Invariance of Initial Conditions in P and NP and Structural Incompatibility Between Problems with Hypothetical "Additional Hidden Properties" Allowing Partial Solutions to Each

Author: Dobri Bozhilov

April 26th 2025

1. Introduction

The question of the relationship between the classes P and NP is one of the central problems in modern complexity theory. The usual line of attack on the problem seeks either to find a polynomial-time algorithm for solving an NP-complete problem or to prove its nonexistence.

In the present work, we will consider an alternative approach: we will explore the possibility of resolving the P versus NP problem by establishing a contradiction between the necessary conditions for efficiently solving different NP-complete problems. That is, if a solution is found for one problem through some new hidden property, this property would be in conflict with an analogous property needed to solve another problem. The focus will be on the role of hidden properties in the input of the problems and their impact on the possibility of universal mutual reduction between them.

2. Thought Experiment: Contradiction Between Hidden Properties of NP-Complete Problems

2.1 Main Setup

We consider two classical NP-complete problems:

- Subset Sum Problem (SSP)
- Knapsack Problem (KP)

As part of the preliminary study, a third problem — the Travelling Salesman Problem (TSP) — was also tested, and it showed a similar result.

We hypothetically assume that for each of these problems, a special hidden property is discovered that allows fast (polynomial-time) solving:

- SSP: property X
- KP: property Y:

2.2 Contradiction Between Subset Sum and Knapsack Problem

We assume that for Subset Sum there exists a property X that allows a fast selection of the correct numbers, and for Knapsack, a property Y that marks the optimal items.

We analyze:

- Transformation SSP → KP: Possible loss of the hidden property.
- Transformation KP → SSP: Analogous violation of transferability.

Consequence: The hidden properties disrupt the possibility of polynomial-time mutual reduction between the problems.

More specifically and concretely, the contradiction can be described as follows:

We have two NP-complete problems, each with a different nature:

Subset Sum Problem (SSP)

Here, the main difficulty arises from the need to discover some unknown property of the numbers themselves, which would allow a direct (non-exhaustive) selection.

Knapsack Problem (KP)

In KP, the difficulty is not entirely numerical in nature. It is more combinatorial and related to optimization: the items we are trying to fit into the knapsack are defined not only by their numerical values but also by combinations of weights and values. That is, the constraints there are more structural and combinatorial.

3. Let Us Formalize the Process "Step by Step":

Step 1: Subset Sum Problem (SSP)

Definition:
The Subset Sum Problem is formally defined as follows:

Given a set of positive integers:

$S=\{a_1, a_2, a_3,\ldots, a_n\}$, $a_i \in Z$

Also given is a positive integer T, called the "target":

$T \in Z$

Question:
Does there exist a subset

$S' \subseteq S$, such that:

$$\sum_{ai \in S'} \text{ai} = \text{T}$$

Difficulty of the problem:

If this problem is solved by an exhaustive approach, the solution time is exponential $O(2^n.\text{n})$

In order to solve it efficiently (in polynomial time), there must exist some unknown numerical property X of the numbers, which allows preliminary pruning of the possible combinations.

The hypothetical numerical property X would enable the elimination of the exponential number of checks, allowing the problem to be solved efficiently.

Step 2: Knapsack Problem (KP)

Definition:
The Knapsack Problem is formally defined as follows:

Given a set of items:

I={1,2,3,…,n}

Each item has two characteristics:

Value:

$v_i \in Z^+$

Weight:

$w_i \in Z^+$

Given a knapsack capacity:

$W \in Z^+$

Question:
Does there exist a subset of items

$I' \subseteq I$, such that the total weight does not exceed

W, and the total value is maximized?

Formally:

Maximize

$$\sum_{i \in I'} vi$$

subject to:

$$\sum_{i \in I'} wi \leq W$$

Difficulty of the problem:

This problem is combinatorial, with the constraint arising from the combination of weights and values.

In order to solve it efficiently (in polynomial time), there must exist some structural-combinatorial property Y that allows preliminary elimination of combinations that are guaranteed not to lead to an optimal solution.

Here, the property Y is expected to be of a completely different nature — not purely numerical, but structural-combinatorial.

Step 3: The Hypothetical Numerical Property X

Let us define more precisely the numerical property necessary for a polynomial-time solution of the Subset Sum problem:

Property X means that for an arbitrary set of numbers S and an arbitrary target number T, we can define a function

$f_X(S,T)$,

which can be computed in polynomial time and eliminates a large portion of the subsets, leaving only a polynomially small number to check.

