Analysis of the computational complexity of the Roma puzzle game

Lonchampt Mathis Supervised by Michail Lampis

Paris Dauphine University

Abstract. This paper provides an analysis of the research article "All roads lead to Rome" by Kevin Goergen, Henning Fernau, Esther Oest, and Petra Wolf [1], which investigates the computational complexity of the puzzle game Roma. Played on an $n \times n$ grid of quadratic cells grouped into boxes of up to four adjacent cells, Roma challenges players to fill empty cells with arrows pointing in cardinal directions. The objective is to ensure that each box contains at most one arrow of each direction, and that following the arrows from any starting point always leads to a special Roma cell. This paper extends the understanding of the computational complexity involved in solving Roma puzzles. The authors demonstrate that completing a Roma board according to its rules is an NP-complete problem. Key contributions include the proof that counting the number of valid solutions is #P-complete, that determining the minimal number of preset arrows needed to ensure a uniquely solvable instance is Σ_2^P-complete, and that solving a given Roma instance on an $n \times n$ board cannot be accomplished in $O(2^{o(n)})$ time under the Exponential Time Hypothesis (ETH) [4]. They complement these theoretical findings with a matching dynamic programming algorithm inspired by Catalan structures. This report elucidates these findings, delves into the methodologies employed, and highlights the identified properties and limitations, offering a comprehensive overview of the significant advancements and challenges in the study of the computational complexity of puzzle games like Roma.

Keywords: Roma Puzzle · Computational Complexity · Exponential Time Hypothesis (ETH) · Dynamic Programming · Catalan Structures

1 Introduction and Fundamental Concepts

1.1 Game and Rules

An instance of the Roma puzzle R is defined on an $n \times n$ grid C with cells $C = \{c_{i,j} \mid i, j \in [n]\}$. The initial configuration includes preset entries specified by a partial function $\rho : C \to \{\circ, \uparrow, \rightarrow, \downarrow, \leftarrow\}$, where only one cell, the Roma-cell $c_R \in C$, is assigned $\circ$, ensuring $|\rho^{-1}(\circ)| = 1$. The remaining cells, forming the set E_R, are initially empty. The grid is divided into boxes given by the set B_R and a total function $\beta : C \to B_R$, where each box contains up to four adjacent

cells. A cell can be added to a non-empty box only if it is a true neighbor of an existing cell in that box.

A solution to an instance is a complete function $\omega : C \to \{\circ, \uparrow, \rightarrow, \downarrow, \leftarrow\}$ that matches ρ where defined and satisfies the following conditions.

From ω, a directed graph $G(\omega) = (V, E)$ is constructed as follows: - $V = C$. - For $c_{i,j}, c_{\ell,k} \in V$, there is an edge $(c_{i,j}, c_{\ell,k}) \in E$ if one of the following holds: - $\ell = i$ and $k = j + 1$ and $\omega(c_{i,j}) = \rightarrow$; - $\ell = i$ and $k = j - 1$ and $\omega(c_{i,j}) = \leftarrow$; - $\ell = i + 1$ and $k = j$ and $\omega(c_{i,j}) = \downarrow$; - $\ell = i - 1$ and $k = j$ and $\omega(c_{i,j}) = \uparrow$.

An assignment ω is valid if it satisfies two conditions:

1. **Box condition**: No box contains more than one arrow of the same direction:

$$\forall c_{i,j}, c_{\ell,k} \in C, (c_{i,j} \neq c_{\ell,k} \wedge \beta(c_{i,j}) = \beta(c_{\ell,k}))\omega(c_{i,j}) \neq \omega(c_{\ell,k}).$$

2. **Graph condition**: The graph $G(\omega)$ is acyclic, weakly connected, and contains a unique vertex of out-degree zero, specifically c_R. This excludes configurations where $\omega(c_{0,0}) = \uparrow$ (since $c_{0,0}$ would have out-degree zero) or where $\omega(c_{0,0}) = \rightarrow$ and $\omega(c_{0,1}) = \leftarrow$ (as the graph would be cyclic or disconnected).

