Sudoku is NP-Complete: 3SAT Reduction with Gadget Visualization

Abstract

This project presents a complete proof that the generalized Sudoku decision problem is NP-complete. The proof consists of two parts:

1. showing that Sudoku is in NP,
2. demonstrating NP-hardness via polynomial-time reductions from:
    - 3SAT using logical gadgets (variable, literal, and clause gadgets)

Accompanying this report is a Manim animation script illustrating the construction.

1 Introduction

Sudoku is a constraint-satisfaction puzzle traditionally played on a 9×9 grid. In computational complexity, we study the generalized version, where the grid size $n^2 \times n^2$ is part of the input. The **Sudoku decision problem** asks whether a partially filled grid can be completed to a valid solution.

We prove the following theorem:

Theorem 1. *The generalized Sudoku problem is NP-complete.*

The proof proceeds by showing:

1. Sudoku $\in$ NP
2. Sudoku is NP-hard via the following reduction:
    - 3SAT $\leq_p$ Sudoku using polynomial-time gadget construction

2 Preliminaries

2.1 Definition of NP-Completeness

A problem is **NP-complete** if:

- It is in NP, meaning solutions can be verified in polynomial time.
- It is NP-hard, meaning every problem in NP reduces to it.

2.2 3SAT

In 3SAT, we are given a Boolean formula

$$C_1 \wedge C_2 \wedge \cdots \wedge C_m$$

where each clause contains exactly three literals. 3SAT is a canonical NP-complete problem.

2.3 Generalized Sudoku

Given an $n^2 \times n^2$ grid divided into $n \times n$ blocks, we ask whether the grid can be completed using symbols $\{1, 2, \ldots, n^2\}$ so that each symbol appears exactly once in each row, column, and block.

3 Sudoku $\in$ NP

Proof. Consider Generalized Sudoku:

> **Input:** An $n^2 \times n^2$ partially filled grid G with symbols from $\{1, 2, \ldots, n^2\}$.

To show that SUDOKU $\in$ **NP**, we provide a certificate and a polynomial-time verifier.

Certificate. A certificate for an input instance $\langle G \rangle$ is a completely filled $n^2 \times n^2$ grid C using symbols from $\{1, \ldots, n^2\}$.

Verification algorithm. Given $\langle G \rangle$ and certificate C, the verifier checks:

1. C contains exactly n^4 entries, each in $\{1, \ldots, n^2\}$.
2. For every pre-filled cell in G, the value in C matches it.

3. Each of the n^2 rows of C contains every symbol from $\{1, \ldots, n^2\}$ exactly once.

4. Each of the n^2 columns of C contains every symbol from $\{1, \ldots, n^2\}$ exactly once.

5. Each of the n^2 blocks of size $n \times n$ contains every symbol from $\{1, \ldots, n^2\}$ exactly once.

Running time. Each of these checks can be performed using simple counting in $O(n^4)$ time. Since the input size is $\Theta(n^4)$, the verification runs in polynomial time.

Conclusion. A valid completion serves as a polynomial-size certificate that can be verified in polynomial time. Therefore,

$$\textsc{Sudoku} \in \mathbf{NP}.$$

4 NP-Hardness of Sudoku via 3SAT Reduction (Gadget Method)

Overview

We prove that the decision version of Sudoku is NP-hard by giving a polynomial-time reduction from the NP-complete problem 3SAT. Given a 3-CNF Boolean formula $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ over variables $x_1, \ldots, x_n$, we construct a Sudoku instance $S(\varphi)$ that is solvable if and only if φ is satisfiable.

The reduction uses three types of gadgets: *variable gadgets*, *literal-propagation gadgets*, and *clause gadgets*. Each gadget occupies a constant-size region of the grid.

1. Variable Gadget

For each Boolean variable x_i, we construct a designated 3×3 subgrid V_i. This gadget satisfies the following formal properties:

1. **(Binary Choice)** There exist exactly two Sudoku-consistent assignments to the cells of V_i, denoted by:

$$\text{Pattern}_i^T \quad \text{and} \quad \text{Pattern}_i^F$$

corresponding to $x_i = \mathsf{TRUE}$ and $x_i = \mathsf{FALSE}$, respectively.

2. **(Mutual Exclusivity)** The Sudoku row/column constraints ensure that no other pattern is valid and that exactly one pattern must be chosen in any full solution.

3. **(Signal Output)** The gadget has three "output cells" (one for each literal occurrence of x_i in the formula) whose values are fixed by the choice of Pattern^T or Pattern^F. These outputs will be used by the literal propagation gadgets.

Thus each variable gadget implements a Boolean choice that must be globally consistent across the grid.

2. Literal Propagation Gadget

For each literal occurrence $\ell \in \{x_i, \neg x_i\}$ in a clause, we construct a propagation gadget $P(\ell)$. This gadget connects the output cells of V_i to the corresponding clause gadget.

The propagation gadget satisfies the following formal rules:

1. **(Consistency)** Let the variable gadget V_i output a digit d_i^T under TRUE and a digit d_i^F under FALSE. Then the propagation gadget forces a corresponding digit $d(\ell)$ to appear in its output cell:

$$d(\ell) = \begin{cases} d_i^T & \text{if } \ell = x_i, \\ d_i^F & \text{if } \ell = \neg x_i. \end{cases}$$

2. **(Uniqueness)** No other digit can appear in the output cell without violating a Sudoku rule.