Formally:

There exists a polynomial-time function

$f_X(S,T)$, such that:

$f_X(S, T) \Rightarrow S`` \subseteq S, |S``| \leq P(n)$, where P(n) is a polynomial.

and such that if a solution exists, it must be contained within $S``$.

Step 4: The Combinatorial Property Y

Let us also formalize the structural-combinatorial property Y necessary for a polynomial-time solution of the Knapsack problem:

Property Y would represent a structural constraint (on the combinations), which is also computable in polynomial time and eliminates inappropriate combinations.

Formally:
There exists a polynomial-time function

$f_Y(I,W)$, which reduces the solution space to polynomial complexity:

$f_Y(I,W) \Rightarrow I`` \subseteq I$ , $|I``| \leq Q(n)$, where $Q(n$ is a polynomial, and such that if a solution exists, it must be contained within $I``$.

Step 5: Central Hypothesis of the Contradiction

The idea is:

If we assume that X (for SSP) exists, it must be applicable to any set of numbers and must be based on a fundamental numerical property.

But if property Y (for KP) fundamentally excludes the existence of numerical property X, then the two assumptions will mutually contradict each other:

$Y \Rightarrow \neg X$

Thus, it would be impossible for a universal approach to exist that satisfies both the Subset Sum and the Knapsack problems simultaneously.

Therefore, a polynomial-time solution for the entire NP class cannot exist.

Step 6: Analysis of the Fundamental Nature of Property X (Subset Sum)

For X to exist, the following must be true:

- In the numbers of any set S, there must always exist a numerical regularity that significantly reduces the number of possible sum combinations.

This regularity could be of the following types:

- **Algebraic** (e.g., modular arithmetic) – numbers forming a sum T have certain remainders under some modulus.
- **Geometric or vector-based** – numbers forming geometric or positional structures leading to restrictions.
- **Purely combinatorial symmetry** – specific regularities in the combination of numbers.

Conclusion about X:

Property X requires a **global regularity**, based solely on the numbers themselves (without any characteristics beyond numerical value).

Step 7: Analysis of the Nature of Property Y (Knapsack Problem)

- Property Y would be structural-combinatorial:
  - It is based on combinations of two parameters (value and weight), which do not depend solely on the numerical values, but on the interaction between them.
  - In Knapsack, the reduction of the solution space is based on whether a combination of weights allows reaching the optimum for the value.
- For Y to be universally applicable, it must:
  - Be able to effectively (in polynomial time) eliminate item combinations based on the dual criterion (weight and value).
  - Not require any universal numerical regularity among the values or weights, because such a regularity would heavily constrain the possible inputs and make the problem trivially solvable.

Conclusion about Y:

Property Y is **local-combinatorial**. It cannot rely on universal numerical regularities in the numbers.

Step 8: Key Logical Contradiction

- If the universal numerical property X exists, then all numbers possess a regularity that allows efficient selection.
- Then the numbers used as weights and values in the Knapsack problem must also possess such a regularity.
- However, in the Knapsack problem, the numbers (weights and values) are independent parameters, with no mandatory correlation between them. Nothing prevents complete randomness between them.
- Therefore, if it turns out that for a polynomial-time solution to Knapsack (property Y) the numbers must **not** contain hidden regularities (i.e., property X), we will have a direct contradiction.

Brief Formulation of the Contradiction:

- If X exists, all numbers have regularity.
- If Y exists, the numbers used in KP cannot have such universal regularity.
- Therefore, the two properties cannot simultaneously exist.

Step 9: Formalization of the Contradiction

Formally, it becomes:

- Suppose that P=NP. Then both SSP and KP have polynomial-time solutions.
- If SSP has a polynomial-time solution ⇒ a numerical property X exists for all numbers.

- If KP has a polynomial-time solution ⇒ a combinatorial property Y exists for all combinations of numbers (weights and values).
- But the existence of Y ⇒ numbers must be free from universal regularities, because KP necessarily allows arbitrary independent input parameters.
- This leads to a direct contradiction:
    - X requires regularity in all numbers.
    - Y requires the absence of universal regularity.

Conclusion:
The contradiction means that our assumption P=NP is false.

Therefore, P≠NP.

4. Input and the Limitations of the Turing Machine

A fundamental element for all NP-complete problems is the requirement that they must be solvable on a "Turing Machine." Currently, this is possible (in non-polynomial time) because the problems have a relatively low level of initial complexity. They are simple in concept but heavy in the number of operations.