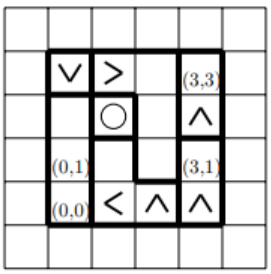

Fig. 1: Example of a 4 × 4 Roma game board

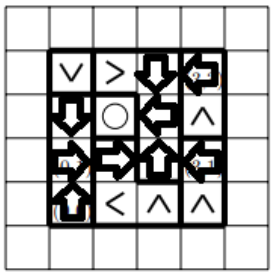

Fig. 2: Solving the Roma puzzle

1.2 Planar 3-SAT

In this section, we provide an overview of the Planar 3-SAT problem, which plays a crucial role in our subsequent proofs.

Definition of Literal A literal ℓ is a propositional variable v_j (positive literal) or the negation of a propositional variable $\neg v_j$ (negative literal).

Definition of Clause A clause is a disjunction of the form $\bigvee_{i=1}^{n} \ell_i = (\ell_1 \vee \ell_2 \vee \ldots \vee \ell_n)$, where the ℓ_i are literals.

Definition of Conjunctive Normal Form (CNF) A formula is in Conjunctive Normal Form (CNF) if it is a conjunction of clauses. Let $C_1, C_2, \ldots, C_n$ be clauses. The conjunction of these clauses, denoted as $\bigwedge_{i=1}^{n} C_i$, is the logical AND of all these clauses: $C_1 \wedge C_2 \wedge \ldots \wedge C_n$.

Definition of SAT Problem The SAT (Boolean satisfiability) problem is to determine whether there exists an assignment of truth values to the propositional variables of a Boolean formula such that the formula evaluates to true.

Definition of 3-SAT Problem The 3-SAT problem is a restriction of the SAT problem to formulas that are in CNF with at most 3 literals per clause.

Lichtenstein's Representation of a 3-SAT Formula The graph $G(\varphi) = (V, E)$ is defined where $V = X \cup C$, with X being the variables of φ and C being its clauses [3]. The edge set E contains two types of edges:

- **Incidence edges:** $xc \in E$ if variable x occurs (either positively or negatively) in clause c.
- **Cycle edges:** $G[X]$ is a cycle.

Definition of Planar 3-SAT A 3-SAT formula φ is planar if the graph $G(\varphi) = (V, E)$ admits an embedding in the plane, i.e., it can be represented without any edge crossing another.

Theorem Planar 3-SAT is an NP-Complete problem.

1.3 Exponential Time Hypothesis (ETH)

The Exponential Time Hypothesis (ETH) posits that there is no $O(2^{o(n)})$-time algorithm for solving 3-SAT, where n is the number of variables.

2 Building a Roma puzzle from Planar 3SAT

In this section, we will show how to construct a Roma puzzle $R(\phi)$ from a given planar 3-SAT formula ϕ in polynomial time. This implicitly assumes a planar embedding of the graph $G(\phi) = (V, E)$. The edges of $G(\phi)$ require the distribution of a signal between the variable and clause gadgets that we describe in the next sections.

Corner Gadget In order to model the edges of the graph in a discretized manner within $R(\phi)$, we require a simple gadget capable of rotating signals by 90 degrees. This necessity leads to the development of a construction known as the corner gadget.

Lemma 1. $\omega(c_{2,0}) = \leftarrow$ *implies* $\omega(c_{0,3}) = \uparrow$ *and* $\omega(c_{0,3}) = \downarrow$ *implies* $\omega(c_{1,0}) = \rightarrow$.

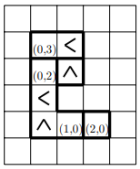

Fig. 3: Illustration of the Corner Gadget (Signal Rotation)

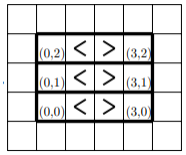

Fig. 4: Illustration of the Straight-line Gadget (Signal along one direction)

Straight-line Gadget We also need to move a signal along in one direction. This can be done with a straight-line gadget, described next.

Lemma 2. *If any cell in a straight-line gadget is validly assigned, there is only one valid way to assign the other empty cells.*

Fanout Gadget Consider now the graph $G(\phi)$. Since the vertices of the graph representing the variables may have a degree greater than 1, we need a gadget able to fan a signal out in order to deliver information regarding a variable to multiple gadgets representing clauses containing the said variable. This is done by using the Fanout gadget.

