_2)$

player selects an oriented edge group $m_i^a \in M_F$, with $a \in \{2,3,4,5\}$, he loses directly. This is because by choosing the oriented edge group m_i^a the other player can close the directed cycle $t_i \to t'_i \to t''_i$ by choosing a $m_i^b \in M_F$, with $b \in \{2,3,4,5\}$. Assuming a player chooses m_i^2 or m_i^4, an appropriate response to create a directed cycle would be to choose m_i^3 or m_i^5 respectively.
Since each player can always choose an oriented edge group from M_T, we say, without limiting the generality, that if the player has to choose an oriented edge group, he will always choose one from M_T.
It is still necessary to demonstrate that each instance of *impartial seek positive 3-DNF SAT* can be converted into an equivalent *seek oriented cycle game* instance by constructing the graph G and the oriented edge groups M as previously mentioned where an optimal strategy for player A of the *impartial seek positive 3-DNF SAT* game is equivalent to an optimal strategy for player 1 of the *seek oriented cycle game*, and an optimal strategy

for player B of the *impartial seek positive 2-DNF SAT* game is equivalent to an optimal strategy for player 2 of the *seek oriented cycle game*. We will demonstrate this by showing:

$$\text{choosing } m_i^1 \in M_T \Leftrightarrow \text{ choosing } x_i \in X$$

We start with choosing $m_i^1 \in M_T \Rightarrow$ choosing $x_i \in X$. Three oriented edges result from choosing m_i^1: $(w_i, w_i'), (y_i, y_i')$ and (z_i, z_i'). Selecting m_i^1 could lead to one of two outcomes: either G has a directed cycle as a result of the selection, or it does not. It is enough for us to take a look at what happens when the choice of an oriented edge group m_i^1 has led to a directed cycle in G. To do this, we must first understand which directed cycle can appear in G. We have already created the six layers in the construction of the mixed graph and the property that stands out is that there is no directed cycle within a layer that can happen with a move $m_i^1 \in M_T$. This means that a directed cycle in G must go over several layers. The next observation is that there are only directed edges from a lower layer to a higher layer, the directed edge from the end node to the start node being the exception. For this, there is no incoming edge to layer 6 from another layer and also no outgoing edge to another layer, thus layer 6 is not interesting for the directed cycle. This means that the directed cycle must go over the first five layers. Since layers 1 and 5 each only have one node, they must be present in the directed cycle. Thus, the directed cycle must have the following form:

$$startnode \to \cdots \to endnode$$

Since the start node is only connected to the nodes $\{w_1, \ldots, w_a\}$ with a directed edge, a $w_j \in \{w_1, \ldots, w_n\}$ must be contained in the directed cycle. The only way to go down a layer is through the node w_j', so a player must have chosen m_j^1 at some point, otherwise, the directed edge (w_j,w_j') would not exist. Thus the directional cycle looks as follows:

$$startnode \to w_j \to w_j' \to \cdots \to endnode$$

It may be that w_j' has a directed edge to $y_l^k \in \{y_o^u, \ldots, y_p^n\}$, where y_l^k is in layer 3. From y_l^k there is only one way to get down to layer 4, and that is with $y_l^{k'}$. For this, however, a player must have selected m_l^1. Thus, the directed cycle must look like this:

$$startnode \to w_j \to w_j' \to y_l^k \to y_l^{k'} \to \cdots \to endnode$$

In the fourth layer, only nodes z_b can occur, so they must occur in the directed cycle. The node z_b can also have only one possibility to come down to layer 5, namely with z_b'. This means that m_b must be selected by a player. From z_b' there can be the directed edge to the end node, thus we have identified the directed cycle.

$$startnode \to w_j \to w_j' \to y_l^k \to y_l^{k'} \to z_b \to z_b' \to endnode$$

In order for this directed cycle to be created, all four directed edges $(startnode, w_j)$, (w_j', y_l^k), $(y_l^{k'}, z_b)$ and $(z_b', endnode)$ must first be present. This can only be the case if a clause $C_k = x_j \wedge x_l \wedge x_b$ exists in the SAT game. At the same time, the undirected edges $[w_j, w_j']$, $[y_l^k, y_l^{k'}]$ and $[z_b, z_b']$ must have been oriented from w_j, y_l^k and z_b to w_j', $y_l^{k'}$ and z_b', respectively. This can only be the case if m_j^1, m_l^1, m_k^1 have been chosen by the players. However, this orientation corresponds to selecting and assigning the variables x_j, x_l, and x_b to true.
Since the move m_i created a directed cycle, $m_i \in \{m_j, m_l, m_b\}$ and the other moves $\{m_j, m_l, m_b\} \setminus m_i$ necessary for the directed cycle were made by the players before it. However, since this directed cycle was created, the equivalent move x_i would satisfy clause C and therefore satisfy Φ. Thus x_i satisfies Φ if m_i creates a directed cycle in G.

We now show choosing $m_i^1 \in M_T \Leftarrow$ choosing $x_i \in X$.
Selecting x_i can again have two results: Either Φ is fulfilled or not. It is enough for us to look at what happens when choosing x_i has led to Φ being fulfilled. For Φ to be satisfied, a clause must be fulfilled. Let $C_k = x_i \wedge x_j \wedge x_l$ be this clause. For the clause C_k to be satisfied by selecting x_i, the variables x_j and x_l must have been selected beforehand. Thus, in the *seek oriented cycle game*, m_j^1 and m_l^1 are already selected before m_i^1 is selected. Since C_k is a clause in Φ, there are by construction of G the directed edges $(start, w_i), (w_i', y_j^k), (y_j^{k'}, z_l)$ and (z_l', end). Since the moves m_j^1 and m_l^1 were already played before m_i^1 was played, there are additionally the two directed edges: $(y_j^k, y_j^{k'})$ and (z_l, z_l'). Due to the move m_i^1 there is now at least the edge (w_i, w_i') additionally in the mixed graph G. Since there is additionally the directed edge $(end, start)$, we have found a directed cycle that was created because m_i^1 was selected.

Overall, we have thus shown that:

$$\text{choosing } m_i \in M_T \Leftrightarrow \text{ choosing } x_i \in X$$

□

Since we have now seen that the *seek oriented cycle game* is already PSPACE-hard when M is *perfectly complete*, *symmetrical*, *unique* and *max* $M \leq t(\Phi) + 3$, where $t(\Phi) \in \mathbb{N}$ is the number of times a variable can be at most in the middle of the clauses of *Impartial seek positive 3-DNF SAT* Φ, we ask ourselves: can we keep the requirements on M even lower and thus keep the game PSPACE-hard? The answer is yes, as we will see in Theorem 4.4, but for this, we need auxiliary theorems since it is not clear how to strengthen the reduction from *seek positive 3-DNF-SAT*. To do this, we introduce *impartial seek positive 3-color 3-DNF SAT* and prove that it is PSPACE-hard. The special thing about *impartial seek positive 3-color 3-DNF SAT* is that we also have in the input a coloring for the variables, either F, C, or B standing for Front, Center, Back. Each variable is colored with one color. If a variable x_i is colored with F, it means that there is a permutation of the variables in the clauses so that x_i never appears in front. If a variable x_i has the coloring C, then this means that there is a permutation of the variables in the clauses, so that x_i never appears in the center. And lastly, if a variable x_i has coloring B, then there is a permutation of the variables in the clauses so that x_i never appears at the back. The clauses are already exactly according to this permutation.