Introducing additional "properties" into any problem or solution would change its initial complexity. And this new complexity could not be embedded into the machine. That is, the problems and the machine are interconnected, and the machine can solve exactly the specific problems.

Changing the problem means changing the machine.

Simply put — we cannot introduce a "magical additional property" into the machine, even if it exists. The machine itself is the bottleneck, and everything starts from it.

The Turing Machine operates on pure binary strings without additional external signs such as colors or markers.

If hidden properties are not explicitly encoded, they are inaccessible to the machine.

Introducing hidden properties without encoding changes the very nature of the problem and removes it from the standard class of NP-complete problems.

The alternative would be to introduce different programs into the machine for different problems, because different problems would rely on different "hidden properties".

But this would change the character of the machine and the compatibility of the problems (NP-complete), which must be solvable on the same machine with the same program.

The problems are compatible with each other precisely because they can all be reduced to a "simple input" for the Turing Machine. If this becomes impossible or if the input changes, we step outside the scope of the problem.

More precisely formulated:

"NP-complete problems are compatible with each other because they can all be reduced to equally structured binary inputs, which are processed by a universal deterministic computational model. If the input can no longer be represented in this way — for example, if it contains additional non-encodable properties — then the problems fall outside the classical definition of NP."

Formal Presentation

1. The Turing Machine Model

- Inputs are represented as strings over a fixed alphabet (usually {0,1}).
- Numbers are encoded as bit strings.
- The machine has no access to "hidden" properties — only to what is explicitly encoded in the string.

2. What Does This Mean for Problems Like Subset Sum?

- When we receive numbers for Subset Sum, they are simply bit representations (e.g., 1101 for 13).
- The machine knows nothing about "symmetry", "hidden dependencies", "geometric relations", or any other potential hidden properties of these numbers.
- It can only read bits and manipulate them according to elementary rules (read, write, move).

3. Therefore:

- If somewhere in the "real" world of mathematics there exists a hidden property of the numbers, **it will not be visible to the Turing Machine** unless it is explicitly encoded in the bits.
- Therefore, the hidden property is inaccessible to the algorithm.
- Therefore, the algorithm must work as if the hidden property does not exist.
- This preserves the high complexity of the problem.

In short:

P≠NP not because there is no "shortcut", but because the very world of the Turing Machine is rigid, clear, and inevitably limited, and it will not allow the use of such a "shortcut" if the goal is to preserve the machine's parameters.

5. Nature of the Difficulty in NP-Complete Problems

NP-complete problems are hard to solve because of the exponential number of possible solutions, but they are structurally simple.

Any attempt to ease the solution by using hidden properties violates the definition of the problem.

6. Final Conclusion

Stability Formula:

NP-complete problems are combinatorially hard but structurally simple.

Hidden properties violate the invariance of the input and destroy the possibility of universal mutual reduction between problems.

Thus:

- Either such hidden properties do not exist,
- Or, if they do exist, the class NP breaks into incompatible subproblems,
- Which fundamentally supports the claim P≠NP.

7. Other Considerations About P vs. NP and SSP

Where does the possibility come from for the problem to be solved in reduced exponential time ($2^{n/2}.n$)? (Sahni–Horowitz algorithm)

It comes from the one-dimensional nature of the number line, which has only two directions. This gives the numbers a property that allows reducing the complexity — but only by half. If hypothetically the number line was multidimensional, the complexity could be further reduced. However, it is not — in the initial conditions of the problem, which is defined in the domain of integers.

Why does dynamic programming allow a polynomial-time solution (for specific variants)?

Because there the complexity is shifted from the solution itself into the definition of the numbers. If the numbers are represented not by digits (which contain a lot of information and complexity), but by sticks placed side by side, then the solution becomes simple — but the representation of the numbers becomes complicated and grows exponentially.

8.Acknowledgments

The author expresses gratitude to GPT-4 (OpenAI ChatGPT), whose interactive assistance helped resolve many of the technical questions and computations.

9.References:

1. **S. A. Cook**, *The Complexity of Theorem-Proving Procedures*, Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC), 1971.
2. **E. Horowitz and S. Sahni**, *Fundamentals of Computer Algorithms*, Computer Science Press, 1978.
3. **M. Sipser**, *Introduction to the Theory of Computation*, Cengage Learning, 3rd ed., 2012.
4. **C. Papadimitriou**, *Computational Complexity*, Addison-Wesley, 1994.

==========

Contact:

E-mail: zadrugata2019 (at) gmail (dot) com

Phone: ++359 887 467 818