3. **(Non-interference)** Propagation gadgets for different literals occupy disjoint regions and do not constrain each other except through global Sudoku rules.

Thus literal truth values are copied reliably from variables to clauses.

3. Clause Gadget

For each clause $C_j = (\ell_1 \vee \ell_2 \vee \ell_3)$ we construct a clause gadget G_j with three input cells, one for each literal.

The clause gadget enforces:

1. **(Satisfaction Condition)** The local Sudoku constraints admit a completion of G_j if and only if at least one of the input literals carries the *TRUE* digit.

2. **(Rejection Condition)** If all three input digits correspond to FALSE literals, then the gadget has no valid Sudoku completion. Thus:

$$G_j \text{ is fillable} \iff d(\ell_1) = T \text{ or } d(\ell_2) = T \text{ or } d(\ell_3) = T.$$

3. **(Polynomial Size)** Each clause gadget is of constant area (e.g., a single 3×3 block or a bounded combination of rows and columns).

This ensures that the Sudoku instance is solvable exactly when all clauses are satisfied.

4. Correctness of the Reduction

($\Rightarrow$) If φ is satisfiable, then $S(\varphi)$ has a solution.

Let a be a satisfying Boolean assignment. For each variable x_i:

$$x_i = \mathsf{TRUE} \implies \text{place Pattern}_i^T \qquad x_i = \mathsf{FALSE} \implies \text{place Pattern}_i^F.$$

The propagation gadgets copy these truth values to the clause gadgets. Since every clause has at least one true literal, each clause gadget has a valid completion. Thus the entire Sudoku grid can be completed without conflict.

($\Leftarrow$) If $S(\varphi)$ has a solution, then φ is satisfiable.

Every variable gadget must choose one of its two possible patterns. Interpret this choice as a Boolean assignment.

Since each clause gadget is fillable only when at least one input literal is TRUE, all clauses are satisfied under this assignment. Hence φ is satisfiable.

5. Polynomial-Time Construction

The reduction runs in polynomial time because:

1. Each gadget type has constant size.

2. We create one variable gadget per variable and one clause gadget per clause.

3. Literal propagation gadgets introduce only linear overhead.

Thus the total size of the Sudoku instance is $O(n+m)$, and the transformation

$$3\text{SAT} \leq_p \textsc{Sudoku}$$

is a valid polynomial-time reduction.

Example: How the Reduction Maps Variables to Clauses

To illustrate the gadget-based reduction concretely, we present an explicit example showing how truth values chosen inside variable gadgets propagate to a clause gadget.

Example Formula

Consider the clause

$$C_j = (x_1 \vee \neg x_3 \vee x_5).$$

In the constructed Sudoku instance, each of the literals x_1, $\neg x_3$, and x_5 receives its value through the corresponding variable gadget and propagation gadget.

Step 1: Variable Gadget Assignments

Assume the variable gadgets choose the following truth values:

$$x_1 = \text{TRUE}, \qquad x_3 = \text{FALSE}, \qquad x_5 = \text{TRUE}.$$

Each variable gadget has exactly two possible Sudoku-consistent fillings:

$$\text{Pattern}_i^T \quad \text{and} \quad \text{Pattern}_i^F.$$

These enforce different digits in their output cells.

For this example, suppose the variable gadgets are designed to output:

$$d(x_1) = 3, \qquad d(\neg x_3) = 9, \qquad d(x_5) = 2.$$

Step 2: Literal Propagation

Each literal occurrence has a propagation gadget $P(\ell)$. These gadgets are placed so that a specific digit must appear in an output cell of the clause gadget, otherwise a Sudoku conflict occurs.

Thus the digits transmitted to clause gadget G_j are:

$$(3,\ 9,\ 2),$$

corresponding to $(x_1,\ \neg x_3,\ x_5)$ respectively.

Step 3: Clause Gadget Behavior

The clause gadget G_j has three input cells. It is constructed so that:

- the gadget is solvable *if and only if* at least one input digit corresponds to a TRUE literal, and
- the gadget is unsolvable if all three correspond to FALSE.

In this example:

$$x_1 = \text{TRUE}, \qquad x_5 = \text{TRUE}.$$

Thus at least one of the digits $\{3, 2\}$ is a TRUE-digit. Therefore the clause gadget admits a valid Sudoku completion:

$$G_j \text{ is fillable.}$$

Conclusion of Example

The example shows how:

1. variable gadgets encode truth values as distinct Sudoku digits,
2. literal propagation gadgets copy those digits along forced rows/columns, and
3. the clause gadget admits a completion if and only if one of the transmitted digits corresponds to a TRUE literal.

Thus the Sudoku instance is solvable exactly when the Boolean formula is satisfiable.

Conclusion

Since Sudoku is in NP and 3SAT reduces to Sudoku, the Sudoku decision problem is NP-complete.

5 Conclusion

We have shown:

- Sudoku $\in$ NP,
- Sudoku is NP-hard via reductions from 3SAT.

Hence, generalized Sudoku is NP-complete.