Faster Winner Determination Algorithms for (Colored) Arc Kayles

Tesshu Hanaka[1][0000−0001−6943−856X], Hironori Kiya[2][0000−0002−0742−4477], Michael Lampis[3][0000−0002−5791−0887], Hirotaka Ono[4][0000−0003−0845−3947], and Kanae Yoshiwatari[4][0000−0001−5259−7644]

[1] Department of Informatics, Faculty of Information Science and Electrical Engineering, Kyushu University, Fukuoka, Japan
hanaka@inf.kyushu-u.ac.jp
[2] Department of Informatics, Department of core informatics, Osaka Metropolitan University, Japan kiya@omu.ac.jp
[3] Université Paris-Dauphine, PSL University, CNRS, LAMSADE, 75016, Paris,France
michail.lampis@lamsade.dauphine.fr
[4] Department of Mathematical Informatics, Graduate School of Informatics, Nagoya University, Nagoya, Japan
yoshiwatari.kanae.w1@s.mail.nagoya-u.ac.jp,ono@nagoya-u.jp

Abstract. Arc Kayles and Colored Arc Kayles, two-player games on a graph, are generalized versions of well-studied combinatorial games Cram and Domineering, respectively. In Arc Kayles, players alternately choose an edge to remove with its adjacent edges, and the player who cannot move is the loser. Colored Arc Kayles is similarly played on a graph with edges colored in black, white, or gray, while the black (resp., white) player can choose only a gray or black (resp., white) edge. For Arc Kayles, the vertex cover number (i.e., the minimum size of a vertex cover) is an essential invariant because it is known that twice the vertex cover number upper bounds the number of turns of Arc Kayles, and for the winner determination of (Colored) Arc Kayles, $2^{O(\tau^2)}n^{O(1)}$-time algorithms are proposed, where τ is the vertex cover number and n is the number of vertices. In this paper, we first give a polynomial kernel for Colored Arc Kayles parameterized by τ, which leads to a faster $2^{O(\tau \log \tau)}n^{O(1)}$-time algorithm for Colored Arc Kayles. We then focus on Arc Kayles on trees, and propose a $2.2361^{\tau}n^{O(1)}$-time algorithm. Furthermore, we show that the winner determination Arc Kayles on a tree can be solved in $O(1.3831^n)$ time, which improves the best-known running time $O(1.4143^n)$. Finally, we show that Colored Arc Kayles is NP-hard, the first hardness result in the family of the above games.

Keywords: Arc Kayles, Combinatorial Game Theory, Exact Exponential-Time Algorithm, Vertex Cover

1 Introduction

ARC KAYLES is a combinatorial game played on a graph. In ARC KAYLES, a player chooses an edge of an undirected graph G and then the selected edge and its neighboring edges are removed from G. In other words, a player chooses adjacent two vertices to occupy. The player who cannot choose adjacent two vertices loses the game.

NODE KAYLES, a vertex version of ARC KAYLES, and ARC KAYLES were introduced in 1978 by Schaefer [12]. The complexity of NODE KAYLES is shown to be PSPACE-complete, whereas that of ARC KAYLES is less known. An important aspect of ARC KAYLES is a graph generalization of CRAM, which is a well-studied combinatorial game introduced in [6]. CRAM is a simple board game: two people alternately put a domino on a checkerboard, and the player who cannot place a domino will lose the game. CRAM is interpreted as ARC KAYLES, when a graph is a two-dimensional grid graph. Though CRAM is quite more restricted than ARC KAYLES, the complexity remains open. Since an algorithm for ARC KAYLES is available for CRAM, a study for ARC KAYLES would help the study for CRAM.

This paper presents new winner-determination algorithms together with elaborate running time analyses. The running time of our algorithms is parameterized by the vertex cover number of a graph. Note that the vertex cover number of a graph is strongly related to the number of turns of ARC KAYLES, which is the total number of actions taken by two players, as seen below. Intuitively, the number of turns tends to reflect the complexity of a game because it is the depth of the game tree, and it is reasonable to focus on it when we design winner-determination algorithms.

The relation between the number of turns of ARC KAYLES is observed as follows. During a game of ARC KAYLES, edges chosen by the players form a matching, and the player who completes a maximal matching wins; the number of turns in a gameplay is the size of the corresponding maximal matching. Since the maximum matching size is at most twice the minimum maximal matching size, which is also at most twice the minimum vertex cover number, the number of turns of ARC KAYLES is linearly upper and lower bounded by the vertex cover number.

1.1 Partisan variants of Arc Kayles

In this paper, we also study partisan variants of ARC KAYLES: COLORED ARC KAYLES and BW-ARC KAYLES. In combinatorial game theory, a game is said to be *partisan* if some actions are available to one player and not to the other. COLORED ARC KAYLES, intoroduced in [17], is played on an edge-colored graph $G = (V, E_{\rm B} \cup E_{\rm W} \cup E_{\rm G})$, where $E_{\rm B}, E_{\rm W}, E_{\rm G}$ are disjoint. The subscripts B, W, and G of $E_{\rm B}, E_{\rm W}$, and $E_{\rm G}$ respectively stand for black, white, and gray. For every edge $e \in E_{\rm B} \cup E_{\rm W} \cup E_{\rm G}$, let $c(e)$ be the color of e, that is, B if $e \in E_{\rm B}$, W if $e \in E_{\rm W}$, and G if $e \in E_{\rm G}$. If $\{u, v\} \notin E_{\rm B} \cup E_{\rm W} \cup E_{\rm G}$, we set $c(\{u, v\}) = \emptyset$ for convenience. Since the first (black or B) player can choose black or gray edges, and the second (white or W) player can choose white or gray edges, COLORED ARC KAYLES is a partisan game. Note that COLORED ARC KAYLES with empty E_{B} and E_{W} is actually ARC KAYLES, which is no longer a partisan and is said to be *impartial*. We also name COLORED ARC KAYLES with empty E_{G} BW-ARC KAYLES, which is still partisan. This paper presents an fixed-parameter tractable (FPT) winner-determination algorithm also for COLORED ARC KAYLES, which is parameterized by vertex cover number.

Here, we introduce another combinatorial game called DOMINEERING. DOMINEERING is a partisan version of CRAM; one player can place a domino only vertically, and the other player can place one only horizontally. As ARC KAYLES is a graph generalization of CRAM, BW-ARC KAYLES is a graph generalization of DOMINEERING. Note that DOMINEERING is also a well-studied combinatorial game. In fact, several books of combinatorial game theory (e.g., [1]) use DOMINEERING as a sample of partisan games, though its time complexity is still unknown as well as CRAM. Our algorithm mentioned above works for DOMINEERING.

1.2 Related work

Node Kayles and Arc Kayles As mentioned above, NODE KAYLES and ARC KAYLES were introduced in [12]. NODE KAYLES is the vertex version of ARC KAYLES; the action of a player in NODE KAYLES is to select a vertex instead of an edge, and then the selected vertex and its neighboring vertices are removed. The winner determination of NODE KAYLES is known to be PSPACE-complete in general [12], though it can be solved in polynomial time by using Sprague-Grundy theory [2] for graphs of bounded asteroidal numbers, such as comparability graphs and cographs. For general graphs, Bodlaender et al. propose an $O(1.6031^n)$-time algorithm [3]. Furthermore, they show that the winner of NODE KAYLES can be determined in time $O(1.4423^n)$ on trees. In [9], Kobayashi sophisticates the analysis of the algorithm in [3] from the perspective of the parameterized complexity and shows that it can be solved in time $O^*(1.6031^\mu)$, where μ is the modular width of an input graph[1]. He also gives an $O^*(3^\tau)$-time algorithm, where τ is the vertex cover number, and a linear kernel when parameterized by neighborhood diversity.

Different from NODE KAYLES, the complexity of ARC KAYLES has remained open for more than 30 years. Even for subclasses of trees, not much is known. For example, Huggans and Stevens study ARC KAYLES on subdivided stars with three paths [8]. To our best knowledge, until a few years ago no exponential-time algorithm for ARC KAYLES is presented except for an $O^*(4^{\tau^2})$-time algorithm proposed in [11]. In [7,17], the authors show that the winner determination of ARC KAYLES on trees can be solved in $O^*(2^{n/2}) = O(1.4143^n)$ time, which improves $O^*(3^{n/3})(= O(1.4423^n))$ by a direct adjustment of the analysis of Bodlaender et al.'s $O^*(3^{n/3})$-time algorithm for Node Kayles.

[1] The $O^*(\cdot)$ notation suppresses polynomial factors in the input size.

BW-Arc Kayles and Colored Arc Kayles BW-Arc Kayles and Colored Arc Kayles are introduced in [7,17]. The paper presents an $O^*(1.4143^{\tau^2+3.17\tau})$-time algorithm for Colored Arc Kayles, where τ is the vertex cover number. The algorithm runs in time $O^*(1.3161^{\tau^2+4\tau})$ and $O^*(1.1893^{\tau^2+6.34\tau})$ for BW-Arc Kayles, and Arc Kayles, respectively. This is faster than the previously known time complexity $O^*(4^{\tau^2})$ in [11]. They also give a bad instance for the proposed algorithm, which implies the running time analysis is asymptotically tight. Furthermore, they show that the winner of Arc Kayles can be determined in time $O^*((n/\nu+1)^\nu)$, where ν is the neighborhood diversity of an input graph. This analysis is also asymptotically tight.