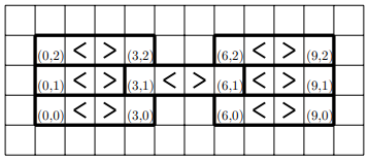

Fig. 5: Illustration of the Fanout Gadget (Size 2)

Lemma 3. *If any empty cell in a fanout gadget is assigned, there is only one valid way to assign the other empty cells.*

Variable Gadget From the two previous gadgets (Fanout and Corner) we build the variable gadget. In $G(\phi)$, all vertices have been connected via a cycle. In our construction, we actually only need a connection via a kind of path which we refer to as the core-line which ensures that the final signal, whatever its starting point, ends up in the Roma cell. Moreover, the cells filled in around this core line simply ensure the uniqueness of the generated instance (parsimonious representation).

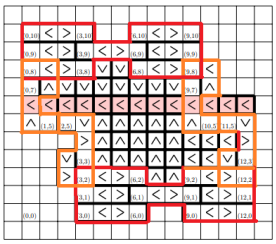

Fig. 6: Illustration of the Variable Gadget

Lemma 4. *There are exactly two ways to fill in the empty cells of the variable gadget. The filled-in arrows on the left side are interpreted as setting the variable to true, while the way the arrows are filled in on the right side should mean that the variable is set to false.*

As the variable and clause gadgets are interconnected through the fanout gadgets, the signal is transmitted through the green parts (see Figure 8 for more details).

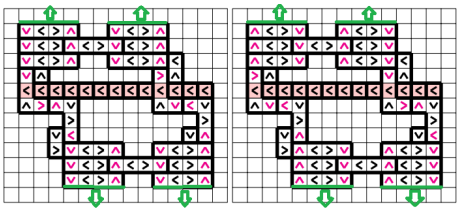

Fig. 7: Illustration of the only two cases where empty cells are filled.

Literal Gadget From the two previous gadgets (Fanout and Corner) we build the literal gadget. The purple arrows represent where we connect the gadget-variables to their literal gadget. The top part of the literal gadget is connected to a loop to form a clause gadget. It is worth noting that if a literal is satisfied, the signal exits the loop.

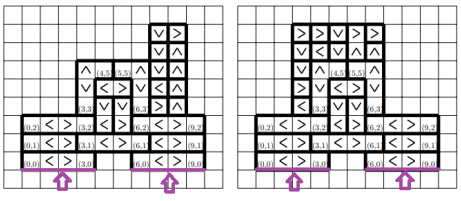

Fig. 8: Literals may occur either positive (on the left) or negated (on the right)

Lemma 5. *There are exactly two ways to fill in the empty cells of the literal gadget (true or false variables being connected to these gadgets).*

Clause Gadget From the previous literal gadget we build the clause gadget by linking a few of them together.

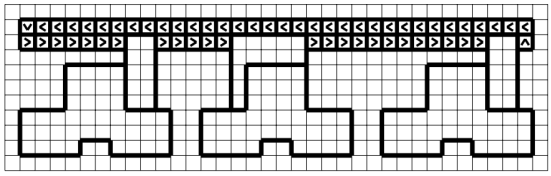

Fig. 9: Illustration of the Clause Gadget consisting of two positive and one negative literal.

Lemma 6. *If the three literal gadgets of the clause gadget are not satisfied, then the signal of the loop will remain inside it, forming a cycle and an inconsistency in the puzzle.*

Planar 3-SAT to Roma With all that has been said above, we can build a Roma puzzle from a Planar 3-SAT formula. Here is a final example of building a Roma puzzle from a Planar 3-SAT:

$$(\neg a_1 \vee \neg\gamma) \wedge (a_1 \vee b_1 \vee \gamma) \wedge (\neg b_1 \vee \neg\gamma)$$

3 Main Complexity Results

3.1 Solving Roma is NP-Complete

The puzzle grid has $n \times n$ cells, totaling n^2 cells. Since each cell can be filled in one of four directions, each cell can be described using 2 bits (e.g., 00 for up, 01 for down, 10 for left, 11 for right). The total number of bits needed to describe a solution is $2n^2$.

Checking a solution can be done in polynomial time: it suffices to check that the graph associated with the solution is acyclic, weakly connected, and contains a unique vertex of out-degree zero, namely c_R, and there is no box to which ω assigns the same arrow twice. Hence, solving Roma is in NP.

Since we have demonstrated that there exists a parcimonius polynomial reduction from Planar 3-SAT to Roma, and solving the constructed instance of Roma is equivalent to solving Planar 3-SAT, we conclude that Roma is NP-hard.

