Why $P = NP$ With a Low-Degree Algorithm Could Transform Drug Discovery for Diseases We Cannot Cure Today

Alwin
Universitas Indonesia

May 2026

Abstract. The question P versus NP appears, at first, to belong entirely to theoretical computer science. Yet drug discovery is filled with tasks that look like enormous search problems under many constraints: find a molecule, protein, antibody, gene-editing strategy, drug combination, or patient-specific therapy that satisfies many biological, chemical, and safety requirements at once. This article explains why a proof that $P = NP$ would matter for medicine only if it produced a practical low-degree algorithm, and why such an algorithm could, in principle, accelerate the discovery of treatments for diseases that remain incurable today. The central idea is simple: if a candidate treatment can be checked efficiently by a good computational verifier, then a practical $P = NP$ algorithm could help find such a candidate efficiently.

Core thesis

A practical low-degree algorithm for $P = NP$ would not magically cure diseases. Its possible revolutionary impact would be more specific: it could turn certain impossible-looking biological and chemical searches into efficiently solvable searches, provided the disease model and verifier are good enough.

1. A Mathematical Question That Seems Far From Medicine

Every so often, mathematics produces a question that seems to belong entirely to another planet.

The question P versus NP is one of them.

It talks about algorithms, verification, polynomial time, Boolean formulas, Turing machines, and other objects that do not immediately look like hospitals, cells, proteins, patients, or medicine.

And yet, if a certain version of $P = NP$ were ever discovered, it could become one of the most consequential scientific breakthroughs in human history.

Not because it would magically cure cancer next Tuesday.

Not because a theorem in computer science would suddenly replace biology, chemistry, clinical trials, doctors, or laboratories.

But because it could change something more basic:

our ability to search.

And drug discovery is, to a surprising extent, a search problem.

We are often searching for a molecule, a protein, an antibody, a gene-editing strategy, a combination of drugs, or a personalized treatment plan that satisfies a long list of constraints. The difficulty is not merely that we do not know what we want. Often, we know roughly what we want. The difficulty is that the space of possible answers is unimaginably large.

This is where P versus NP enters the story.

2. The Small Puzzle Behind the Big Problem

Let us start with a simple distinction.

Suppose I give you a completed Sudoku puzzle. Checking whether it is valid is relatively easy. You look at the rows, columns, and boxes, and verify that the rules are satisfied.

But if I give you an empty Sudoku puzzle and ask you to solve it, that is a different matter.

Checking is easier than finding.

This is the intuition behind NP problems. In rough terms, NP contains problems where a proposed solution can be checked efficiently. P contains problems where a solution can be found efficiently. The Clay Mathematics Institute describes the essence of P versus NP as the question of whether every problem whose solution can be checked quickly can also be solved quickly.[1]

So the question is:

If a solution is easy to verify, is it also easy to find?

Most computer scientists believe the answer is no.

But suppose the answer were yes.

Suppose $P = NP$.

And suppose, even more importantly, that the proof gave us a practical algorithm.

That last sentence matters more than it may first appear.

3. The Low-Degree Condition

There is a quiet trap in the phrase "polynomial time."

An algorithm that runs in time

$$n^2$$

or

$$n^3$$

[1] Clay Mathematics Institute, "P vs NP," https://www.claymath.org/millennium/p-vs-np/.

may be very useful.

An algorithm that runs in time

$$n^{1000}$$

is also polynomial time.

But it is useless in practice.

This is why the real-world version of the dream is not merely:

$P = NP$.

The useful version is:

$P = NP$ with a low-degree polynomial-time algorithm.

Something like

$$O(n^2), \quad O(n^3), \quad O(n^4), \quad \text{or perhaps } O(n^5)$$

would be extraordinary.

Something like

$$O(n^{1000})$$

would mostly be an interesting monument in the museum of theoretical computer science.

So from now on, when this essay says "a practical $P = NP$ algorithm," it means something stronger than a purely theoretical proof. It means an algorithm that can actually run on real scientific instances, with a polynomial exponent small enough to matter.

This is the difference between a theorem that changes our understanding of computation and an algorithm that changes laboratories.