Cram and Domineering Cram and Domineering are well-studied in the field of combinatorial game theory. In [6], Gardner gives winning strategies for some simple cases. For Cram on an $a \times b$ board, the second player can always win if both a and b are even, and the first player can always win if one of a and b is even and the other is odd. This can be easily shown by the so-called Tweedledum and Tweedledee strategy. For specific sizes of boards, computational studies have been conducted [15]. In [14], Cram's endgame databases for all board sizes with at most 30 squares are constructed. As far as the authors know, the complexity to determine the winner for Cram on general boards still remains open.

Finding the winning strategies of Domineering for specific sizes of boards by using computer programs is well studied. For example, the cases of 8×8 and 10×10 are solved in 2000 [4] and 2002 [5], respectively. The first player wins in both cases. Currently, the status of boards up to 11×11 is known [13]. In [16], endgame databases for all single-component positions up to 15 squares for Domineering are constructed. The complexity of Domineering on general boards also remains open. Lachmann, Moore, and Rapaport show that the winner and a winning strategy of Domineering on $m \times n$ board can be computed in polynomial time for $m \in \{1, 2, 3, 4, 5, 7, 9, 11\}$ and all n [10].

1.3 Our contribution

In this paper, we present FPT winner-determination algorithms with the minimum vertex cover number τ as a parameter, which is much faster than the existing ones. To this end, we show that Colored Arc Kayles has a polynomial kernel parameterized by τ, which leads to a $2^{O(\tau \log \tau)} n^{O(1)}$ time algorithm where n is the number of the vertices (Section 3); this improves the previous time complexity $2^{O(\tau^2)} n^{O(1)}$. For Arc Kayles on trees, we show that the winner determination can be done in time $O^*(5^{\tau/2}) (= O(2.2361^\tau))$ (Section 4), together with an elaborate analysis of time $O^*(7^{n/6}) (= O(1.3831^n))$, which improves the previous bound $O^*(2^{n/2}) (= O(1.4142^n))$ (Section 5). Finally, Section 6 shows that BW-Arc Kayles is NP-hard, and thus so is Colored Arc Kayles. Note that this might be the first hardness result on the family of the combinatorial games shown in Section 1.2 except for Node Kayles.

2 Preliminaries

Let $G = (V, E)$ be an undirected graph. We denote $n = |V|$ and $m = |E|$, respectively. For an edge $e = \{u, v\} \in E$, we define $\Gamma(e) = \{e' \mid e \cap e' \neq \emptyset\}$. For a graph $G = (V, E)$ and a vertex subset $V' \subseteq V$, we denote by $G[V']$ the subgraph induced by V'. For simplicity, we denote $G - v$ instead of $G[V \setminus \{v\}]$. For an edge subset E', we also denote by $G - E'$ the subgraph obtained from G by removing all edges in E' from G. A vertex set S is called a *vertex cover* if $e \cap S \neq \emptyset$ for every edge $e \in E$. Let τ denote the size of a minimum vertex cover of G, which is also called the *vertex cover number* of G.

3 A Polynomial Kernel for Colored Arc Kayles

Our main result in this section is that Colored Arc Kayles admits a polynomial kernel when parameterized by the size τ of a given vertex cover. Since Colored Arc Kayles generalizes standard Arc Kayles and our kernelization algorithm proceeds by deleting edges of the input graph, we obtain the same result for Arc Kayles.

Before we proceed, let us give some intuition about the main idea. To make things simpler, let us first consider (standard) Arc Kayles parameterized by the size of a vertex cover τ. One way in which we could hope to obtain a kernel could be via the following observation: if a vertex $x \in C$, where C is the vertex cover, has high degree (say, degree at least $k+1$), then we can guarantee that this vertex can always be played, or more precisely, that it is impossible to eliminate this vertex by playing on edges incident on its neighbors, since the game cannot last more than τ rounds. One could then be tempted to argue that, therefore, when a vertex has sufficiently high degree, we can delete one of its incident edges. If we thus bound the maximum degree of vertices of C, we obtain a polynomial kernel.

Unfortunately, there is a clear flaw in the above intuition: suppose that x is a high-degree vertex of C as before, xy an edge, and $x'y$ another edge of the graph, for $x' \in C$. If x' is a low-degree vertex, then deciding whether to play xy or another edge incident on x is consequential, as the player needs to decide whether the strategy is to eliminate x' by playing one of its incident edges, or by playing edges incident on its neighbors. We therefore need a property more subtle than simply a vertex that has high degree.

To avoid the flaw described in the previous paragraph, we therefore look for a dense sub-structure: a set of vertices $X \subseteq C$ such that there exists a set of vertices $Y \subseteq V \setminus C$ where all vertices of X have many neighbors in Y and at the same time vertices of Y have no neighbors outside X. In such a structure the initial intuition does apply: playing an edge xy with $x \in X$ and $y \in Y$ is equivalent to playing any other edge xy' with $y' \in Y$, because other vertices of X "don't care" which vertices of Y have been eliminated (since vertices of X have high degree), while vertices outside of X "don't care" because they are not connected to Y. Our main technical tool is then to give a definition (Definition 1) which captures and generalizes this intuition to the colored version of the game: we are looking for sets $X_W, X_B \subseteq C$ and $Y \subseteq V \setminus C$, such that each vertex of X_W and X_B has many edges playable by White or Black respectively with the other endpoint in Y, while edges incident on Y have their other endpoint in some appropriate part of $X_W \cup X_B$ (white edges in X_W, black edges in X_B, and gray edges in $X_W \cap X_B$). We show that if we can find such a structure, then we can safely remove an edge (Lemma 1) and then show how in polynomial time we can either find such a structure or guarantee that the size of the graph is bounded to obtain the main result (Theorem 1).

Definition 1. *Let $G = (V, E)$ be an instance of* Colored Arc Kayles*, with $E = E_W \cup E_B \cup E_G$, $C \subseteq V$ be a vertex cover of G of size τ, and $I = V \setminus C$. Then, for three sets of vertices X_W, X_B, Y we say that (X_W, X_B, Y) are a* dense triple *if we have the following: (i) $X_W, X_B \subseteq C$ and $Y \subseteq I$ (ii) for each $x \in X_W$ (respectively $x \in X_B$) there exist at least $\tau + 1$ edges in $E_W \cup E_G$ (respectively in $E_B \cup E_G$) incident on x with the other endpoint in Y (iii) for all $y \in Y$ all edges of E_G incident on y have their second endpoint in $X_W \cap X_B$ (iv) for all $y \in Y$ all edges of E_W (respectively of E_B) incident on y have their second endpoint in X_W (respectively in X_B).*

Lemma 1. *Let G, C, I, τ be as in Definition 1 and (X_W, X_B, Y) be a dense triple of G. Then, for any edge $e \in E$ incident on a vertex of Y and any $r > 0$ we have the following: a player (Black of White) has a strategy to win Arc Kayles in G in at most r moves if and only if the same player has a strategy to win Arc Kayles in $G - e$ in at most r moves.*

Proof. We prove the lemma by induction on the size τ of C. For $\tau = 0$ the lemma is vacuous, as there are no edges to delete. We therefore start with $\tau = 1$, so C contains a single vertex, say $C = \{x\}$. Suppose without loss of generality that Black is playing first (the other case is symmetric). If $X_B = \emptyset$ and $X_W = \emptyset$, then Y may only contain isolated vertices, so again there is no edge e that satisfies the conditions of the lemma, so the claim is vacuous. If $X_B = \emptyset$ and $X_W = \{x\}$, then any e that satisfies the conditions of the lemma must have $e \in E_W$. Clearly, deleting such an edge does not affect the answer, as this edge cannot be played. If on the other hand, $X_B = \{x\}$, then $|E_B| + |E_G| \geq 2$ (to satisfy condition (ii)), hence the current instance is a win for Black in one move, and removing any edge does not change this fact.

For the inductive step, suppose the lemma is true for all graphs with vertex cover at most $\tau - 1$. We must prove that optimal strategies are preserved in both directions. To be more precise, the optimal strategy of a player is defined as the strategy which guarantees that the player will win in the minimum number of rounds, if the player has a winning strategy or guarantees that the game will last as long as possible if the player has no winning strategy.