Theorem 4.3. *Impartial seek positive 3-color 3-DNF SAT is PSPACE-hard.*

Proof. For the proof, we will take Schäfer's proof [Sch78] for the PSPACE-hardness of *impartial seek positive 3-DNF SAT* as inspiration and modify it slightly so that we can show:

Impartial avoid positive 2-DNF SAT $\leq_p$ *Impartial seek positive 3-color 3-DNF SAT*

We are given an instance Φ of the *Impartial avoid positive 2-DNF SAT*. We call the set of variables used in Φ X. The formula used in Φ is a dis-junction of clauses $C_1 \vee \cdots \vee C_n$, where each clause C_k is a conjunction of two positive variables, i.e. $C_k = x_i \wedge x_j$. In addition, each player must choose a variable that has not been chosen before and set it to true. In the individual clauses, variables only occur not negated. The goal of player A and player B respectively is to avoid satisfying the formula first.
In the following step, we give a procedure that generates an *impartial seek positive 3-color 3-DNF SAT* instance Ψ depending on Φ, where player 1 on Ψ has a winning strategy if and only if player A on Φ has a winning strategy. We call the set of variables used in Ψ X'. The steps for the construction of Ψ are as follows:

1. Create two new variables $x_a, x_b \notin X$ in addition to all the existing variables. $X' = X \cup \{x_a, x_b\}$.
2. For each clause C_k in Φ with $C_k = (x_i \wedge x_j)$ we introduce two new clauses for Ψ, $C'_k = (x_i \wedge x_j \wedge x_a)$ and $C''_k = (x_i \wedge x_j \wedge x_b)$. We assign the color B to x_i and x_j and the color F to x_a and x_b for the color assignment.
3. Create a new clause C_{n+1} with $C_{n+1} = (1 \wedge x_a \wedge x_b)$.

The first thing that stands out is the fact that our choice of clauses directly satisfies the 3-color-ability because, for each variable in Φ, the variables in Ψ appear in the front or in the center, and they have the color B, and the extra introduced variables appear in the center or in the back, and they have the color F.

If we construct our formula as shown above, we will be able to demonstrate how every instance of *Impartial avoid positive 2-DNF SAT* can be converted into an instance of *Impartial seek positive 3-color 3-DNF SAT*, where a strategy for player A is available to avoid Φ if and only if there is a strategy for player 1 to seek Ψ. Here, the optimal strategy is equivalent and can be immediately transferred from one problem to the other.

We first demonstrate that player 1 seeks on Ψ if player A is able to avoid fulfilling Φ. We assume that both players play optimally in the respective games. Player 1's approach is as follows: Copy player A if player 2 selects a variable that already appears in Φ. If player A wins, then player B has satisfied a clause $C_k = x_i \wedge x_j$. Now that a Φ clause has been satisfied, player 1 can simply set x_a to true, winning the game by satisfying $C'_k = x_i \wedge x_j \wedge x_a$. If player 2 selects a newly introduced variable, then it can only be x_a or x_b. As a result of the fact that $C_{n+1} = (1 \wedge x_a \wedge x_b)$ is a clause, player 1 can choose the other variable in this scenario and wins without a contest.

Next, we demonstrate that if player 1 can satisfy Ψ first, player A can avoid fulfilling Φ. First, choosing x_a or x_b by the players on Ψ is illogical because doing so results in an immediate loss of the game because the other player can fulfill the clause C_{n+1}. But because every clause has either x_a or x_b, it follows that player 1 wins the game by choosing x_a or x_b in the last step. By symmetry, we may assume without limiting the generality that player 1 wins by choosing x_a and satisfied $C'_k = x_i \wedge x_j \wedge x_a$. Thus the last move of player 2 was to select x_i or x_j, otherwise, player 2 would have selected x_a himself and won the game. Because $C_k = x_i \wedge x_j$ is a clause in Φ, player B has satisfied C_k by choosing x_i or x_j. As a result, player A has been able to avoid Φ. □

Now that we have a new problem that is provably PSPACE-hard, we can perform a new proof. Again, we take the standard *seek oriented cycle game*, where a player wins if he closes a directed cycle first. This time we also set special properties for our M, but this time more restrictive ones. We expect M to have the following properties, that M is *perfectly complete* and we define $max\ M \coloneqq max\{|m| \colon m \in M\}$.

For a 3-DNF SAT formula Φ with the clauses $\{C_1, \ldots, C_n\}$ we define $t(\Phi)$, where $t(\Phi) \in \mathbb{N}$ is the number of times a variable can be at most in the middle of the clauses $\{C_1, \ldots, C_n\}$. Since the clauses are conjunctions of three literals, the order of the literals can be reversed within the clauses. We look at the permutation where $t(\Phi)$ is minimal.

Theorem 4.4. *Seek oriented cycle game is PSPACE-hard even if M is* perfectly complete. *Moreover, if impartial seek positive 3-color 3-DNF SAT is PSPACE-hard with $t(\Phi) = t^*$, $t^* \in \mathbb{N}$, then seek oriented cycle game is PSPACE-hard for $max\ M \leq t^* + 2$*

Proof. To prove that our game is PSPACE-hard, we provide a reduction:

Impartial seek positive 3-color 3-DNF SAT $\leq_p$ *seek oriented cycle game*

We get an instance Φ of impartial seek positive 3-color 3-DNF SAT. The formula used in Φ is a conjunction of clauses $C_1 \wedge \cdots \wedge C_n$, where each clause is a disjunction of three positive literals, i.e. $C_k = x_i \vee x_j \vee x_l$. The number $t(\Phi) \in \mathbb{N}$ is the number of times a variable can be at most in the middle of the clauses $\{C_1, \ldots, C_k\}$. In addition, no clause has a negated literal and both players take turns choosing a literal and setting it to true. Each variable has been assigned a color, either F, C, or B. There is a permutation of variables within the clauses so that if a variable has the color F, C, or B, it never is in the front, center, or back of the clauses respectively. Let the clauses $C_1 \wedge \cdots \wedge C_n$ already be exactly according to this permutation. The aim of both players A and B is to be the first to fulfill the boolean formula.
Next, we create the mixed graph G together with the oriented edge groups M depending on Φ, where player 1 wins the *seek oriented cycle game* on (G, M) if and only if player A of the impartial *seek positive 3-color 3-DNF SAT* wins on Φ. The steps to create G, M are as follows:

1. Create six layers in which the nodes of G come in.
2. Make a start and end node; from the end node to the start node, there is a directed edge. Put the start node in layer 1 and the end node in layer 5.
3. For each variable $x_i \in X$ we introduce seven nodes, depending on the color of this variable.
   a) If x_i has the color F, then introduce the nodes $\{z_i, z_i', t_i, t_i', t_i''\}$. Put the nodes $n_2 = \{z_i, z_i'\}$ in layer 4 and the nodes $n_3 = \{t_i, t_i', t_i''\}$ in layer 6. We create four undirected edges $[z_i, z_i']$, $[t_i, t_i']$, $[t_i, t_i'']$, and $[t_i', t_i'']$.
   b) If x_i has the color C, then introduce the nodes $\{w_i, w_i', z_i, z_i', t_i, t_i', t_i''\}$. Put the nodes $n_1 = \{w_i, w_i'\}$ in layer 2, the nodes $n_2 = \{z_i, z_i'\}$ in layer 4 and the nodes $n_3 = \{t_i, t_i', t_i''\}$ in layer 6. We create five undirected edges $[w_i, w_i']$, $[z_i, z_i']$, $[t_i, t_i']$, $[t_i, t_i'']$, and $[t_i', t_i'']$.
   c) If x_i has the color B, then introduce the nodes $\{w_i, w_i', t_i, t_i', t_i''\}$. Put the nodes $n_1 = \{w_i, w_i'\}$ in layer 2 and the nodes $n_3 = \{t_i, t_i', t_i''\}$ in layer 6. Here, we create four undirected edges $[w_i, w_i']$, $[t_i, t_i']$, $[t_i, t_i'']$, and $[t_i', t_i'']$.
4. For each clause C_k with $C_k = x_i \wedge x_j \wedge x_l$ there are two nodes $\{y_j^k, y_j^{k'}\}$ in layer 3. There are also four directed edges. We add $(start, w_i)$, (w_i', y_j^k), $(y_j^{k'}, z_l)$, and (z_l', end). We also add $[y_j^k, y_j^{k'}]$ to G.
5. For the set of oriented edge groups M, we will first define two subsets M_T, M_F, with $M_T \dot\cup M_F = M$.
   a) If x_i has the color F, add $m_i^1 = \{(y_j^k, y_j^{k'}) | x_i \text{middle variable of clause} C_k\} \cup \{(z_i, z_i')\}$ to M_T and $m_i^2 = \{(y_j^{k'}, y_j^k) | x_i \text{middle variable of clause} C_k\} \cup \{(z_i', z_i), (t_i', t_i)\}$, $m_i^3 = \{(t_i, t_i''), (t_i'', t_i')\}$, $m_i^4 = \{(t_i, t_i')\}$, and $m_i^5 = \{(t_i', t_i''), (t_i'', t_i)\}$ to M_F.
   b) If x_i has the color C, add $m_i^1 = \{(w_i, w_i'), (z_i, z_i')\}$ to M_T and $m_i^2 = \{(w_i', w_i), (z_i', z_i), (t_i', t_i)\}$, $m_i^3 = \{(t_i, t_i''), (t_i'', t_i')\}$, $m_i^4 = \{(t_i, t_i')\}$, and $m_i^5 = \{(t_i', t_i''), (t_i'', t_i)\}$ to M_F.
   c) If x_i has the color B, add $m_i^1 = \{(w_i, w_i')\} \cup \{(y_j^k, y_j^{k'}) | x_i \text{middle variable of clause} C_k\}$ to M_T and $m_i^2 = \{(w_i', w_i)\} \cup \{(y_j^{k'}, y_j^k) | x_i \text{middle variable of clause} C_k\} \cup \{(t_i', t_i)\}$, $m_i^3 = \{(t_i, t_i''), (t_i'', t_i')\}$, $m_i^4 = \{(t_i, t_i')\}$, and $m_i^5 = \{(t_i', t_i''), (t_i'', t_i)\}$ to M_F.