Why the exponent matters

The phrase "polynomial time" is not enough. For medicine, a useful algorithm must have an exponent small enough to run on real biological and chemical inputs. A theorem with runtime n^{1000} may be theoretically historic but practically irrelevant.

4. Drug Discovery Is Not One Problem

It is tempting to say:

Drug discovery is hard because biology is complicated.

This is true, but it hides too much.

Drug discovery is hard because it combines many hard problems at once.

A useful drug candidate should bind to the right target. It should avoid the wrong targets. It should not be too toxic. It should survive long enough in the body. It should reach the relevant tissue. It should be manufacturable. It should not trigger dangerous immune reactions. It should work at a reasonable dose. It should remain useful even when a virus, tumor, or bacterial population evolves.

And after all of that, it still has to pass preclinical and clinical testing. The FDA describes drug development as a sequence beginning with discovery and development, followed by preclinical research, clinical research, review, and post-market safety monitoring.[2]

A theorem does not remove these steps.

But it might transform the earliest and most mysterious one:

finding promising candidates in the first place.

5. A First Mathematical Model

Let us try to compress the early drug discovery problem into a mathematical toy model.

Let

$$X$$

be the set of possible interventions.

An element

$$x \in X$$

could be a small molecule, an antibody, a designed protein, a genetic intervention, a drug combination, or a treatment schedule.

Now let

$$D$$

represent the disease.

This might include a disease-causing protein, a mutation, a viral mechanism, a cancer pathway, an inflammatory process, or a failure in a cellular system.

The problem is to find an x that helps with D.

But "helps" is not one condition. It is many conditions.

So we define constraints:

$$C_1(x), C_2(x), C_3(x), \ldots, C_k(x),$$

where each C_i is a requirement.

For example:

$$C_1(x) = \text{“}x \text{ binds to the disease target”},$$
$$C_2(x) = \text{“}x \text{ is not highly toxic”},$$
$$C_3(x) = \text{“}x \text{ reaches the correct tissue”},$$
$$C_4(x) = \text{“}x \text{ remains stable long enough”},$$
$$C_5(x) = \text{“}x \text{ can be manufactured”},$$
$$C_6(x) = \text{“}x \text{ avoids dangerous off-target effects”}.$$

Then a candidate drug is an x such that

$$C_1(x) \wedge C_2(x) \wedge C_3(x) \wedge \cdots \wedge C_k(x) = \text{true}.$$

In plain English:

Find an intervention that satisfies all the important constraints.

This is already beginning to look like an NP-style problem.

Not because real biology is literally Sudoku.

But because the structure is familiar:

many constraints, enormous search space, easier verification than discovery.

[2] U.S. Food and Drug Administration, "The Drug Development Process," https://www.fda.gov/patients/learn-about-drug-and-device-approvals/drug-development-process.

6. The Verifier

In complexity theory, an NP problem has a verifier.

A verifier checks whether a proposed solution works.

Let us define a computational verifier:

$$V(D, x),$$

where D is the disease model and x is the proposed intervention.

The verifier outputs

$$V(D, x) = \begin{cases} 1, & \text{candidate passes model checks,} \\ 0, & \text{otherwise.} \end{cases}$$

For drug discovery, $V(D, x)$ might check predicted binding strength, predicted toxicity, stability, selectivity, manufacturability, delivery, metabolic behavior, and resistance risk.

Of course, this verifier is not the same thing as reality.

A computational model can be wrong. A molecule can look excellent in simulation and fail in living cells. A treatment can work in animals and fail in humans.

But even with that caveat, the verifier captures an important computational structure:

> *Given a proposed candidate, we can often test or simulate many relevant properties more easily than we can find the candidate from scratch.*

This is the NP-like flavor of drug discovery.

The hard part is not merely checking a molecule.

The hard part is finding a molecule worth checking.

7. The Search Problem

Now we can write the core search problem:

$$\text{Given } D, \text{ find } x \in X \text{ such that } V(D, x) = 1.$$

That sentence is short.

It is also where the difficulty lives.

The set X may be astronomically large. If X is the space of possible drug-like molecules, then brute force is hopeless. If X is the space of possible drug combinations and dosages, it may be even worse.