For the easy direction, suppose that the first player has an optimal strategy in $G - e$ which starts by playing an edge $f = ab$. This edge also exists in G, so we formulate a strategy in G that is at least as good for the first player by again initially playing f in G. Now, let $G_1 = G - \{a, b\}$ be the resulting graph, and $G_2 = G - e - \{a,b\}$ be the resulting graph when we play in $G - e$. If e has an endpoint in $\{a,b\}$, then G_1, G_2 are actually isomorphic, so clearly the first player's strategy in G is at least as good as her strategy in $G - e$ and we are done. Otherwise, $G_2 = G_1 - e$ and we claim that we can apply the inductive hypothesis to G_1 and G_2, proving that the two graphs have the same winner in the same number of moves and hence our strategy is winning for G. Indeed, G_1 has a vertex cover of size at most $\tau - 1$. Furthermore, if (X_W, X_B, Y) is a dense triple of G, then $(X_W \setminus \{a,b\}, X_B \setminus \{a,b\}, Y \setminus \{a,b\})$ is a dense triple of G_1, because Y contains at most one vertex from $\{a,b\}$, hence each vertex of X_W, X_B has lost at most one edge connecting it to Y. Therefore, the inductive hypothesis applies, as $G_2 = G_1 - e$ and e is an edge incident on $Y \setminus \{a,b\}$.

For the more interesting direction, suppose that the first player has an optimal strategy in G for which we consider several cases:

1. The optimal strategy in G initially plays an edge f that shares no endpoints with e.
2. The optimal strategy in G initially plays an edge f that shares exactly one endpoint with e.
3. The optimal strategy in G initially plays e.

For the first case, let G_1 be the graph resulting from playing f in G, and G_2 be the graph resulting from playing f in $G - e$. Again, as in the previous direction, we observe that we can apply the inductive hypothesis on G_1, G_2, and therefore playing f is an equally good strategy in $G - e$.

For the second case, it is even easier to see that playing f is an equally good strategy in $G - e$, as G_1, G_2 are now isomorphic (playing f in G removes the edge e that distinguishes G from $G - e$).

Finally, for the most interesting case, suppose without loss of generality that Black is playing first in G and has an optimal strategy that begins by playing e, therefore $e \in E_B \cup E_G$. Let $e = xy$ with $x \in X_B$ and $y \in Y$. We will attempt to find an equally good strategy for Black in $G - e$. By condition (ii) of Definition 1, x has $\tau > 1$ other incident edges that Black can play, whose second endpoint is in Y. Let $e' = xy'$ be such an edge, with $y' \in Y$. Let $G_1 = G - \{x,y\}$ and $G_2 = G - \{x,y'\}$. It is sufficient to prove that G_1 and G_2 have the same winner in the same number of moves, if White plays first on both graphs. For this, we will again apply the inductive hypothesis, though this time it will be slightly more complicated, since G_1, G_2 may differ in many edges. We will work around this difficulty by *adding* (rather than removing) edges to both graphs, so that we eventually arrive at isomorphic graphs, without changing the winner.

Take G_1 and observe that $(X_W \setminus \{x\}, X_B \setminus \{x\}, Y \setminus \{y\})$ is a dense triple, as the vertex cover of G_1 has size at most $\tau - 1$, and each vertex of $X_B \cup X_W$ has lost at most one neighbor in Y. Add the vertex y to G_1 as an isolated vertex (this clearly does not affect the winner). Furthermore, $(X_W \setminus \{x\}, X_B \setminus \{x\}, Y)$ is a dense triple of the new graph. We now observe that adding a white edge from y to $X_W \setminus \{x\}$, or a black edge from y to $X_B \setminus \{x\}$, or a gray edge from y to $(X_W \cap X_B) \setminus \{x\}$ does not affect the fact that $(X_W \setminus \{x\}, X_B \setminus \{x\}, Y)$ is a dense triple. Hence, by inductive hypothesis, it does not affect the winner or the number of moves needed to win. Repeating this, we add to G_1 all the edges incident on y in G_2. We then take G_2, add to it y' as an isolated vertex, and then use the same argument to add to it all edges incident to y' in G_1 without changing the winner. We have thus arrived at two isomorphic graphs. □

Theorem 1. *There is a polynomial time algorithm which takes as input an instance G of* Colored Arc Kayles *and a vertex cover of G of size τ and outputs an instance G', such that G' has $O(\tau^3)$ edges, and for all $r > 0$ a player (Black or White) has a strategy to win in r moves in G if and only if the the same player has a strategy to win in r moves in G'. Hence,* Colored Arc Kayles *admits a kernel with $O(\tau^3)$ edges.*

Proof. We describe an algorithm that finds a dense triple, if one exists, in the input graph $G = (V, E)$. If we find such a triple, we can invoke Lemma 1 to delete an edge from the graph, without changing the answer, and then repeat the process. Otherwise, we will argue that the G must already have the required number of edges. We assume that we are given a vertex cover C of G of size $\tau \geq 1$ and $I = V \setminus C$. If not, a 2-approximate vertex cover can be found in polynomial time using standard algorithms.

The algorithm executes the following rules exhaustively, until no rule can be applied, always preferring to apply lower-numbered rules.

1. If C contains an isolated vertex, delete it.
2. If there exists $x \in C$ such that x is incident on at most τ edges of $E_B \cup E_G$ and at most τ edges of $E_W \cup E_G$, then delete $N(x) \cap I$ from G.
3. If there exists $x \in C$ such that x is incident on at least 1 and at most τ edges of $E_B \cup E_G$, then for each $y \in I$ such that $xy \in E_B \cup E_G$, delete y from G.
4. If there exists $x \in C$ such that x is incident on at least 1 and at most τ edges of $E_W \cup E_G$, then for each $y \in I$ such that $xy \in E_W \cup E_G$, delete y from G.

The rules above can clearly be executed in polynomial time. We now first prove that the rules are safe via the following two claims.

Claim. If G contains a dense triple (X_W, X_B, Y), then applying any of the rules will result in a graph where (X_W, X_B, Y) is still a dense triple (in particular, the rules will not delete any vertex of $X_W \cup X_B \cup Y$).

Proof. It is in fact sufficient to prove that the rules will never delete a vertex of $X_W \cup X_B \cup Y$, because if we only delete vertices outside a dense triple, the dense triple remains valid. Vertices removed by the first rule clearly cannot belong to $X_B \cup X_W$. For the second rule, we observe that if x satisfies the conditions of the rule, then $x \notin X_W \cup X_B$, as that would violate condition (ii) of Definition 1. Since $x \notin X_W \cup X_B$, for any $y \in I$ such that $xy \in E$, it must be the case that $y \notin Y$, therefore it is safe to delete such vertices. For the third rule, we observe that $x \notin X_B$, because that would violate condition (ii) of Definition 1. Therefore, if $y \in I$ such that $xy \in E_B \cup E_G$, we have $y \notin Y$ by conditions (iii) and (iv) of Definition 1, and it is safe to delete such vertices. The last rule is similar. □

Claim. If after applying the rules exhaustively, the resulting graph is not edge-less, then we can construct a dense triple (X_W, X_B, Y) as follows: place into X_W (respectively into X_B) all remaining vertices of C which are still incident on an edge of E_W (respectively of E_B), place all vertices of C still incident on an edge of E_G into both X_W and X_B, and place all remaining vertices of I into Y. The dense triple thus constructed is also a dense triple in the original graph.

Proof. We prove the claim by induction on the number of rule applications. Let $G_0 = G, G_1, G_2, \ldots, G_\ell$ be the sequence of graphs we obtain by executing the algorithm, where G_{i+1} is obtained from G_i by applying a rule. We first show that (X_W, X_B, Y) is a dense triple in the final graph G_ℓ. Consider a vertex $x \in X_W \setminus X_B$. By construction x is incident on an edge of E_W in G_ℓ but on no edge of $E_B \cup E_G$. We can see that x satisfies condition (ii) of Definition 1 because if it were incident on at most k edges of E_W, the second rule would have applied. Similarly, vertices of $X_B \setminus X_W$ satisfy condition (ii). For $x \in X_W \cap X_B$, by construction either x is incident on an edge of E_G or it is incident on edges from both E_W and E_B. Therefore, x is incident on at least 1 edge of $E_W \cup E_G$ and at least 1 edge of $E_B \cup E_G$. As a result, if x violated condition (ii), the third or fourth rules would have applied. Condition (iii) is satisfied because we placed all vertices of C incident on an edge of E_G into $X_W \cap X_B$. Condition (iv) is satisfied because we placed all vertices of C incident on an edge of E_W into X_W (similarly for E_B).

Having established the base case, suppose we have some $r < \ell$ such that (X_W, X_B, Y) is a dense triple in all of $G_{r+1}, \ldots, G_\ell$. We will show that (X_W, X_B, Y) is also a dense triple in G_r. If G_{r+1} is obtained from G_r by applying the first rule, this is easy to see, as adding an isolated vertex to G_{r+1} does not affect the validity of the dense triple. If on the other hand, we obtained G_{r+1} by applying one of the other rules, then we deleted from G_r some vertices of I. However, adding to G_{r+1} some vertices to I does not affect the validity of the dense triple, as the vertices of Y do not obtain new neighbors (hence conditions (iii) and (iv) remain satisfied), while condition (ii) is unaffected. We conclude that the constructed triple is valid in G. □

The last claim shows how to construct a dense triple in G if after applying the rules exhaustively the remaining graph is not edge-less. The kernelization algorithm is then the following: apply the rules exhaustively. When this is no longer possible, if the remaining graph is not edge-less, construct a dense triple and invoke Lemma 1 to remove an arbitrary edge of that triple. Run the kernelization algorithm on the remaining graph and return the result. Otherwise, if the graph obtained after applying all the rules is edge-less, we return the initial graph G.