Since NP-Complete = NP $\cap$ NP-hard, we conclude that solving Roma is NP-Complete.

Theorem 1. *Solving Roma is NP-Complete.*

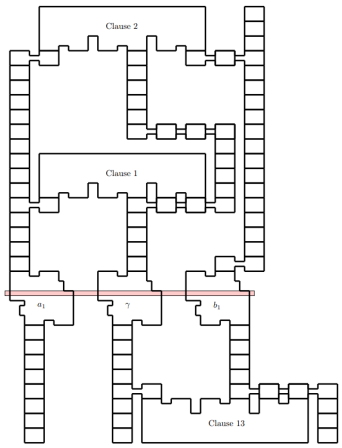

Fig. 10: Illustration of the Roma instance built from the previous Planar 3-SAT formula

3.2 Counting the number of valid completions, Roma is P-Complete

Note: A parsimonious reduction is a type of reduction between computational problems where each instance of the source problem is associated with at most one instance of the target problem. Since in our case each variable and clause in the 3-SAT planar instance are uniquely represented by clause and variable gadgets, our reduction is parsimonious.

Non-deterministic Turing Machine (NTM): A computational model that can explore multiple computation paths simultaneously. It can have several possible transitions from a given configuration to another. It can be in multiple states at once, enabling it to explore various possibilities in parallel.

Proof: We can design a nondeterministic polynomial-time Turing machine that has as many accepting paths as the original Roma instance has solutions.

Conversely, the counting variant of Planar 3-SAT is #P-complete, and the reduction from Planar 3-SAT to Roma is parsimonious. Therefore, the number of satisfying assignments of the 3-SAT formula equals the number of solutions of the constructed Roma instance.

Theorem 2. *#Roma is #P-complete.*

3.3 FCP Roma is Σ_2^P-complete

Definition: The Fewest Clues Problem (FCP) is a computational problem that involves determining the minimum number of clues or pieces of information required to uniquely solve a puzzle or problem. For Roma, it involves determining

the minimum number of preset arrows needed to make the instance uniquely solvable.

Proof:

- We constructed an instance of Roma which is equivalent to a given instance of Planar 3SAT above.
- Instead of a partial assignment of k variables, we now assign k empty cells.
- For each assignment of a variable in the instance of Planar 3SAT reduced from, we need to assign exactly one cell in the resulting instance of Roma, specifically one cell of the corresponding variable-gadget as described above.
- Thus, the number of additional variables to be assigned k is preserved.
- As FCP Planar 3SAT is Σ_2^P-complete, FCP Roma is Σ_2^P-complete.

Theorem 3. *FCP Roma is Σ_2^P-complete.*

4 Algorithm for solving Roma

4.1 First Approach for Solving Roma

A basic initial algorithm for solving Roma consists of testing all possible assignments for each empty cell. Suppose n is the total number of cells and k is the number of empty cells. Each empty cell can receive one of four possible directions, resulting in 4^k combinations. For each combination, we check if it satisfies the puzzle conditions using a connectivity algorithm (such as BFS) in linear time. The complexity of this algorithm is therefore polynomial in n and exponential in k, with a runtime of $O(4^k \cdot n^2)$.

4.2 Search Tree Algorithm for Roma

The second algorithm uses a search tree. For each cell c_{ij}, we first consider the four possible directions. We eliminate directions already assigned to other cells in the same box and those that would create a closed cycle. If only one direction remains valid, we assign it and move to the next cell, repeating the process recursively. In the worst case, this algorithm has a complexity of $O(3.3^k \cdot n^2)$ for puzzles without cells forming boxes of size one.

4.3 Dynamic Programming Approach for Roma

Paper Approach: The algorithm initially considers a straightforward dynamic programming (DP) approach along the rows of a Roma puzzle board. It utilizes a sliding row mechanism steadily progressing downwards. Each row with n squares is associated with a string of length at most $3n$ over the alphabet $\Sigma = \Delta \cup \Delta \times \Delta \cup \Delta \times \Delta 2 \cup B$, where $\Delta = \{\circ, \uparrow, \downarrow, \rightarrow, \leftarrow\}$ represents the alphabet of Roma cell states, and $B = \{[,]\}$ denotes brackets.