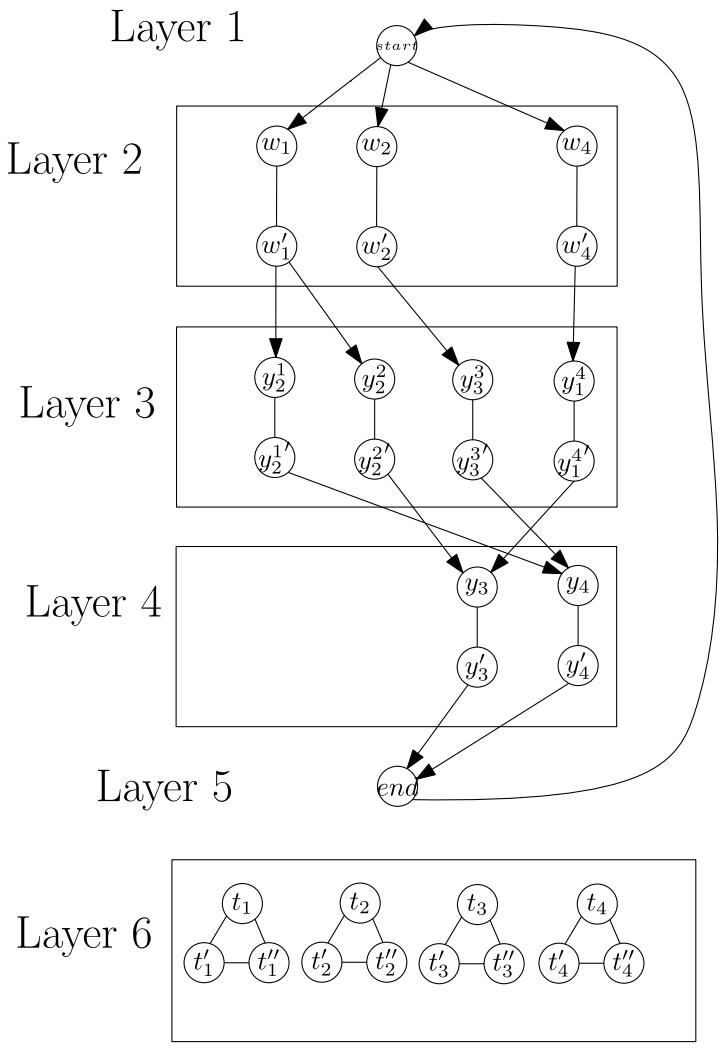

Figure 4.2: The resulting graph, where the boolean formula used in Φ is $(x_1 \wedge x_2 \wedge x_4) \vee (x_1 \wedge x_2 \wedge x_3) \vee (x_2 \wedge x_3 \wedge x_4) \vee (x_4 \wedge x_1 \wedge x_3)$ and the two variables x_1, x_2 were colored with B, the variable x_4 was colored with C and the variable x_3 was colored with F.

The construction of a mixed graph G with the oriented edge set M can be seen in Fig. 4.2 as an example.

Since each variable and clause requires a constant number of nodes and edges, the transformation takes place in polynomial space.
Notice that by construction our oriented edge group M has the desired properties because each oriented edge is in exactly one oriented edge group and because m_i^2 has the most oriented edges if it has the color F or B since it then has exactly two oriented edges through the layers 2,6 or 3,6 and also one oriented edge for each occurrence in the middle of the clause. This makes it $t(\Phi) + 2$ edges.
The first thing we notice is that orienting an undirected edge is the same as selecting a particular oriented edge group because when orienting an undirected edge, the entire group is oriented and since the oriented edge groups are perfectly complete, this means that each oriented edge is in exactly one oriented edge group. Now, however, we can see that if a player selects an oriented edge group $m_i^a \in M_F$, with $a \in \{2, 3, 4, 5\}$, he loses directly. This is because by choosing the oriented edge group m_i^a the other player can close the directed cycle $t_i \to t'_i \to t''_i$ by choosing a $m_i^b \in M_F$, with $b \in \{2, 3, 4, 5\}$. Assuming a player chooses m_i^2 or m_i^4, an appropriate response to create a directed cycle would be to choose m_i^3 or m_i^5

respectively.
Since each player can always choose an oriented edge group from M_T, we say, without limiting the generality, that if the player has to choose an oriented edge group, he will always choose one from M_T.
An observation here is that if a variable x_i has the color F, then this variable has no nodes in layer 2. If a variable x_i has the color C, then this variable has no nodes in layer 3. And lastly, if a variable x_i has the color B, then this variable has no nodes in layer 4.
It is still necessary to demonstrate that each instance of *impartial seek 3-color positive 3-DNF SAT* can be converted into an equivalent *seek oriented cycle game* instance by constructing the graph G and the oriented edge groups M as previously mentioned where an optimal strategy for player A of the *impartial seek positive 3-color 3-DNF SAT* game is equivalent to an optimal strategy for player 1 of the *seek oriented cycle game*, and an optimal strategy for player B of the *impartial seek 3-color positive 3-DNF SAT* game is equivalent to an optimal strategy for player 2 of the *seek oriented cycle game*. To demonstrate this we can show that:

$$\text{choosing } m_i^1 \in M_T \Leftrightarrow \text{ choosing } x_i \in X$$

Another observation we can make is that for each possible occurrence of a variable at a position in a clause we need two nodes and an undirected edge for layers 2 and 4 and for layer 3 we need two nodes and an undirected edge for each occurrence of a variable. In Theorem 4.2 we have in principle added two nodes and an undirected edge for each variable in layers 2 and 4 and for layer 3 we have added two nodes and an undirected edge for each occurrence of a variable. This means that in Theorem 4.2 variables could appear anywhere and we had to make sure that they could be set to true at any position in the clauses. Since we could exclude in this proof that a certain variable is at a certain position, we did not need a truth constraint for this variable and position and the corresponding layer does not need two nodes and an undirected edge for this variable. This is the essential difference between the construction of Theorem 4.2 and Theorem 4.4. Since the difference between the construction is so minimal, and the proof of correctness becomes very identical, we omit it here and refer to the proof in Theorem 4.2. □

We give another proof about another version and show that it is also PSPACE-hard. We again use the standard *seek oriented cycle game*, but we require different properties for the oriented cycle groups than in Theorem 4.4. For the oriented cycle groups M we demand, that M is *perfectly complete* and we define $max\ M \coloneqq max\{|m| \colon m \in M\}$.