What remains is to prove that when the kernelization algorithm ceases to make progress (that is, when applying all rules produces an edge-less graph), this implies that the given graph must have $O(\tau^3)$ edges. To see this, observe that to apply any rule, we need a vertex $x \in C$ which satisfies certain conditions. Once we apply that rule to x, the same rule cannot be applied to x a second time, because we delete an appropriate set of its neighbors. As a result, the algorithm will perform $O(\tau)$ rule applications. Each rule application deletes either an isolated vertex or at most $O(\tau)$ vertices of I. Each vertex of I is incident on $O(\tau)$ edges (since the other endpoint of each such edge must be in C). Therefore, each rule application removes $O(\tau^2)$ edges from the graph and after $O(\tau)$ rule applications we arrived at an edge-less graph. We conclude that the given graph contained $O(\tau^3)$ edges. □

Corollary 1. COLORED ARC KAYLES *can be solved in time* $\tau^{O(\tau)} + n^{O(1)}$ *on graphs on* n *vertices, where* τ *is the size of a minimum vertex cover of the input graph.*

Proof. Suppose that we have a vertex cover C of size τ (otherwise one can be found with standard FPT algorithms in the time allowed). We first apply the algorithm of Theorem 1 in polynomial time to reduce the graph to $O(\tau^3)$ edges. Then, we apply the simple brute force algorithm which considers all possible edges to play for each move. Since the game cannot last for more than τ rounds (as each move decreases the vertex cover), this results in a decision tree of size $\tau^{O(\tau)}$. □

Finally, a corollary of the above results is that ARC KAYLES also admits a polynomial kernel when parameterized by the number of rounds. This follows because the first player has a strategy to win in a small number of rounds only if the graph has a small vertex cover. Notice that this corollary cannot automatically apply to the colored version of the game, because if Black has a strategy to win in a small number of rounds, this only implies that the graph induced by the edge of $E_W \cup E_G$ (that is, the edges playable by White) has a small vertex cover.

Corollary 2. ARC KAYLES *admits a kernel of* $O(r^3)$ *edges and can be solved in time* $r^{O(r)} + n^{O(1)}$*, where the objective is to decide if the first player has a strategy to win in at most* r *rounds.*

Proof. Given an instance of ARC KAYLES G we first compute a maximal matching of G. If the matching contains at least $2r + 1$ edges, then we answer no, as the game will go on for at least $r + 1$ rounds, no matter which strategy the players follow. Otherwise, by taking both endpoints of all edges in the matching we obtain a vertex cover of size at most $4r$, and we can apply Theorem 1 and Corollary 1. □

4 Arc Kayles for Trees Parameterized by Vertex Cover Number

In [3], Bodlaender et al. showed that the winner of NODE KAYLES on trees can be determined in time $O^*(3^{n/3}) = O(1.4423^n)$. Based on the algorithm of Bodlaender et al., Hanaka et al. showed an $O^*(2^{n/2}) = O(1.4143^n)$ time algorithm to determine the winner of Arc Kayles and Node Kayles on trees in [7]. This improvement is achieved by not considering the ordering of subtrees. Now, we show that the improved algorithm in [7] also runs in time $O^*(5^{\tau/2}) = O(2.2361^\tau)$, where τ is the vertex cover number.

We start with an introduction to the algorithm. The algorithm is based on the algorithm for Node Kayles of Bodlaender et al. [3], which uses the Sprague–Grundy theory. Any position of a game can be assigned a non-negative integer called nimber. 0 is assigned to a position P if and only if the second player wins in P in the game. Thus, in Arc Kayles nimber of a graph G is 0 when G has no edge. When a graph has some edges, we calculate $mex(S)$. $mex(S)$ is the smallest non-negative integer which is not contained in S, where S is the set of non-negative integers. In a general game, for a position P where the winner is not trivial, S consists of nimbers of positions reachable from P in one move, and the nimber of P is $mex(S)$. Thus, in Arc Kayles a nimber of a graph G with some edges is $mex(S)$, where S is the set of the nimbers of graphs which are reachable from G in one move. In addition, when the graph G is unconnected, the nimber of G can be obtained by computing XOR of the nimbers for each connected component.

The algorithm to determine the winner for Arc Kayles on trees using Sprague–Grundy theory is as follows: Like a DFS, we calculate the nimber of input graph by calculating the nimbers of graphs which are reachable from input graph in one move, and so on. Once the position has been examined, the calculation result is held and is not calculated again. In memoization, each connected components of a tree is memorized and when for any vertex only the order of its children is different, it is regarded as the same tree.

The exponential part of the running time of the algorithm depends on the number of connected components that can be played in the game. When we play Arc Kayles with a input graph T, which is a tree and the vertex cover number of T is τ, we claim that the number of connected components that can be played in the game is $O^*(5^{\tau/2}) = O(2.2361^\tau)$ (See appendix for details).

Theorem 2. *The winner of* Arc Kayles *on a tree whose vertex cover number is τ can be determined in time $O^*(5^{\tau/2})(= O(2.2361^\tau))$.*

5 Arc Kayles for Trees

Continued from section 4, we further analyze the winner determination algorithm in [7] for Arc Kayles on trees. In [7], Hanaka et al. showed an $O^*(2^{n/2}) = O(1.4143^n)$-time algorithm to determine the winner of Arc Kayles and Node Kayles on trees, and we gave another running time of the algorithm of [7] with respect to vertex cover number in section 4. Now, we improve the estimation of the running time of the algorithm and show that the winner of Arc Kayles and Node Kayles on trees can be determined in time $O^*(7^{n/6})(= O(1.3831^n))$.

Theorem 3. *The winner of* Arc Kayles *on a tree with n vertices can be determined in time $O^*(7^{n/6}) = O(1.3831^n)$.*

Theorem 4. *The winner of* Node Kayles *on a tree with* n *vertices can be determined in time* $O^*(7^{n/6})(= O(1.3831^n))$.

6 NP-hardness of BW-Arc Kayles

The complexity to determine the winner of combinatorial games is expected to be PSPACE-complete, though no hardness results are known for (Colored) Arc Kayles so far. In this section, we prove that Bw-Arc Kayles is NP-hard.

Theorem 5. BW-Arc Kayles *is NP-hard.*

Proof. We give a polynomial-time reduction from Vertex Cover, which is the problem to decide whether G has a vertex cover of size at most τ. Let $G = (V, E)$ and τ be an instance of Vertex Cover. Now we construct an edge-colored graph G' from G such that the first black player has a winning strategy on G' if and only if G has a vertex cover of size at most τ.

We construct G' as follows. The graph G' consists of three layers as shown in Figure 1. The bottom layer corresponds to $G = (V, E)$; the vertex and edge sets are copies of V and E, which we call with the same name V and E. The edges in E are colored in white. The middle layer is a clique with size $2\tau - 1$, where the vertex set is $U = \{u_1, \ldots, u_{2\tau-1}\}$ and all edges are colored in black. The top layer consists of two vertex sets $B = \{b_1, \ldots, b_{2\tau-1}\}$ and $W = \{w_1, \ldots, w_{2\tau-1}\}$, where they are independent. The bottom and middle layers are completely connected by black edge set $E_{V,U} = \{\{v, u\} \mid v \in V, u \in U\}$. The middle and top layers are connected by black edge set $E_{U,B} = \{\{u_i, b_i\} \mid i = 1, \ldots, 2\tau - 1\}$ and white edge set $E_{U,W} = \{\{u_i, w_i\} \mid i = 1, \ldots, 2\tau - 1\}$.

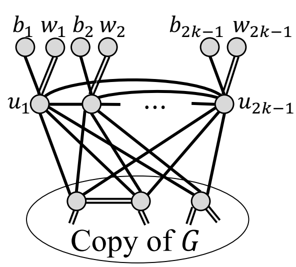

Fig. 1. graph G'

Let S be a vertex cover of G of size τ. For S, we define $E_{S,U} = \{\{v, u\} \in E_{V,U} \mid v \in S\}$. Note that the second (white) player can choose only edges in $E_{U,W}$ or E. The strategy of the first (black) player is as follows. In the first turn, the black player just chooses an edge in $E_{S,U}$. After that, the black player chooses an edge according to which edge the white player chooses right before the black turn. If the white player chooses an edge in $E_{U,W}$, let the black player choose an edge in $E_{S,U}$ in the next black turn. Otherwise, i.e., the white player chooses an edge in E, let the black player choose an edge in the middle layer in the next black turn. This is the strategy of the black player.

We now show that this is a winning strategy for the black player. If the black player following this strategy can choose an edge in every turn right after the white player's action, the black player is the winner. In fact, this procedure continues at most $2\tau - 1$ turns because exactly two vertices in U and at least one vertex in S are removed in every two turns (a white turn and the next black turn) under this strategy; after $2\tau - 1$ turns, no white edge is left and the next player is the white player. Thus what we need to show here is that the black player following this strategy can choose an edge in every turn right after the white player's action. Under this strategy, E can become empty before $2\tau - 1$ turns. In this case, the black player chooses an edge in $E_{U,B}$ instead of an edge in $E_{S,U}$ if the white player chooses an edge $E_{U,W}$. This makes that exactly two vertices in U are removed in every two turns. Then, the black player wins in the same way as above.