Fig. 11: Roma line example

Naive Approach - Arrow Consistency Check

- **Single Symbols from Δ (Type 0 or 3)**:
  - Indicates one of two scenarios for a symbol in a cell:
    1. Type 0: Cell's box does not continue in the successor row.
    2. Type 3: Symbol is the only one fixed for the box, which starts in this sweep row.
- **Pairs from $\Delta \times \Delta$ (Type 1)**:
  - Meaning of a pair (a, b):
    1. a is in the current cell.
    2. b is the only available symbol in that box in the successor row.
- **Combinations from $\Delta \times \Delta 2$ (Type 2)**:
  - Meaning of a combination $(a, \{b, c\})$:
    1. a is in the current cell.
    2. {b, c} represent remaining available symbols in that box in the successor row and possibly beyond.

To understand the updates in the DP procedure, we continue with the example:

- The light blue first row is showing only one possibility of how the previous row (with respect to the sweep row in dark blue) could look like.
- In the last row, a possible successor row is indicated in gray, although the two first gray arrows are enforced by the sweep row.
- The black arrows were given as hints in the very beginning.

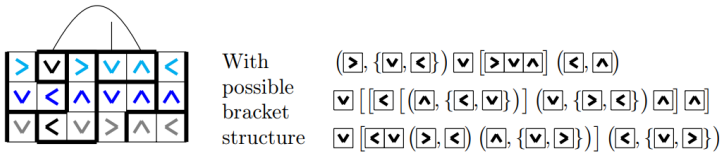

Fig. 12: The sweep row inherits information from its predecessor and passes it to its successor.

How does a successor row arrow consistency check work? We consider the string the last row as a possible successor, one of many that we have to check.

- Arrow Consistency Checks:
  1. The first symbol is ↓ - the only feasible choice:
     - → would contradict with the preset ← in the second cell.

   - $\leftarrow$ points to the left wall, which is not permissible.
   - $\uparrow$ would contradict $\downarrow$ on the sweep row above.
2. The second symbol is a preset $\leftarrow$.
3. The third symbol is $\downarrow$ because:
   - It sees $(\uparrow, \{\leftarrow, \downarrow\})$ above.
   - In the current box, the only symbols available are $\leftarrow$ and $\downarrow$.
   - As $\leftarrow$ was fixed, the symbol $\downarrow$ is enforced.
4. etc.

Paper Approach - Cycle Consistency Check The primary goal of this approach is to prevent the formation of directed cycles in the constructed solution. Assuming the Roma-cell is not in the upper ('forgotten') part of the board, any path that enters this upper zone via a symbol in the sweep row must also exit it via a symbol in the sweep row. These paths can be visualized as a river system, where planarity ensures they never cross, though they might merge, forming river basins. Brackets are used to uniquely describe these basins: the closing bracket is placed to the right of the upward arrow symbol, and the corresponding opening bracket is placed to the right of the downward arrow, indicating where the path exits the upper zone. This bracket structure is crucial for maintaining path consistency across successive rows.

- **Brackets Convention**: The upward arrow points northwest, allowing us to insert a closing bracket to the right of the upward arrow symbol. The corresponding opening bracket is inserted to the right of the downward arrow, indicating where this path exits the upper part again.
- **Specific Examples (Figure 13)**: The second path starts at the penultimate upward arrow, so we insert a closing bracket to the right of the upward arrow symbol. We insert the matching opening bracket to the right of the downward arrow that indicates where this path exits the upper part again.
- **Rejection of Path-Closing Rows**:
  - A successor row must be rejected if it closes a path through the upper part of the board.
  - This prevention of path closure ensures the constructed solution remains cycle-free.
- **Induction of Bracket Structures**:
  - Any bracket structure in a sweep row that is consistent with the currently considered successor row will uniquely induce a bracket structure on the successor row.
  - This method preserves the integrity of path consistency and direction across rows.
- **Illustrative Example**:
  - Referencing Figure, the bracket structure and corresponding river basin are mapped from the predecessor to the successor row.
  - A single path originates at the only upward arrow, moves northwest, and re-enters via a preset downward arrow, which is tracked through to the sweep row.

Paper Approach - Complexity Result Now we count all possible configurations step by step:

- **Bracket Structure Assumption**: Assume that we consider a fixed bracket structure, and we also fixed for each bracket pair which of the two brackets (opening or closing) is associated to an upward arrow.
- **Matching Brackets with Arrows**: Assume we have $n^{[]}$ bracket pairs involved and that we have $n^{[]}_{\downarrow}$ downward arrows that could be matched with these brackets on the sweep row.