For a 3-DNF SAT formula Φ with the clauses $\{C_1, \ldots, C_n\}$ we define $t(\Phi)$, where $t(\Phi) \in \mathbb{N}$ is the number of times a variable can be at most in the middle of the clauses $\{C_1, \ldots, C_n\}$. Since the clauses are conjunctions of three literals, the order of the literals can be reversed within the clauses. We look at the permutation where $t(\Phi)$ is minimal.

Theorem 4.5. *Seek oriented cycle game is PSPACE-hard, even if M is* unique. *Moreover, if impartial seek positive 3-color 3-DNF SAT is PSPACE-hard with $t(\Phi) = t^*$, $t^* \in \mathbb{N}$, then seek oriented cycle game is PSPACE-hard for $max\ M \leq t^* + 1$*

Proof. To prove that our game is PSPACE-hard, we provide a reduction:

$$\textit{Impartial seek positive 3-color 3-DNF SAT} \leq_p \textit{seek oriented cycle game}$$

We get an instance Φ of impartial seek positive 3-color 3-DNF SAT. The formula used in Φ is a conjunction of clauses $C_1 \wedge \cdots \wedge C_n$, where each clause is a disjunction of three positive

literals, i.e. $C_k = x_i \vee x_j \vee x_l$. The number $t(\Phi) \in \mathbb{N}$ is the number of times a variable can be at most in the middle of the clauses $\{C_1, \ldots, C_k\}$. In addition, no clause has a negated literal and both players take turns choosing a literal and setting it to true. Each variable has been assigned a color, either F, C, or B. There is a permutation of variables within the clauses so that if a variable has the color F, C, or B, it never is in the front, center, or back of the clauses respectively. Let the clauses $C_1 \wedge \cdots \wedge C_n$ already be exactly according to this permutation. The aim of both players A and B is to be the first to fulfill the boolean formula.
Next, we create the mixed graph G together with the oriented edge groups M depending on Φ, where player 1 wins the *seek oriented cycle game* on (G, M) if and only if player A of the impartial *seek positive 3-color 3-DNF SAT* wins on Φ. The steps to create G, M are as follows:

1. Create five layers in which the nodes of G come in.
2. Make a start and end node; from the end node to the start node, there is a directed edge. Put the start node in layer 1 and the end node in layer 5.
   a) If x_i has the color F, then introduce the nodes $\{z_i, z_i'\}$. Put the nodes $n_2 = \{z_i, z_i'\}$ in layer 4. We create one undirected edge $[z_i, z_i']$.
   b) If x_i has the color C, then introduce the nodes $\{w_i, w_i', z_i, z_i'\}$. Put the nodes $n_1 = \{w_i, w_i'\}$ in layer 2 and the nodes $n_2 = \{z_i, z_i'\}$ in layer 4. We create two undirected edges $[w_i, w_i']$ and $[z_i, z_i']$.
   c) If x_i has the color B, then introduce the nodes $\{w_i, w_i'\}$. Put the nodes $n_1 = \{w_i, w_i'\}$ in layer 2. Here, we create one undirected edge $[w_i, w_i']$.
3. For each clause C_k with $C_k = x_i \wedge x_j \wedge x_l$ there are two nodes $\{y_j^k, y_j^{k\prime}\}$ in layer 3. There are also four directed edges. We add $(start, w_i)$, (w_i', y_j^k), $(y_j^{k\prime}, z_l)$, and (z_l', end). We also add $[y_j^k, y_j^{k\prime}]$ to G.
4. For the oriented edge groups M we proceed as follows:
   a) If x_i has the color F, add $m_i^1 = \{(y_j^k, y_j^{k\prime}) | x_i \text{middle variable of clause} C_k\} \cup \{(z_i, z_i')\}$ to M
   b) If x_i has the color C, add $m_i^1 = \{(w_i, w_i'), (z_i, z_i')\}$ to M
   c) If x_i has the color B, add $m_i^1 = \{(w_i, w_i')\} \cup \{(y_j^k, y_j^{k\prime}) | x_i \text{middle variable of clause} C_k\}$ to M

The construction of a mixed graph G with the oriented edge set M can be seen in Fig. 4.3 as an example.

Since each variable and clause requires a constant number of nodes and edges, the transformation takes place in polynomial time. Notice that by construction our oriented edge group M has the desired properties because each oriented edge is in one or no oriented edge group and because m_i has the most oriented edges if it has the color F or B since it then has exactly one oriented edges through the layers 1 or 3 and also one oriented edge for each occurrence in the middle of the clause. This makes it $t(\Phi) + 1$ edges.
To complete the reduction, we need to show that any instance of *impartial seek 3-color positive 3-DNF SAT* can be converted to an equivalent *seek oriented cycle game* instance by building the mixed graph G and the oriented edge groups M as mentioned. The optimal strategy for player A on Φ of the *impartial seek 3-color positive 3-DNF SAT* must be equivalent to an optimal strategy for player 1 on (G, M) of the *avoid oriented cycle game*. To demonstrate this we can show that:

$$\text{choosing } m_i \in M \Leftrightarrow \text{ choosing } x_i \in X$$

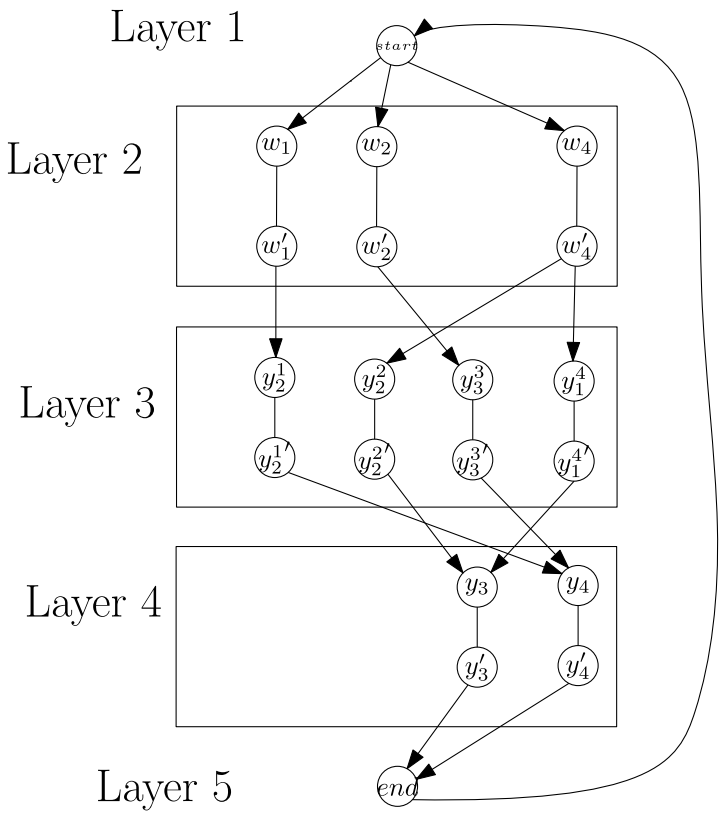

Figure 4.3: The resulting graph, where the boolean formula used in Φ is $(x_1 \wedge x_2 \wedge x_4) \vee (x_4 \wedge x_2 \wedge x_3) \vee (x_2 \wedge x_3 \wedge x_4) \vee (x_4 \wedge x_1 \wedge x_3)$ and the two variables x_1, x_2 were colored with B, the variable x_4 was colored with C and the variable x_3 was colored with F.

The first thing we notice is that M and the oriented edge groups M_T from Theorem 4.4 are equal to each other. Thus a proof would be exactly the same as in Theorem 4.4. Since Theorem 4.4 is very identical to Theorem 4.2 and we have already not formally proved Theorem 4.4, we will also omit this proof. □

4.3 Seek Oriented Cycle Game in P

Theorem 4.6. *Seek oriented cycle game* $\in P$.