Next, we show that the white player has a winning strategy if G does not have a vertex cover of size τ, i.e. $|S| \geq \tau + 1$. The white player can win the game by selecting an edge in $E_{U,W}$ in every turn. Under this strategy, exactly one vertex in U is removed in white player's turn, and then the black player can move at most τ because the size of U is $2\tau - 1$. An edge which black player can choose in his turn is $E_{U,B}$ or $E_{V,U}$, then the black player can remove vertices in S at most τ. Therefore, after $2\tau - 1$ turns there are some vertices and white edges in the bottom layer and there is no black edge because U is empty. The winner is the white player. □

Now, we consider Colored Arc Kayles. Colored Arc Kayles is generalized of Arc Kayles and BW-Arc Kayles; edges are colored black, white and gray, and the black (resp., white) edges are selected by only the black (resp., white) player, while both the black and white players can select gray edges. Since Colored Arc Kayles includes BW-Arc Kayles, we also obtain the following corollary.

Corollary 3. Colored Arc Kayles *is NP-hard.*

References

1. Michael Albert, Richard Nowakowski, and David Wolfe. Lessons in play: An introduction to combinatorial game theory. 2007.
2. Hans L. Bodlaender and Dieter Kratsch. Kayles and nimbers. *Journal of Algorithms*, 43(1):106–119, 2002.
3. Hans L. Bodlaender, Dieter Kratsch, and Sjoerd T. Timmer. Exact algorithms for kayles. *Theoretical Computer Science*, 562:165–176, 2015.

4. D.M. Breuker, J.W.H.M. Uiterwijk, and H.J. van den Herik. Solving 8×8 domineering. *Theoretical Computer Science*, 230(1):195–206, 2000.
5. Nathan Bullock. Domineering: Solving large combinatorial search spaces. *ICGA Journal*, 25(2):67–84, 2002.
6. Martin Gardner. Mathematical games: Cram, crosscram and quadraphage: new games having elusive winning strategies. *Scientific American*, 230(2):106–108, 1974.
7. Tesshu Hanaka, Hironori Kiya, Hirotaka Ono, and Kanae Yoshiwatari. Winner determination algorithms for graph games with matching structures. *Algorithmica*, to appear.
8. Melissa A Huggan and Brett Stevens. Polynomial time graph families for arc kayles. *Integers*, 16:A86, 2016.
9. Yasuaki Kobayashi. On structural parameterizations of node kayles. In *Proceedings of the 21st Japanese Conference on Discrete and Computational Geometry and Graphs (JCDCGGG 2018)*, volume 13034 of *Lecture Notes in Computer Science*, pages 96–105. Springer, 2018.
10. Michael Lachmann, Cristopher Moore, and Ivan Rapaport. Who wins domineering on rectangular boards? *arXiv preprint math/0006066*, 2000.
11. Michael Lampis and Valia Mitsou. The computational complexity of the game of set and its theoretical applications. In *Proceedings of the 11th Latin American Symposium on Theoretical Informatics (LATIN 2014)*, volume 8392 of *Lecture Notes in Computer Science*, pages 24–34. Springer, 2014.
12. Thomas J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16(2):185–225, 1978.
13. Jos WHM Uiterwijk. 11×11 domineering is solved: The first player wins. In *International Conference on Computers and Games*, pages 129–136. Springer, 2016.
14. Jos WHM Uiterwijk. Construction and investigation of cram endgame databases. *ICGA Journal*, 40(4):425–437, 2018.
15. Jos WHM Uiterwijk. Solving cram using combinatorial game theory. In *Advances in Computer Games*, pages 91–105. Springer, 2019.
16. Jos WHM Uiterwijk and Michael Barton. New results for domineering from combinatorial game theory endgame databases. *Theoretical Computer Science*, 592:72–86, 2015.
17. Kanae Yoshiwatari, Hironori Kiya, Tesshu Hanaka, and Hirotaka Ono. Winner determination algorithms for graph games with matching structures. In *International Workshop on Combinatorial Algorithms*, pages 509–522. Springer, 2022.

A Appendix

A.1 Proof of Theorem 2

The exponential part of the running time of the algorithm depends on the number of connected components that can be played in the game. Now we estimate it. Let us consider a tree $T = (V, E)$. By Sprague–Grundy theory, if all connected subtrees of T are enumerated, one can determine the winner of Arc Kayles. Furthermore, once a connected subtree T' is listed, we can ignore subtrees isomorphic to T'.

Proposition 1. *If edge-colored graphs $G^{(1)}$ and $G^{(2)}$ are isomorphic, $G^{(1)}$ and $G^{(2)}$ have the same outcome for* Arc Kayles.

Here we define an isomorphism of rooted trees.

Definition 2. *Let $T^{(1)} = (V^{(1)}, E^{(1)}, r^{(1)})$ and $T^{(2)} = (V^{(2)}, E^{(2)}, r^{(2)})$ be trees rooted at $r^{(1)}$ and $r^{(2)}$, respectively. Then, $T^{(1)}$ and $T^{(2)}$ are called* isomorphic with respect to root *if for any pair of $u, v \in V^{(1)}$ there is a bijection $f : V^{(1)} \to V^{(2)}$ such that $\{u, v\} \in E^{(1)}$ if and only if $\{f(u), f(v)\} \in E^{(2)}$ and $f(r^{(1)}) = f(r^{(2)})$.*

For a tree T rooted at r, two subtrees T' and T'' are simply said *non-isomorphic* if T' with root r and T'' with root r are not isomorphic with respect to root. Now, we estimate the number of non-isomorphic connected subgraphs of T based on the isomorphism of rooted trees. For $T = (V, E)$ rooted at r, a connected subtree T' rooted at r is called an *AK-rooted subtree* of T, if there exists a matching $M \subseteq E$ such that $T[V \setminus \bigcup M]$ consists of T' and isolated vertices, where $\bigcup M$ is a set of endpoints of $e \in M$. Note that M can be empty, AK-rooted subtree T' must contain root r of T, and the graph consisting of only vertex r can be an AK-rooted subtree.

Lemma 2. *Any tree rooted at r of vertex cover number τ has $O^*(5^{\tau/2})(= O(2.2361^\tau))$ non-isomorphic AK-rooted subtrees rooted at r.*

Proof. Let $f(\tau)$ be the maximum number of non-isomorphic AK-rooted subtrees of any tree rooted at some r with vertex cover number τ and r is in a vertex cover, and let $g(\tau)$ be the maximum number of non-isomorphic AK-rooted subtrees of any tree rooted at some r with vertex cover number τ and r is NOT in a vertex cover. We claim that $f(\tau) \le 5^{\tau/2} - 2$ and $g(\tau) \le 5^{\tau/2} - 1$ for all $\tau \ge 2$, which proves the lemma. We will prove the claim by induction.

For $\tau \le 2$, the values of $f(\tau)$'s and $g(\tau)$'s are as follows: $f(1) = 1, f(2) = 3, g(0) = 1, g(1) = 2$, and $g(2) = 4$. These can be shown by concretely enumerating trees. For $\tau = 1$, the candidates of T are shown in Figure 2. For I-f_a in Figure 2, an AK-rooted subtree is the tree itself even if r has any vertices as its children. Thus we have $f(1) = 1$. For I-g_a, AK-rooted subtrees are the tree itself and isolated r, and for I-g_b, an AK-rooted subtree is only the tree itself; thus we have $g(1) = 2$. Similarly, we can show $f(2) = 3$ and $g(2) = 4$. ($g(0)$ is isolated r.)

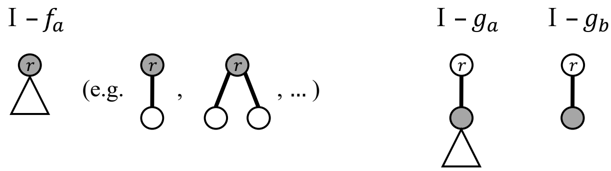

Fig. 2. Trees rooted at r and whose vertex cover number is 1

As the induction hypothesis, let us assume that both of the claims are true for all $\tau' < \tau$ except 0 and 1, and consider a tree T rooted at r and whose vertex cover number is τ. Let $u_1, u_2, \ldots, u_p$ be the children of root r, and T_i be the subtree of T rooted at u_i and whose vertex cover number is τ_i for $i = 1, 2, \ldots, p$.

Note that for an AK-rooted subtree T' of T, the intersection of T' and T_i for each i is either empty or an AK-rooted subtree of T_i rooted at u_i. Based on this observation, we take a combination of the number of AK-rooted subtrees of T_i's, which gives an upper bound on the number of AK-rooted subtrees of T.

First, we will prove the claim $f(\tau) \le 5^{\tau/2} - 2$. In this case, from the definition of $f(\tau)$, r is in a vertex cover and u_i's are not necessary to be in a vertex cover. For $\tau \ge 2$, we estimate AK-rooted subtrees of any T_i. Here, we have that, by the induction hypothesis, $f(\tau) \le 5^{\tau/2} - 2$ and $g(\tau) \le 5^{\tau/2} - 1$ for $\tau \ge 2$. For $\tau = 1$, we have three types of trees, I-f_a, I-g_a, and I-g_b.