A naive algorithm might try

$$x_1, x_2, x_3, \ldots, x_N$$

one by one.

If N is enormous, this is not science. It is waiting for the heat death of the universe with a clipboard.

A practical $P = NP$ algorithm would change the situation.

If the problem can be encoded as an NP problem, and if $P = NP$ with a low-degree algorithm, then there could exist an algorithm A such that

$$A(D) = x,$$

where

$$V(D, x) = 1,$$

and the running time is

$$O(n^c)$$

for some small constant c.

This is the key mechanism.

The algorithm would not merely test molecules faster.

It would find candidates by solving the constraint problem.

That is a very different kind of power.

8. From "Does There Exist?" to "Find One"

P versus NP is usually stated as a decision problem.

A decision problem asks for a yes-or-no answer.

For example:

> *Does there exist a molecule that binds to target T strongly enough and is predicted to be safe?*

Mathematically:

$$\exists x \in X \quad \text{such that} \quad V(D, x) = 1.$$

This only asks whether a solution exists.

But in medicine, we need the actual candidate.

We do not merely want the answer:

> *Yes, somewhere in chemical space there exists a useful molecule.*

That would be comforting but not helpful.

We want the molecule.

Fortunately, for many NP problems, if one can efficiently decide whether a solution exists, one can often also construct a solution by repeatedly asking decision questions.

Very roughly, the algorithm can ask:

> *Is there a solution whose first bit is 0?*

If yes, keep that bit. If no, the first bit must be 1.

Then ask the same question for the second bit, and so on.

In this way, a decision algorithm can often be converted into a search algorithm.

So the dream is not merely

$$\exists x,$$

but

$$\text{find } x.$$

This is the transition that would matter for drug discovery.

It would turn the statement:

> *There may exist a treatment candidate satisfying these constraints.*

into:

> *Here is a candidate worth testing.*

9. Optimization: Because Valid Is Not Enough

Real drug discovery is not just about finding any molecule that barely passes the test.

Scientists want better candidates.

So instead of a pure yes-or-no verifier, we might define a score function:

$$S(D, x),$$

where $S(D, x)$ measures how promising intervention x is for disease D.

A simplified score might look like

$$S(D, x) = w_1 B(x) - w_2 Tox(x) + w_3 Stab(x) + w_4 Del(x) - w_5 Risk(x),$$

where:

- $B(x)$ is binding quality,
- $Tox(x)$ is toxicity,
- $Stab(x)$ is stability,
- $Del(x)$ is delivery efficiency,
- $Risk(x)$ is side-effect or resistance risk,
- $w_1, \ldots, w_5$ are weights.

Then the problem becomes

$$\max_{x \in X} S(D, x)$$

subject to safety and biological constraints.

This is an optimization problem.

P versus NP is usually about decision problems, but many optimization problems can be converted into decision problems.

Instead of asking:

> *What is the best score?*

ask:

$$\exists x \in X \quad \text{s.t.} \quad S(D, x) \geq K?$$

If we can answer this efficiently, we can search over possible values of K. This turns the decision algorithm into an optimization engine.

In simple terms:

> *A practical $P = NP$ algorithm could help us not only find a candidate, but search for a better candidate.*

10. A Concrete Disease-Target Example

Suppose a disease is driven by a harmful protein P.

Scientists want a molecule x that blocks P, but does not damage healthy systems.

We might write:

$$Bind(x, P) \geq b,$$

$$Toxicity(x) \leq t,$$

$$Stability(x) \geq s,$$

$$Delivery(x) \geq d,$$

$$OffTarget(x) \leq r.$$

In words:

- binding must be strong enough,
- toxicity must be low enough,
- stability must be high enough,
- delivery must be good enough,
- off-target activity must be limited.

Then the problem is:

$$\exists x \in X :$$

$$Bind(x, P) \geq b \wedge Toxicity(x) \leq t \wedge Stability(x) \geq s \wedge Delivery(x) \geq d \wedge OffTarget(x) \leq r.$$

This is the whole difficulty in miniature.

Each condition may be understandable by itself.

But satisfying all of them together is hard.