Proof. As in Theorem 3.5, we will again prove that there can be no mixed graph G where a player is forced to play a move that loses. This makes it easy to find an algorithm that is in polynomial time by calculating only for the next two moves whether it is a losing move. That is, we look at the graph G before the last two moves were made and ask ourselves if a player was forced to lose or if it was avoidable.
We assume that player 2 is forced to orient an undirected edge on the mixed graph G, which allows player 1 to then directly create a directed cycle. Thus we know that the mixed graph G consists of several directed cycles, all of which have 2 undirected edges so that player 2 can orient one of the two edges, and player 1 can then close the directed cycle by orienting the other of the two edges. Let e_1 be the undirected edge that player 1 orients (from b_1 to a_1) to create the directed cycle C_1. With this, we ask ourselves, what was the last move of player 2? If player 2 did not orient an undirected edge in the directed cycle

C_1, player 2 could have oriented e_1 himself and won the game. Thus, player 2 must have oriented an undirected edge e_2 (from a_2 to b_2), which is contained in C_1 as shown in Fig. 4.4.

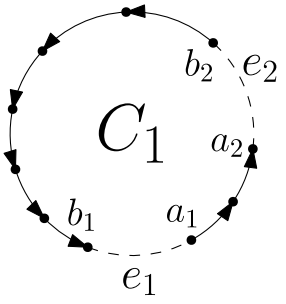

Figure 4.4: A directed cycle except for 2 undirected edges

Now the question is, why did player 2 not direct the undirected edge e_2 the other way around (i.e. from b_2 to a_2)? By orienting the undirected edge e_2 the other way around, the directed cycle C_1 would be prevented. Thus, there must be another directed cycle C_2 with which player 1 can win directly after player 2 has oriented e_2 (from b_2 to a_2). The directed cycle C_2 must still have an undirected edge e_3 for player 1 to win. The directed cycle C_2 must also contain e_2, because if C_2 does not contain e_2, then player 2 could directly orient e_3 and win. Therefore the directed cycle C_2 must both contain e_2 as well as an undirected edge e_3 that allows player 1 to win as shown in Fig. 4.5.

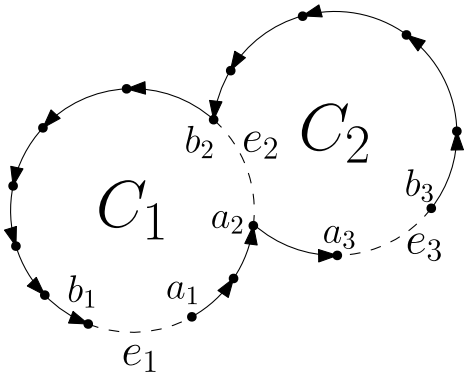

Figure 4.5: Two directed cycles with 2 undirected edges each

However, there is still the case that $e_1 = e_3$ and thus the two directed cycles C_1 and C_2 share two undirected edges instead of only e_2. In this case, however, we see that the undirected edges no longer play a role, since we have already closed a directed cycle and C_1, together with C_2, form a directed cycle as shown in Fig. 4.6.

Thus we can ignore this case, as it is against our assumption that the game is not over yet. This gives us an undirected edge e_3 that player 2 could orient. Orienting e_3 (from a_3 to b_3) makes no sense, as now player 1 can orient e_2 appropriately and create a directed cycle. However, orienting e_3 (from b_3 to a_3) makes sense because it prevents the directed cycle C_2. But since it is assumed that player 1 can create a directed cycle after player 2's move and player 1 can not create a directed cycle C_1 on his move, there must be another directed cycle C_3. An undirected edge e_4 must also be contained in C_3 in order for player 1 to win after player 2 orients e_3 (from b_3 to a_3). The directed cycle C_3 must also contain e_3 because if C_3 did not, player 2 could win by directly orienting e_4. Therefore e_3 as well as an undirected edge e_4 must be contained in C_3 as shown in Fig. 4.5.

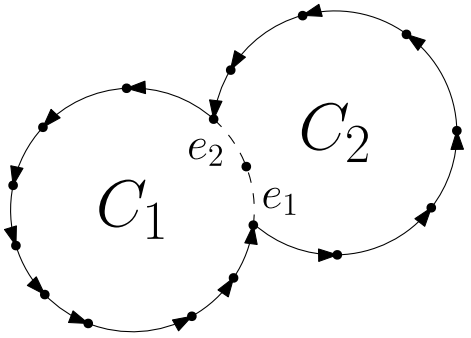

Figure 4.6: Two directed cycles with 2 undirected edges each and a directed cycle

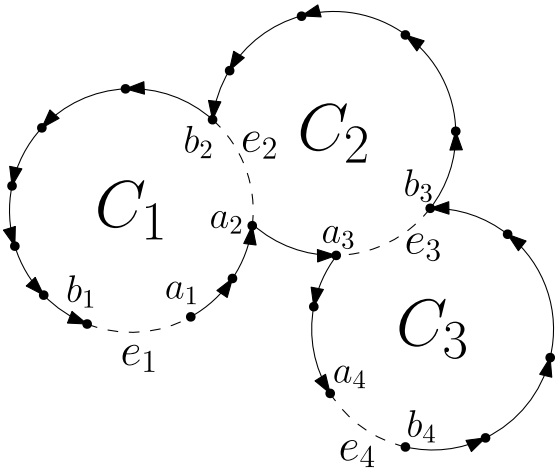

Figure 4.7: Three directed cycles with two undirected edges each

Here it is again true that $e_2 \neq e_4$, since C_2, together with C_3, close a directed cycle and thus violates our assumption that the game is not yet over. This, however, gives us another undirected edge e_4, which player 2 can take and orient (from b_3 to a_3) so that the directed cycle C_3 is prevented. Now it is again true that this orientation must also lose directly according to our assumption.
Thus in the mixed graph G there must be a directed cycle C_k each time, so that in the directed cycle C_{k-1} the undirected edge e_k cannot be oriented by player 2 in such a way that the directed cycle (from b_k to a_k) C_{k-1} is prevented. Since the mixed graph G is finite and thus can only have finitely many undirected edges, there must be a directed cycle C_{k-1}, which does not depend on the directed cycle C_k. The undirected edge e_k must be contained in another directed cycle C_l, because if it were not, player 2 could orient the undirected edge in such a way that the directed cycle C_{k-1} would be prevented and would thus have found a move that does not lose directly. Thus we have $e_k = e_l$ for some $k > l + 1$ with k small as possible. For $k = l + 2$ we have already seen that the directed cycles C_{k-1} together with C_l close a circle, as for example in the case $e_1 = e_3$ as shown in Fig. 4.6. We can say without limitation of generality that in every directed cycle C_i except for C_k there is a path from a_i to a_{i+1}, where a_i,a_{i+1} are nodes of undirected edges. For C_k there is a path from a_{k-1} to a_l. Thus we have found a directed cycle, namely: $a_l \to a_{l+1} \to \cdots \to a_{k-1}$. This is against our assumption that the game is not yet over. Thus, there cannot be a mixed graph G where a player is forced to make a losing move.
This case, where a directed cycle C_{k-1} is added and its two undirected edges are already present in the mixed graph G, is also shown in Fig. 4.8, where we have the case that here $k = 5$ and $l = 1$.