As shown in above, the maximum number of AK-rooted subtrees in a I-f_a tree is one, which does not satisfy $f(\tau) \le 5^{\tau/2} - 2$. Here we use another hypothesis $g(\tau) \le 5^{\tau/2} - 1$ because $1 \le 5^{1/2} - 1$. Similarly, we estimate other T_i for $\tau \ge 2$ with $g(\tau) \le 5^{\tau/2} - 1$ from the point of view of simplifying the estimation.

Next, suppose that T has q children of r forming I-g_a. The maximum number of AK-rooted subtree of a I-g_a tree is two and it does not meet the assumption. Since each I-g_a tree can form in T' empty, a single vertex, or I-g_a tree itself, the number of possible forms of subforests of all I-g_a of T is

$$\left(\!\!\binom{q}{3}\!\!\right) = \binom{q+2}{2}.$$

On the other hand, the number of AK-rooted subtrees of a I-g_b tree is one and we can apply the assumption. Then, the number of AK-rooted subtrees of T is at most

$$\binom{q+2}{2} \prod_{i:\tau_i \neq 0,1} (g(\tau_i) + 1) \cdot \prod_{i:T_i \text{ is I}-\mathrm{g_b}} (g(\tau_i) + 1) \le \frac{(q+2)(q+1)}{2} \cdot 5^{\tau - q - 1/2}.$$

Thus, to prove the claim, it is sufficient to show that $\frac{1}{2}(q+2)(q+1)5^{(\tau-q-1)/2} \le 5^{\tau/2} - 2$ for any pair of integers τ and q satisfying $\tau \ge 3$ and $1 \le q$. This inequality is transformed to the following

$$\frac{\frac{1}{2}(q+1)(q+2)}{5^{\frac{q+1}{2}}} \le 1 - \frac{1}{5^{\frac{\tau}{2}}}.$$

Since the left hand and right hand of the inequality are monotonically decreasing with respect to q and monotonically increasing with respect to τ, respectively, the inequality always holds if it is true for $\tau = 3$ and $q = 1$. In fact, we have

$$\frac{\frac{1}{2}(1+1)(1+2)}{5^{\frac{1+1}{2}}} = \frac{3}{5^{\frac{2}{2}}} \leq 1 - \frac{1}{5^{\frac{3}{2}}}.$$

Next, we will prove the claim $g(\tau) \leq 5^{\tau/2} - 1$. From the definition of $g(\tau)$, r is not in a vertex cover, and hence each of u_i's is necessary to be in a vertex cover. Since $f(2) = 3 \leq 5^{\tau/2} - 2$, $f(2)$ is used as the base case of induction. For $\tau = 1$, since each I-f$_a$ tree can form in T' empty or I-f$_a$ tree itself, $T_1, \ldots, T_s$ of T, the number of possible forms of subforests of $T_1, \ldots, T_s$ of T is

$$\left(\!\binom{s}{2}\!\right) = \binom{s+1}{1}.$$

Since the number of subforests of T_i's other than $T_1, \ldots, T_s$ are similarly evaluated as above, we can bound the number of AK-rooted subtrees by

$$\binom{s+1}{1} \prod_{i:\tau_i \neq 0,1} (f(\tau_i) + 1) \leq (s+1)5^{(\tau-s)/2},$$

Thus, to prove the claim, it is sufficient to show that $(s+1)5^{(\tau-s)/2} \leq 5^{\tau/2} - 1$ for any pair of integers τ and s satisfying $n \geq 3$ and $1 \leq s$. This inequality is transformed to the following

$$\frac{s+1}{5^{\frac{s}{2}}} \leq 1 - \frac{1}{5^{\frac{\tau}{2}}}.$$

Since the left hand and right hand of the inequality are monotonically decreasing with respect to s and monotonically increasing with respect to τ, respectively, the inequality always holds if it is true for $\tau = 3$ and $s = 1$. In fact, we have

$$\frac{1+1}{5^{\frac{1}{2}}} \leq 1 - \frac{1}{5^{\frac{3}{2}}}.$$

□

We showed an estimate when the root is fixed. Here, a vertex v is the root means that v is left until the end in the game. Since any vertices can be left to the end in the game, the number of all connected components which can be played in the game is at most $(5^{\tau/2} - 1) \times n$.

A.2 Proof of Theorem 3

First, we discuss Arc Kayles on trees. In the Hanaka et al.'s algorithm, a only tree whose size was three was exceptionally estimated by considering isomorphism. In addition, now we estimate two types of trees exceptionally and then we show that any tree rooted at r has $O^*(7^{n/6}) (= O(1.3831^n))$ non-isomorphic AK-rooted subtrees rooted at r.

Lemma 3. *Any tree rooted at r has $O^*(7^{n/6})(= O(1.3831^n))$ non-isomorphic AK-rooted subtrees rooted at r, where n is the number of the vertices.*

Proof. Let $R(n)$ be the maximum number of non-isomorphic AK-rooted subtrees of any tree rooted at some r with n vertices. We claim that $R(n) \leq 7^{n/6} - 1$ for all $n \geq 5$, which proves the lemma.

We will prove the claim by induction. For $n \leq 5$, the values of $R(n)$'s are as follows: $R(1) = 1, R(2) = 1, R(3) = 2$, $R(4) = 3$, and $R(5) = 4$. These can be shown by concretely enumerating trees. For example, for $n = 2$, a tree T with 2 vertices is unique, and an AK-rooted subtree of T containing r is also unique, which is T itself. For $n = 3$, the candidates of T are shown in Figure 3. For III_a in Figure 3, AK-rooted subtrees are the tree itself and isolated r, and for III_b, an AK-rooted subtree is only the tree itself; thus we have $R(3) = 2$. Similarly, we can show $R(4) = 3$ and $R(5) = 4$ as seen in Figure 3 and Figure 4, respectively. Note that $R(1) = 1 > 7^{1/6} - 1$, $R(2) = 1 > 7^{2/6} - 1$, $R(3) = 2 > 7^{3/6} - 1$, $R(4) = 3 > 7^{4/6} - 1$, and $R(5) = 4 \leq 7^{5/6} - 1$. This $R(5)$ is used as the base case of induction.

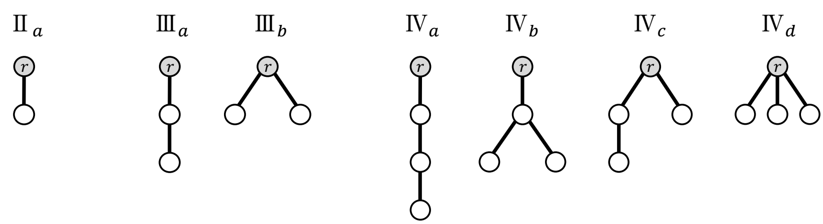

Fig. 3. Trees with 2, 3 and 4 vertices rooted at r

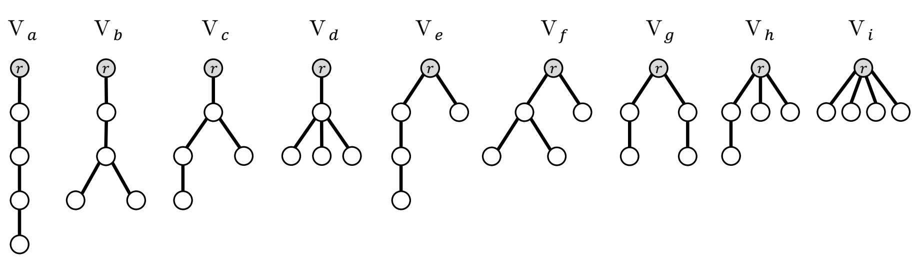

Fig. 4. Trees with 5 vertices rooted at r

As the induction hypothesis, let us assume that the claim is true for all $n' < n$ except 1, 2, 3 and 4, and consider a tree T rooted at r with n vertices. Let $u_1, u_2, \ldots, u_p$ be the children of root r, and T_i be the subtree of T rooted at u_i with n_i vertices for $i = 1, 2, \ldots, p$. Note that for an AK-rooted subtree T' of T, the intersection of T' and T_i for each i is either empty or an AK-rooted subtree of T_i rooted at u_i. Based on this observation, we take a combination of the number of AK-rooted subtrees of T_i's, which gives an upper bound on the number of AK-rooted subtrees of T. We consider eight cases: (1) for any i, $n_i \neq 2, 3, 4$, (2) for any i, $n_i \neq 3, 4$ and for some i, $n_i = 2$, (3) for any i, $n_i \neq 2, 4$ and for some i, $n_i = 3$, (4) for any i, $n_i \neq 2, 3$ and for any i, $n_i = 4$, (5) for any i, $n_i \neq 4$ and for some i, j, $n_i = 2, n_j = 3$, (6) for any i, $n_i \neq 3$ and for some i, j, $n_i = 2, n_j = 4$, (7) for any i, $n_i \neq 2$ and for some i, j, $n_i = 3, n_j = 4$, (8) for any i, $n_i = 2, 3, 4$. For case (1), the number of AK-rooted subtrees of T is at most

$$\prod_{i:n_i>1} (R(n_i)+1) \cdot \prod_{i:n_i=1} 1 \leq \prod_{i:n_i>1} 7^{n_i/6} = 7^{\sum_{i:n_i>1} n_i/6} \leq 7^{(n-1)/6} \leq 7^{n/6} - 1.$$

That is, the claim holds in this case. Here, in the left hand of the first inequality, $R(n_i)+1$ represents the choice of AK-rooted subtree of T_i rooted at u_i or empty, and "1" for i with $n_i = 1$ represents that u_i needs to be left as is because otherwise edge $\{r, u_i\}$ must be removed, which violates the condition "rooted at r". The first inequality holds since any n_i is not 2 or 3 or 4 and thus the induction hypothesis can be applied. The last inequality holds by $n \geq 6$.