A molecule that binds strongly may be toxic. A molecule that is safe may not reach the correct tissue. A molecule that reaches the tissue may be metabolized too quickly. A molecule that looks promising may be impossible to manufacture at scale.

Drug discovery is not about satisfying one beautiful equation.

It is about surviving a gauntlet of constraints.

11. Why Undruggable Might Mean Unsearched

Some disease targets are called "undruggable."

This does not always mean that no possible intervention exists. Sometimes it means that no intervention has been found with current methods.

A protein may have no obvious binding pocket. It may change shape. It may interact with many other proteins. It may be too similar to proteins that the body needs. Or the target may be located in a part of the cell that is difficult for drugs to reach.

In mathematical language, this might mean that the solution set is tiny.

Let

$$Good(D) = \{x \in X : V(D, x) = 1\}.$$

For an easy target, $Good(D)$ may be relatively large.

For a hard target, $Good(D)$ may be extremely small.

If

$$|Good(D)| \ll |X|,$$

then random search is nearly hopeless.

But a search algorithm does not need the good set to be large. It needs structure. If the constraints have exploitable structure and can be encoded in a useful way, then a practical $P = NP$ algorithm could find an element of $Good(D)$, even if that set is very small.

This is why the idea is so powerful.

The algorithm would not make the target biologically easy.

It would make the search less blind.

12. Proteins Are Not Locks, Drugs Are Not Keys

One popular explanation says that a drug fits into a protein like a key fits into a lock.

This analogy is useful for five minutes.

Then it becomes dangerous.

Proteins move. They fold. They breathe. They interact with water, membranes, ions, sugars, nucleic acids, and other proteins. Drug molecules also move. Their shapes are not fixed. Binding is not just geometry; it involves energy, entropy, charge, flexibility, and the surrounding biological environment.

So the real question is not simply:

> *Does this shape fit that shape?*

It is closer to:

> *Is there a molecular configuration that produces the desired biological effect while avoiding unacceptable side effects?*

That is a search and optimization problem over a vast physical landscape.

We might describe it as minimizing an energy-like function:

$$E(x, P, c),$$

where:

- x is the molecule,
- P is the protein,
- c is the biological context.

We want

$$E(x, P, c) \leq \theta,$$

but we also want safety, selectivity, stability, and delivery.

So even the "simple" problem of binding is not simple.

It becomes part of a larger constraint system.

13. Drug Combinations: When the Search Explodes

Now consider diseases where one drug is not enough.

Cancer, HIV, tuberculosis, autoimmune diseases, and many neurological conditions may require combinations of interventions.

Suppose we have n possible drugs:

$$D_1, D_2, \ldots, D_n.$$

A combination can be represented by a vector:

$$x = (x_1, x_2, \ldots, x_n),$$

where

$$x_i = \begin{cases} 1, & \text{if drug } D_i \text{ is used,} \\ 0, & \text{otherwise.} \end{cases}$$

Even before dosages, there are

$$2^n$$

possible subsets.

Now add dosages:

$$q = (q_1, q_2, \ldots, q_n).$$

Now add timing.

Now add patient-specific variables.

The search space grows violently.

A simplified objective might be

$$\max_{x,q} \left[Effect(D, x, q) - Toxicity(D, x, q) - InteractionRisk(x, q)\right]$$

subject to

$$Toxicity(D, x, q) \leq T,$$
$$Effect(D, x, q) \geq E,$$
$$q_i \leq q_i^{max}.$$

This is not merely a large spreadsheet.

It is a combinatorial optimization problem.

A practical low-degree $P = NP$ algorithm could, in principle, search through such spaces far more efficiently than brute force.

This could matter especially for diseases where the cure is not a single molecule, but a precise combination.

14. Personalized Medicine as a Search Problem

Now add the patient.

Let

$$p$$

represent a patient profile.

This profile may include genetic data, tumor mutations, immune markers, metabolism, age, previous treatments, organ function, and disease stage.

The verifier becomes

$$V(D, p, x).$$

This asks:

> *Given disease D, patient profile p, and treatment x, does x work safely for this patient?*

The personalized medicine problem becomes

$$\text{Find } x \text{ such that } V(D, p, x) = 1.$$

Or, in optimization form:

$$\max_x S(D, p, x).$$

A general drug asks:

> *What works for many patients?*

Personalized medicine asks:

What works for this patient?