  As the bracket structure is fixed, there are no more than $n^{[]}_{\downarrow} + n^{[]} n^{[]}_{\downarrow}$.

- Assume that we have $n^{[]}_{\uparrow}$ upward arrows that are involved in potential cycles.
- Recall that if the Roma-cell is in the upper part of the board, then not all upward arrows need to be in a path that returns to the sweep row.
- As each such upward arrow gives rise to a bracket pair, we only have to decide if the opening or closing bracket of this bracket pair is associated with such an upward arrow.
- It is well-known that the number of ways we can properly form bracket expressions with p pairs of brackets is given by $\frac{1}{p+1} \times 2pp$, which is upper-bounded by 4^p (this is also known as Catalan numbers).
- Therefore, we have $8^{n^{[]}_{\uparrow}}$ many possibilities to create bracket structures and fix the association with upward arrows.

- We have to reason about the type of the letters from Δ and how to count them. Since the Roma-cell is unique and pre-set, it does not enter the following considerations.
- The type of a letter is completely determined by the given box structure. Therefore, when considering the set of configurations of the sweep row, each cell may host different letters, but all of them have the same type.
- There are only 4 different letters of type 0 and of type 3.
- There are 12 different combined letters of type 1, because the combined letters $(a, a) \in \Delta \times \Delta$ would never appear, as it makes no sense to put the symbol a into a cell and tell, at the same time, that the last symbol that could be set in that box is also a.
- Similarly, there are also 12 different combined letters of type 2.
- In the worst case, this gives a factor of 3 for each of the 4 basic letters, which corresponds to the factor 3^n in the figure.

Conclusion Theorem: There is an $O(2^{O(n)})$-algorithm for solving an $n \times n$ Roma puzzle.

$$3^n \sum_{t=0}^{n} \binom{n}{t} 4^{n-t} \left(\sum_{k=0}^{t} \binom{t}{k} 8^k \right) = 3^n \sum_{t=0}^{n} \binom{n}{t} 4^{n-t} 9^t = 3^n 13^n = 39^n.$$

Fig. 13: Count for the number of configurations and hence for the space consumption of DP algorithm

5 Discussion

Games, such as the Roma puzzle, can serve as an introduction to theoretical concepts in graphs. The game Generalized Roma is played on a weakly connected directed graph with a special vertex, the Roma-vertex, having an out-degree of zero. The goal is to delete edges (not present in a given set of indices) to obtain a weakly connected directed acyclic graph with a maximum out-degree of one. This game illustrates notions such as spanning trees and feedback arc sets.

The condition of the cells can be modeled by assigning colors to the edges and vertices, ensuring that two edges emanating from vertices of the same color do not have the same color. This allows for various board designs, such as triangular or hexagonal cells.

An algorithmic challenge is to improve the algorithms for solving $n \times n$ Roma puzzles in time $O^*(c^n)$ without using exponential space. Approaches utilizing the treewidth and treedepth of planar graphs can be explored, as demonstrated by recent work on single-exponential algorithms with treedepth [6]. Testing different algorithmic approaches, including linear programming (LP) solvers, on concrete Roma puzzles would also be relevant.

References

1. Goergen, K., Fernau, H., Oest, E., Wolf, P.: All Paths Lead to Rome. Universität Trier, Fachbereich 4 – Abteilung Informatikwissenschaften, 54286 Trier, Germany. s4kegoer,fernau,s4esmeck,wolfp@uni-trier.de
2. Yato, T., Seta, T.: Complexity and completeness of finding another solution and its application to puzzles. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 86-A(5), 1052–1060 (2003)
3. Lichtenstein, D.: Planar formulae and their uses. SIAM Journal on Computing 11(2), 329–343 (1982)
4. Impagliazzo, R., Paturi, R.: On the complexity of k-SAT. Journal of Computer and System Sciences 62(2), 367–375 (2001)
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: *Introduction to Algorithms*. MIT Press, 3rd edn. (2009)
6. van der Zanden, T.C.: Games, puzzles and treewidth. In: Fomin, F.V., Kratsch, S., van Leeuwen, E.J. (eds.) Treewidth, Kernels, and Algorithms - Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday. Lecture Notes in Computer Science, vol. 12160, pp. 247–261. Springer (2020)