For case (2), by the assumption, at least one T_i is II_a in Figure 3. Suppose that T has q children of r forming II_a, which are renamed $T_1, \ldots, T_q$ as canonicalization. Such renaming is allowed because we count non-isomorphic subtrees. Furthermore, we can sort AK-rooted subtrees of $T_1, \ldots, T_q$ as canonicalization. Since each II_a tree can form in T' empty or II_a tree itself, $T_1, \ldots, T_q$ of T, the number of possible forms of subforests of $T_1, \ldots, T_q$ of T is

$$\left(\!\!\binom{q}{2}\!\!\right) = \binom{q+1}{1}.$$

Since the number of subforests of T_i's other than $T_1, \ldots, T_q$ are similar evaluated as above, we can bound the number of AK-rooted subtrees by

$$\binom{q+1}{1} \prod_{i:i>q} 7^{n_i/6} \leq (q+1) 7^{\sum_{i:i>q} n_i/6} \leq (q+1) 7^{(n-2q-1)/6}.$$

Thus, to prove the claim, it is sufficient to show that $(q+1)7^{(n-2q-1)/6} \leq 7^{n/6} - 1$ for any pair of integers n and q satisfying $n \geq 6$ and $1 \leq q \leq (n-1)/2$. This inequality is transformed to the following

$$\frac{q+1}{7^{\frac{2q+1}{6}}} \leq 1 - \frac{1}{7^{\frac{n}{6}}}.$$

Since the left hand and right hand of the inequality are monotonically decreasing with respect to q and monotonically increasing with respect to n, respectively, the inequality always holds if it is true for $n = 6$ and $q = 1$. In fact, we have

$$\frac{1+1}{7^{\frac{2+1}{6}}} = \frac{2}{7^{\frac{1}{2}}} \leq 1 - \frac{1}{7^{\frac{6}{6}}}.$$

For case (3), we further divide into two cases: (3.i), for every i such that $n_i = 3$, T_i is III_b, and (3.ii) otherwise. For case (3.i), since an AK-rooted subgraph of T_i of III_b in Figure 3 is only T_i itself, the number is $1 \leq 7^{3/6} - 1$. Thus, the similar analysis of case (1) can be applied as follows:

$$\prod_{i:n_i \neq 1,2,3,4} (R(n_i)+1) \cdot \prod_{i:T_i \text{ is } \mathrm{III_b}} (7^{3/6} - 1 + 1) \leq \prod_{i:n_i>1} 7^{n_i/6} \leq 7^{n/6} - 1,$$

that is, the claim holds also in case (3.i).

Next, we consider case (3.ii). By the assumption, at least one T_i is III_a in Figure 3. Suppose that T has s children of r forming III_a, which are renamed $T_1, \ldots, T_s$ as canonicalization. Since each III_a tree can form in T' empty, a single vertex, or III_a tree itself, $T_1, \ldots, T_s$ of T, the number of possible forms of subforests of $T_1, \ldots, T_s$ of T is

$$\left(\!\!\binom{s}{3}\!\!\right) = \binom{s+2}{2}.$$

Since the number of subforests of T_i's other than $T_1, \ldots, T_s$ are similarly evaluated as above, we can bound the number of AK-rooted subtrees by

$$\binom{s+2}{2} \prod_{i:i>s} 7^{n_i/6} \leq \frac{(s+2)(s+1)}{2} 7^{\sum_{i:i>s} n_i/6} \leq \frac{(s+2)(s+1)}{2} 7^{(n-3s-1)/6}.$$

Thus, to prove the claim, it is sufficient to show that $\frac{1}{2}(s+2)(s+1)7^{(n-3s-1)/6} \leq 7^{n/6} - 1$ for any pair of integers n and s satisfying $n \geq 6$ and $1 \leq s \leq (n-1)/3$. This inequality is transformed as follows:

$$\frac{\frac{1}{2}(s+1)(s+2)}{7^{\frac{3s+1}{6}}} \leq 1 - \frac{1}{7^{\frac{n}{6}}}.$$

Since the left hand and right hand of the inequality are monotonically decreasing with respect to s and monotonically increasing with respect to n, respectively, the inequality always holds if it is true for $n = 6$ and $s = 1$. In fact, we have

$$\frac{\frac{1}{2}(1+1)(1+2)}{7^{\frac{3+1}{6}}} = \frac{3}{7^{\frac{4}{6}}} \leq 1 - \frac{1}{7^{\frac{6}{6}}}.$$

For case (4), we further divide into two cases: (4.i), for every i such that $n_i = 4$, T_i is IV_b or IV_c or IV_d, and (4.ii) otherwise. For case (4.i), since the AK-rooted subgraphs of T_i of IV_b in Figure 3 are itself and a single vertex, the number is $2 \leq 7^{4/6} - 1$. Similarly, since the AK-rooted subgraphs of T_i of IV_c are itself and a II_a, the number is $2 \leq 7^{4/6} - 1$, and since the AK-rooted subgraphs of T_i of IV_d is itself, the number is $1 \leq 7^{4/6} - 1$. Thus, the similar analysis of Case (1) can be applied as follows:

$$\prod_{i:n_i \neq 1,2,3,4} (R(n_i)+1) \cdot \prod_{i:T_i \text{ is } \mathrm{IV_b},\ \mathrm{IV_c} \text{and } \mathrm{IV_d}} (7^{4/6} - 1 + 1)$$
$$\leq \prod_{i:n_i>1} 7^{n_i/6} \leq 7^{n/6} - 1,$$

that is, the claim also holds in case (4.i).

Next, we consider the case (4.ii). By the assumption, at least one T_i is IV_a in Figure 3. Suppose that T has t children of r forming IV_c or IV_a, which are renamed $T_1, \ldots, T_t$ as canonicalization. Since each IV_c or IV_a tree can form in T' empty, a single vertex, two vertices, or IV_a tree itself, $T_1, \ldots, T_t$ of T, the number of possible forms of subforests of $T_1, \ldots, T_t$ of T is

$$\left(\!\!\binom{t}{4}\!\!\right) = \binom{t+3}{3}.$$

Since the number of subforests of T_i's other than $T_1, \ldots, T_t$ are similarly evaluated as above, we can bound the number of AK-rooted subtrees by

$$\begin{aligned}\binom{t+3}{3} \prod_{i:i>t} 7^{n_i/6} &\le \frac{(t+1)(t+2)(t+3)}{6} 7^{\sum_{i:i>t} n_i/6} \\ &\le \frac{(t+1)(t+2)(t+3)}{6} 7^{(n-4t-1)/6}.\end{aligned}$$

Thus, to prove the claim, it is sufficient to show that $\frac{1}{6}(t+1)(t+2)(t+3)7^{(n-4t-1)/6} \le 7^{n/6} - 1$ for any pair of integers n and t satisfying $n \ge 6$ and $1 \le t \le (n-1)/4$. This inequality is transformed to the following

$$\frac{\frac{1}{6}(t+1)(t+2)(t+3)}{7^{\frac{4t+1}{6}}} \le 1 - \frac{1}{7^{\frac{n}{6}}}.$$

Since the left hand and right hand of the inequality are monotonically decreasing with respect to t and monotonically increasing with respect to n, respectively, the inequality always holds if it is true for $n = 6$ and $t = 1$. In fact, we have

$$\frac{\frac{1}{6}(1+1)(1+2)(1+3)}{7^{\frac{4+1}{6}}} = \frac{4}{7^{\frac{5}{6}}} \le 1 - \frac{1}{7^{\frac{6}{6}}}.$$

By (2), (3), and (4), we get the observation that we have to treat II_a, III_a and IV_a specially. Suppose that T has q children of r forming II_a and has s children of r forming III_a and has t children of r forming IV_a. They are renamed $T_1, \ldots, T_q$, $T_1, \ldots, T_s$, and $T_1, \ldots, T_t$ as canonicalization, respectively. For case (5), (6), (7), and (8), II_a, III_a and IV_a are treated in the same way as (2), (3), and (4) and combined to compute the whole. Note that other trees such that $n = 2, 3, 4$ can be applied to the assumptions.