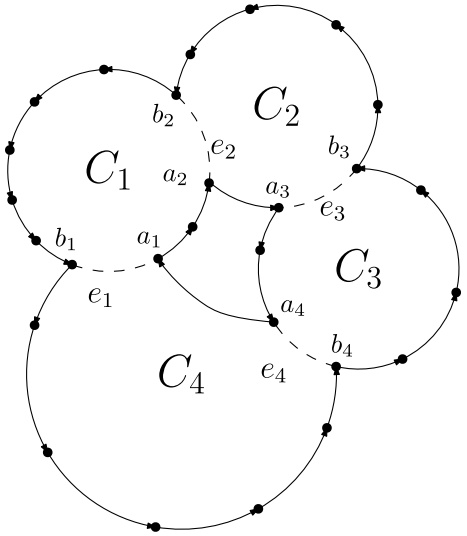

Figure 4.8: Four directed cycles with two undirected edges each and a directed cycle $a_1 \to a_2 \to a_3 \to a_4$

Thus, we have now seen that there can be no mixed graph G in which a player has no more moves that do not directly lose since the attempt to create such a mixed graph G always results in a directed cycle already being present in the graph.
Now we want to show that there can also be no mixed graph G where a player has no move left that does not lose. We assume that there is such a mixed graph G, where one player has no more move which does not lose. So in the course of the game, there must also be a mixed graph G', so that a player no longer has a move that directly loses. However, in the proof above we have seen that such a G' cannot exist, so our assumption is wrong and there is no mixed graph G so that a player has no more moves that do not lose. Therefore, it is sufficient to play one move each round which does not lose directly.
To specify an algorithm that finds an optimal move in polynomial time, we can take a move that does not lose. Since we always have a move that does not lose, all we need to do is make sure a move does not lose directly. It is also true that the opponent has a move such a move, so a winning move can only result from an error on the part of the opponent. The algorithm works as follows:

1. Check both orientations of each undirected edge, if there is a winning move in $\mathcal{O}(|E| * (|V| + |E|))$. If there is, orient this edge accordingly.
2. Find an orientation of an undirected edge so that the opposing player cannot close a directed cycle in the next turn in $\mathcal{O}(|E| * (|V| + |E|))$.

□

This brings the chapter on the *avoid oriented cycle* game to a close. Throughout the chapter, we learned which variations of the game are PSPACE-complete and examined which variations can be solved in polynomial time.

5. Default Oriented Cycle Game

This chapter is about the default *oriented cycle game*. That is, we look at the normal *oriented cycle game*, without oriented edge groups. We will first try to find out whether the decision problem *"Given a mixed graph G, which player has a winning strategy?"* is solvable in polynomial time. Then we analyze some interesting graphs and indicate which player wins on them.

5.1 Default Oriented Cycle Game in P?

The question of which player wins on which graph G has always been asked in the *oriented cycle game*. Béla Bollobás and Tamás Szabó, who invented the *oriented cycle game*, proved in their presentation of the game that player C wins if the graph G has a subgraph with $2n-2$ or more undirected edges [BS98]. Now the question arises, is this perhaps a unique characterization and does player A win if every subgraph of graph G has the most $2n-3$ undirected edges? By this method, we would find an algorithm that finds out in polynomial time which player wins on the graph G because finding out whether G has at most $2n-3$ undirected edges on each subgraph works in polynomial time. We have seen from the example in the Introduction 1 that there is a graph G which has at most $2n-3$ undirected edges for each subgraph of G and player A wins since a triangle fulfills exactly these conditions. We prove that there exists a graph G on which player C wins and each subgraph of G has at most $2n-3$ undirected edges.

Theorem 5.1. *Player C wins on the 3-prism graph G.*

Proof. The 3-prism graph is a graph where each subgraph has at most $2n-3$ undirected edges, as can be seen in Fig. 5.1.

To prove that player C wins on the 3-prism graph, we will show every possibility that player A has and show that player C still manages to create a directed cycle. To do this, we can look at Fig. 5.2, which contains all the game possibilities for player A except for those that are equal to each other and have therefore been left out. Player C starts the game and an arrow with a C symbolizes that player C has made a move. An arrow with an A in Fig. 5.2 symbolizes that player A has made a move, the (forced) means that player A was forced to make this move because if he had not made it, player C could have closed a directed cycle in the next move.

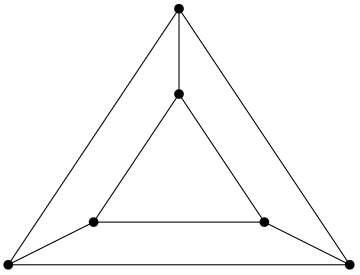

Figure 5.1: The 3-prism graph

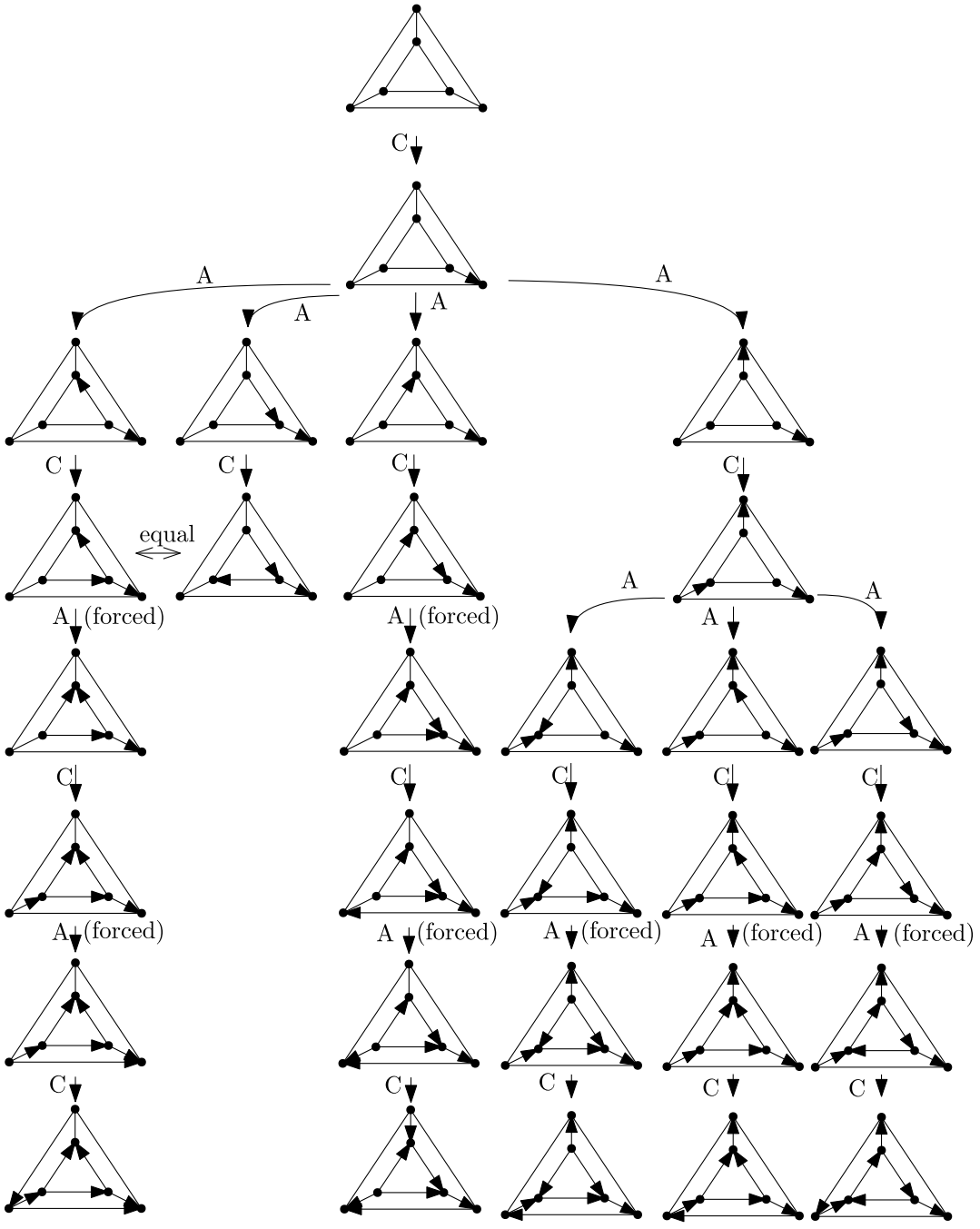

Figure 5.2: The game tree of the 3-prism graph

Thus we have seen that there is a graph G that for each subgraph has at most $2n-3$ undirected edges where player A wins and there is a graph G' that for each subgraph has at most $2n-3$ undirected edges where player C wins. Thus, by examining the subgraphs, one cannot find out which player wins. This still leaves open the question of whether there is an algorithm that receives a graph as input and finds out in polynomial time which player has a winning strategy. □

5.2 Overview of some interesting Graphs

This section is about interesting graphs where we try to figure out which player is winning by giving a winning strategy for that player. We start with a proof where we show that adding a node with 2 undirected edges to a random mixed graph G does not differentiate the outcome of the game if G has just many undirected edges.