For case (5), we assume that at least one T_i is II_a and at least one T_j is III_a in Figure 3. Since the number of subforests of T_i's other than $T_1, \ldots, T_q$ and $T_1, \ldots, T_s$ are similar evaluated as above, we can bound the number of AK-rooted subtrees by

$$\begin{aligned}\binom{q+1}{1}\binom{s+2}{2} \prod_{i:i>q} 7^{n_i/6} &\le (q+1)\frac{(s+1)(s+2)}{2} 7^{\sum_{i:i>q} n_i/6} \\ &\le (q+1)\frac{(s+1)(s+2)}{2} 7^{(n-2q-3s-1)/6}.\end{aligned}$$

Thus, to prove the claim, it is sufficient to show that $\frac{1}{2}(q+1)(s+1)(s+2)7^{(n-2q-3s-1)/6} \leq 7^{n/6}-1$ for any pair of integers n, q, and s satisfying $n \geq 6$, $1 \leq q$, and $1 \leq s$. This inequality is transformed to the following

$$\frac{1}{2 \cdot 7^{\frac{1}{6}}} \frac{q+1}{7^{\frac{2q}{6}}} \frac{(s+1)(s+2)}{7^{\frac{3s}{6}}} \leq 1 - \frac{1}{7^{\frac{n}{6}}}.$$

Since the left hand and right hand of the inequality are monotonically decreasing with respect to q and s and monotonically increasing with respect to n, respectively, the inequality always holds if it is true for $n=6$ and $q=1$. In fact, we have

$$\frac{1}{2 \cdot 7^{\frac{1}{6}}} \frac{1+1}{7^{\frac{2}{6}}} \frac{(1+1)(1+2)}{7^{\frac{3}{6}}} = \frac{6}{7^{\frac{6}{6}}} \leq 1 - \frac{1}{7^{\frac{6}{6}}}.$$

For case (6), we assume that at least one T_i is II_a and at least one T_j is I V_a in Figure 3. Since the number of subforests of T_i's other than $T_1, \dots, T_q$ and $T_1, \dots, T_t$ are similar evaluated as above, we can bound the number of AK-rooted subtrees by

$$\begin{aligned}\binom{q+1}{1}\binom{t+3}{3}\prod_{i:i>q} 7^{n_i/6} &\leq (q+1)\frac{(t+1)(t+2)(t+3)}{6}7^{\sum_{i:i>q} n_i/6} \\ &\leq (q+1)\frac{(t+1)(t+2)(t+3)}{6}7^{(n-2q-4t-1)/6}.\end{aligned}$$

Thus, to prove the claim, it is sufficient to show that $\frac{1}{6}(q+1)(t+1)(t+2)(t+3)7^{(n-2q-4t-1)/6} \leq 7^{n/6}-1$ for any pair of integers n, q, and t satisfying $n \geq 7$, $1 \leq q$, and $1 \leq t$. The reason for $n \geq 7$ is that this case can only happen at $n \geq 7$. This inequality is transformed into as follows:

$$\frac{1}{6 \cdot 7^{\frac{1}{6}}} \frac{q+1}{7^{\frac{2q}{6}}} \frac{(t+1)(t+2)(t+3)}{7^{\frac{4t}{6}}} \leq 1 - \frac{1}{7^{\frac{n}{6}}}.$$

Since the left hand and right hand of the inequality are monotonically decreasing with respect to q and t and monotonically increasing with respect to n, respectively, the inequality always holds if it is true for $n=7$ and $q=1$. In fact, we have:

$$\frac{1}{6 \cdot 7^{\frac{1}{6}}} \frac{1+1}{7^{\frac{2}{6}}} \frac{(1+1)(1+2)(1+3)}{7^{\frac{4}{6}}} = \frac{8}{7^{\frac{7}{6}}} \leq 1 - \frac{1}{7^{\frac{7}{6}}}.$$

For case (7), we assume that at least one T_i is III_a and at least one T_j is I V_a in Figure 3. Since the number of subforests of T_i's other than $T_1, \dots, T_s$ and $T_1, \dots, T_t$ are similar evaluated as above, we can bound the number of AK-rooted subtrees by

$$\begin{aligned}\binom{s+2}{2}\binom{t+3}{3}\prod_{i:i>q} 7^{n_i/6} &\leq \frac{(s+1)(s+2)}{2}\frac{(t+1)(t+2)(t+3)}{6}7^{\sum_{i:i>q} n_i/6} \\ &\leq \frac{(s+1)(s+2)}{2}\frac{(t+1)(t+2)(t+3)}{6}7^{(n-3s-4t-1)/6}.\end{aligned}$$

Thus, to prove the claim, it is sufficient to show that $\frac{1}{12}(s+1)(s+2)(t+1)(t+2)(t+3)7^{(n-3s-4t-1)/6} \leq 7^{n/6}-1$ for any pair of integers n, q, and s satisfying $n \geq 8$, $1 \leq q$, and $1 \leq s$. The reason of $n \geq 8$ is that this case can only be happened at $n \geq 8$. This inequality is transformed to the following

$$\frac{1}{12 \cdot 7^{\frac{1}{6}}} \frac{(s+1)(s+2)}{7^{\frac{3s}{6}}} \frac{(t+1)(t+2)(t+3)}{7^{\frac{4t}{6}}} \leq 1 - \frac{1}{7^{\frac{n}{6}}}.$$

Since the left hand and right hand of the inequality are monotonically decreasing with respect to s and t and monotonically increasing with respect to n, respectively, the inequality always holds if it is true for $n=8$, $s=1$, and $t=1$. In fact, we have

$$\frac{1}{12 \cdot 7^{\frac{1}{6}}} \frac{(1+1)(1+2)}{7^{\frac{3}{6}}} \frac{(1+1)(1+2)(1+3)}{7^{\frac{4}{6}}} = \frac{12}{7^{\frac{8}{6}}} \leq 1 - \frac{1}{7^{\frac{8}{6}}}.$$

For case (8), we assume that at least one T_i is II_a and at least one T_i is III_a and at least one T_j is IV_a in Figure 3. Since the number of subforests of T_i's other than $T_1, \ldots, T_s$ and $T_1, \ldots, T_t$ are similarly evaluated as above, we can bound the number of AK-rooted subtrees by

$$\begin{aligned} & \binom{q+1}{1}\binom{s+2}{2}\binom{t+3}{3} \prod_{i:i>q} 7^{n_i/6} \\ & \leq (q+1)\frac{(s+1)(s+2)}{2}\frac{(t+1)(t+2)(t+3)}{6} 7^{\sum_{i:i>q} n_i/6} \\ & \leq (q+1)\frac{(s+1)(s+2)}{2}\frac{(t+1)(t+2)(t+3)}{6} 7^{(n-2q-3s-4t-1)/6}. \end{aligned}$$

Thus, to prove the claim, it is sufficient to show that $\frac{1}{12}(q+1)(s+1)(s+2)(t+1)(t+2)(t+3)7^{(n-2q-3s-4t-1)/6} \leq 7^{n/6}-1$ for any pair of integers n, q, s, and t satisfying $n \geq 10$, $1 \leq q$, $1 \leq s$, and $1 \leq t$. The reason of $n \geq 10$ is that this case can only be happened at $n \geq 10$. This inequality is transformed into the following

$$\frac{1}{12 \cdot 7^{\frac{1}{6}}} \frac{q+1}{7^{\frac{2q}{6}}} \frac{(s+1)(s+2)}{7^{\frac{3s}{6}}} \frac{(t+1)(t+2)(t+3)}{7^{\frac{4t}{6}}} \leq 1 - \frac{1}{7^{\frac{n}{6}}}.$$

Since the left hand and right hand of the inequality are monotonically decreasing with respect to q, s, and t and monotonically increasing with respect to n, respectively, the inequality always holds if it is true for $n=10$, $q=1$, $s=1$, and $t=1$. In fact, we have

$$\frac{1}{12 \cdot 7^{\frac{1}{6}}} \frac{1+1}{7^{\frac{2}{6}}} \frac{(1+1)(1+2)}{7^{\frac{3}{6}}} \frac{(1+1)(1+2)(1+3)}{7^{\frac{4}{6}}} = \frac{24}{7^{\frac{10}{6}}} \leq 1 - \frac{1}{7^{\frac{10}{6}}},$$

which completes the proof. □

Same as Theorem 2, the number of all connected components that can be played in the game is at most $(7^{n/6}-1) \times n$.

A.3 Proof of Theorem 4

We can determine the winner of Node Kayles for a tree in the same running time as Arc Kayles. The outline of the proof is almost the same as Theorem 3. To prove it, We estimate the number of *NK-rooted subtrees* instead of AK-rooted subtrees for Arc Kayles. The definition of an NK-rooted subtree is as follows. For $T = (V, E)$ rooted at r, a connected subtree T' rooted at r is called an *NK-rooted subtree* of T, if there exists an independent set $U \subseteq V$ such that $T[V \setminus N[U]] = T'$.

Lemma 4. *Any tree rooted at r has $O^*(7^{n/6})(= O(1.3831^n))$ non-isomorphic NK-rooted subtrees rooted at r, where n is the number of the vertices.*

To execute the same induction as Lemma 3, we obtain Lemma 4. (The base cases are completely the same.)