Theorem 5.2. *Player A wins on $G + Hen_1$, if player A wins on G and G has an even number of undirected edges.*

Proof. Let G be a mixed graph on which player A has a winning strategy and let G have an even number of undirected edges. Now let $G + Hen_1$ be a mixed graph resulting from performing a Henneberg 1 step on G. That is, two different nodes x_i,x_j are randomly selected in G and connected to a new node $x_k \notin G$ with an undirected edge e_{ik}, e_{jk} each. A winning strategy for player A on $G + Hen_1$, depending on a winning strategy for player A on G works as follows:

1. If C plays in G, that is, orients an undirected edge of G, then answer in G using the winning strategy for player A on G.
2. If C orients a new undirected edge e_{ik} or e_{jk}, then orient the other undirected edge accordingly so that the node x_k becomes the source or sink.

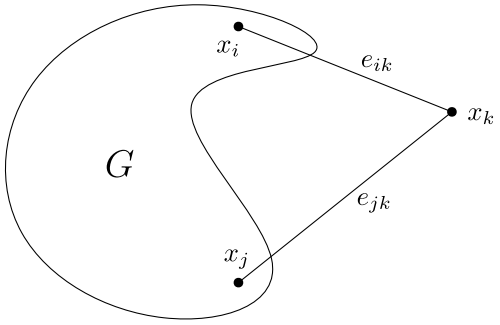

Figure 5.3: Graph G' resulting from performing a Henneberg 1 step on G

We still have to show that this is a winning strategy for $G + Hen_1$ depending on the fact that player A already had a winning strategy for G and that G has many undirected edges. One observation we make here is that player A always has an answer to player C when player C is playing in G. This is because player C starts the game and if player A can no longer answer in G after player C has played in G, then G must have an odd number of undirected edges.
The next observation is that the node x_k cannot be included in any directed cycle, since it is either a source or a sink, because player C is forced to orient an undirected edge e_{ik} or e_{jk} first and then player A orients the other undirected edge accordingly so that the node x_k becomes the source or sink. Thus, the only directed cycle that can occur in $G + Hen_1$ is in G. However, since if player C plays on G, player A has used the winning strategy for G, no directed cycle can occur in G either. Thus we have found a winning strategy with which player A can prevent a directed cycle on $G + Hen_1$ □

With the help of Theorem 5.2, we can now better analyze some mixed graphs and better understand which player is winning. We now use Theorem 5.2 to show that on the $n \times m$ grid graph, player A wins if $n + m$ is even.

Theorem 5.3. *Player A wins on the $n \times m$ grid, if $m + n$ is even.*

Proof. We start with the graph G, which is an $n \times m$ grid, and want to show that there is a winning strategy for player A. For this, we will use Theorem 5.2. The first thing we notice is that the node $x_{n,m}$ at the bottom right has only two undirected nodes as can be seen in Fig. 5.4.

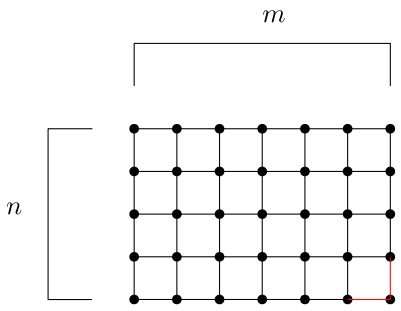

Figure 5.4: An $n \times m$ grid graph G, with $n = 5$ and $m = 7$. Here the undirected edges of nodes $x_{5,7}$ are colored red.

Thus the node $x_{n,m}$ is a Henneberg 1 step which has an undirected edge to the nodes $x_{n-1,m}$ and $x_{n,m-1}$. By Theorem 5.2, we can find a winning strategy for G if we have found a winning strategy for $G' = G - x_{n,m}$ and if G' has an even number of undirected edges. If we now take a closer look at G', we notice that the two nodes $x_{n-1,m}$ and $x_{n,m-1}$ each have only two undirected edges as can be seen in Fig. 5.5.

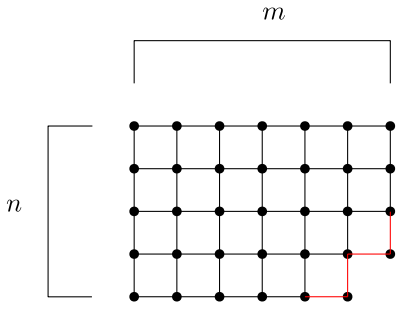

Figure 5.5: The grid graph G', with $n = 5$ and $m = 7$. Here the undirected edges of nodes $x_{5,6}$ and $x_{4,7}$ are colored red.

Thus the nodes $x_{n-1,m}$ and $x_{n,m-1}$ are also the only nodes that have passed through a Henneberg 1 step. By Theorem 5.2 we can again find a strategy for G' if we have found a strategy for $G'' = G' - x_{n-1,m} - x_{n,m-1}$ and G'' has an even number of undirected edges. This process can be continued until all nodes with two undirected edges have been removed. Finally, you get G''', which is a simple path of undirected edges from $x_{1,n}$ to $x_{m,1}$ as can be seen in Fig. 5.6.

It is important that an even number of undirected edges are dropped from the graph in each step, thus G''' has an even number of undirected edges if and only if G'', G' or also G have an even number of undirected edges.
Thus, if we find a winning strategy for G''' and G''' has an even number of undirected edges, then we have found a winning strategy for G. However, since G''' is only a simple

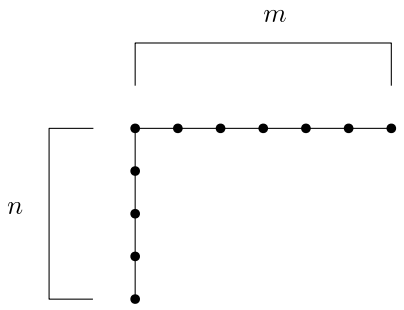

Figure 5.6: The graph G''', which is a simple path of undirected edges from $x_{1,n}$ to $x_{m,1}$, in this case is $n = 5$ and $m = 7$

path, there is no losing strategy for player A here and player A can play on G''' at will and win. For the length of the path, exactly $n - 1$ undirected edges are needed from $x_{n,1}$ to $x_{1,1}$ and exactly $m - 1$ undirected edges from $x_{1,1}$ to $x_{1,m}$. This means that in total G''' has $n - 1 + m - 1 = n + m - 2$ undirected edges. Since $(n + m - 2) \mod 2 = (n + m) \mod 2$, G''' has even undirected edges if $m + n$ is even.
Thus we have shown that player A has a winning strategy on the $n \times m$ grid graph G, if $m + n$ is even. □

Theorem 5.4. *The start player loses on the cube.*

Proof. We consider the cube as a graph G as shown in Fig. 5.7.

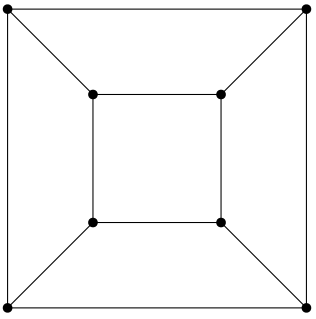

Figure 5.7: The cube as a graph

Since G has 12 undirected edges, a game tree is not so easy here, since there are naive $24!! = 24 \cdot 22 \cdot 20 \cdot \ldots \cdot 2 = 1,961,990,553,600$ possibilities to play the game on G. Since the graph in Theorem 5.1 had only 9 undirected edges, it was possible to specify a game tree in this case. However, we will give pseudocode which calculates in polynomial space whether player A or player B wins. In Algorithm 5.1, it is player C's turn and he searches the entire game tree looking for a strategy to force a directed cycle regardless of A's moves. Algorithm 5.2 does the same, but for player A. To find out that on the cube the starting player loses, we have to run Algorithm 5.1 once and Algorithm 5.2 once with the cube as a graph. We see that both algorithms return *false*.

We give another possible play in which player A loses. We do not go through every possibility or look at equivalence classes, but take the moves for player A that are most logical for us. Since player A loses, he must also start. As player A, we try to create as many sources as possible and orient our edges in such a way that we have a chance to create a source. Fig. 5.8 shows this strategy for player A, but we can see that player C has a strategy to create a directed cycle. This is only meant as an example, as a formal proof would be too time-consuming at this point.

Algorithm 5.1: PlayerCMove

Input: Mixed Graph $G = (V, E, A)$
Output: Whether player C has a winning strategy

```
1  forall e ∈ E do
2      directEdge (e, G)                          // direct edge e randomly
3      if ∃ directed cycle then
4          resetEdge (e, G)
5          return true
6      reverseEdge (e, G)
7      if ∃ directed cycle then
8          resetEdge (e, G)
9          return true
10     if PlayerAMove (G) then
11         reverseEdge (e, G)          // direct edge e the other way around
12         if PlayerAMove (G) then
13             resetEdge (e, G)                   // undirect edge e
14             continue with next iteration
15     resetEdge (e, G)
16     return true
17 return false
```

□

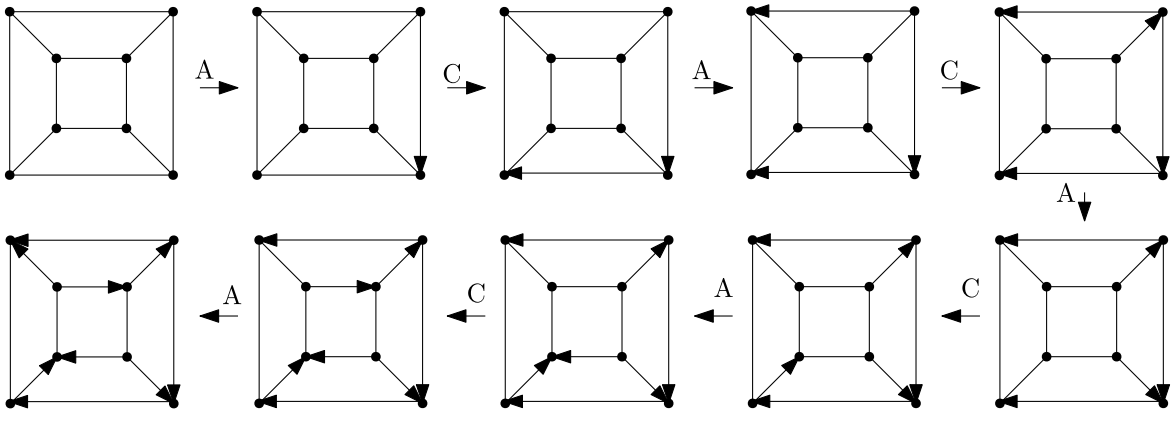

Figure 5.8: A possible way to play the cube

Algorithm 5.2: PlayerAMove

Input: Mixed Graph $G = (V, E, A)$
Output: Whether player A has a winning strategy

```
1  forall e ∈ E do
2      directEdge (e, G)                          // direct edge e randomly
3      if ∃ directed cycle then
4          reverseEdge (e, G)
5          if ∃ directed cycle then
6              resetEdge (e, G)                   // undirect edge e
7              continue with next iteration
8      if PlayerCMove (G) then
9          reverseEdge (e, G)        // direct edge e the other way around
10         if ∃ directed cycle then
11             resetEdge (e, G)
12             continue with next iteration
13         if PlayerCMove (G) then
14             resetEdge (e, G)
15             continue with next iteration
16     resetEdge (e, G)
17     return true
18 return false
```

6. Conclusion

In this thesis, we studied the oriented cycle game in its avoid, seek, and default version. For the seek and avoid version, we always looked at the game together with oriented edge groups. In Chapter 3 we found out that the following versions of the *avoid oriented cycle game* are PSPACE-complete:

1. *avoid oriented cycle game*, even if the set M if oriented edge groups is *perfectly complete*, *symmetrical*, and $max\ M \coloneqq max\{|m|\colon m \in M\} = 3$
2. *avoid oriented cycle game*, even if the set M if oriented edge groups is *unique* and $max\ M = 2$.

In Chapter 3 we also found out that the *avoid oriented cycle game* without the oriented edge groups, equivalently $maxM = 1$ and M perfectly complete, is solvable in polynomial time. After that, in Chapter 4, we focused on the *seek oriented cycle game*. Let $t^* = min\ t \in \mathbb{N}$: *impartial positive 3-DNF SAT* with $t(\Phi) \leq t^*$ is PSPACE-hard. We also found the following versions and proved that they are PSPACE-complete:

1. *seek oriented cycle game*, even if the set M of oriented edge groups is *perfectly complete*. If *impartial seek positive 3-DNF SAT* is PSPACE-hard with $t(\Phi) = t^*$, then the *seek oriented cycle game* is even PSPACE-compelte for $max\ M \leq t^* + 3$
2. *seek oriented cycle game*, even if the set M of oriented edge groups is *perfectly complete*. If *impartial seek positive 3-DNF SAT* is PSPACE-hard with $t(\Phi) = t^*$, then the *seek oriented cycle game* is even PSPACE-compelte for $max\ M \leq t^* + 2$
3. *seek oriented cycle game*, even if the set M of oriented edge groups is *unique*. If *impartial seek positive 3-DNF SAT* is PSPACE-hard with $t(\Phi) = t^*$, then the *seek oriented cycle game* is even PSPACE-compelte for $max\ M \leq t^* + 1$

Where $t(\Phi) \in \mathbb{N}$ is the number of times a variable can be at most in the middle of the clauses of *Impartial seek positive 3-color 3-DNF SAT* Φ. Following this, we also proved in Chapter 4 that the *seek oriented cycle game*, without the oriented edge groups, is solvable in polynomial time. We looked at the default *oriented cycle game* in Chapter 5, proving that there exists a Laman graph, such that Player C has a winning strategy and therefore we cannot determine which player wins based on the number of undirected edges of every subgraph. Finally, we looked at interesting graphs, such as the grid graph, in Chapter 5 and proved which player wins on them. There may be more results that can be obtained in this field. Open questions that would be interesting to investigate in the future would be:

1. Does player A also win on the grid graph if $m + n$ is even and if so, what is his strategy?
2. Does there exist an $t^* \in \mathbb{N}$ such that an instance Φ of *impartial positive 3-DNF SAT* is PSPACE-hard with $t(\Phi) = t^*$?
3. Is the *biased oriented cycle game* PSPACE-complete?
4. Is there an upper bound of undirected edges for graph G and on each subgraph of G such that player A always wins?
5. Does a certain player always win on graph G if G is Laman and, in addition, each node has exactly three undirected edges?
6. What is the complexity of the *oriented cycle game*? Is there an algorithm that finds out in polynomial time which player wins?

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[BS98] Béla Bollobás and Tamás Szabó. The oriented cycle game. *Discrete Mathematics*, 186(1):55–67, 1998.

[CE78] V. Chvátal and P. Erdös. Biased positional games. In B. Alspach, P. Hell, and D.J. Miller, editors, *Algorithmic Aspects of Combinatorics*, volume 2 of *Annals of Discrete Mathematics*, pages 221–229. Elsevier, 1978.

[CKS01] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. Society for Industrial and Applied Mathematics, 2001.

[Dem01] Erik D. Demaine. Playing games with algorithms: Algorithmic combinatorial game theory. In Jiří Sgall, Aleš Pultr, and Petr Kolman, editors, *Mathematical Foundations of Computer Science 2001*, pages 18–33, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[FSU93] Aviezri S Fraenkel, Edward R Scheinerman, and Daniel Ullman. Undirected edge geography. *Theoretical Computer Science*, 112(2):371–381, 1993.

[Hen11] 1850-1933 (viaf)47520410 Henneberg, Lebrecht. *Die graphische Statik der starren Systeme*. Leipzig, 1911.

[Lam70] G. Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4(4):331–340, Oct 1970.

[Sch78] Thomas J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16(2):185–225, 1978.