That extra specificity is medically valuable.

It is also computationally expensive.

A practical $P = NP$ algorithm could make patient-specific treatment search more realistic. It could help convert biological data into tailored treatment candidates.

Again, this does not mean the algorithm cures the patient by itself.

It means it may find better hypotheses for doctors and scientists to test.

15. The Full Mechanism

We can now state the mechanism in one chain:

$$\begin{aligned}&\text{Disease biology} \to \text{model} \to \text{constraints}\\ &\to \text{verifier} \to \text{search algorithm}\\ &\to \text{candidate} \to \text{lab testing}\\ &\to \text{clinical trials} \to \text{therapy.}\end{aligned}$$

The $P = NP$ algorithm affects the middle of the chain.

It does not replace the beginning.

It does not replace the end.

It does not invent correct biology from nothing.

It does not guarantee safety.

It does not skip clinical trials.

But it could transform this step:

$$\text{find a candidate treatment.}$$

That is a very big step.

The mechanism in one sentence

A practical low-degree $P = NP$ algorithm could turn "there exists a candidate satisfying these constraints" into "here is a candidate worth testing."

16. A Proposition, Informally Stated

Here is a more mathematical way to summarize the dream.

Informal Proposition. Suppose that for a disease D, there exists a computationally checkable verifier $V(D, x)$ such that $V(D, x) = 1$ implies that x is a promising treatment candidate. Suppose also that the search problem

$$\exists x \in X : V(D, x) = 1$$

can be efficiently encoded as an NP problem. If $P = NP$ with a practical low-degree algorithm, then one could efficiently find such an x, whenever it exists.

This proposition is not a medical theorem.

It is a computational skeleton.

It says:

If the biology can be modeled well enough, and if candidate quality can be checked well enough, then a practical $P = NP$ algorithm could turn existence into discovery.

That phrase is the heart of the essay:

turn existence into discovery.

Today, for many diseases, it may be that a useful intervention exists somewhere in the space of possibilities.

But we do not know how to find it.

A practical $P = NP$ algorithm would attack exactly that gap.

17. The Necessary Cold Water

At this point, it is easy to become too excited.

So let us pour some cold water on the argument.

First, the verifier must be good.

If $V(D, x)$ is wrong, then the algorithm may find a candidate that satisfies the wrong conditions. A bad verifier plus a powerful search algorithm is not a cure. It is a very fast way to become confidently wrong.

Second, the disease model must be good.

If scientists misunderstand the disease mechanism, the algorithm may optimize the wrong target.

Third, simulations are not patients.

A molecule that works in a computational model may fail in cells. A molecule that works in cells may fail in animals. A molecule that works in animals may fail in humans.

Fourth, safety is not optional.

The goal is not to find a molecule that does something dramatic. The goal is to find a treatment whose benefits outweigh its risks.

Fifth, medicine is regulated for a reason.

Even if a computational breakthrough produced better candidates, those candidates would still require experimental validation, clinical trials, and regulatory review.

So the claim is not:

$P = NP$ would automatically cure every disease.

The claim is:

$P = NP$ with a practical low-degree algorithm could radically accelerate the search for candidate treatments.

That is less magical.

It is also more believable.

Important caveat

A powerful search algorithm is only as useful as the model and verifier it is given. If the disease model is wrong, the algorithm may optimize the wrong thing extremely efficiently.

18. Why This Matters for Diseases We Cannot Cure Today

Many diseases remain incurable for several different reasons.

Sometimes we do not understand the disease well enough.

Sometimes the target is known, but we cannot find a safe intervention.

Sometimes the disease involves many pathways at once.

Sometimes the treatment must cross a biological barrier, such as the blood-brain barrier.

Sometimes the disease evolves, mutates, or escapes treatment.

Sometimes the right answer may be a combination, not a single drug.

In mathematical terms, there may be a solution:

$$\exists x \quad V(D, x) = 1,$$

but current methods cannot find it efficiently.

This is the gap between existence and discovery.

A low-degree $P = NP$ algorithm would matter because it could narrow that gap.

It could allow scientists to search chemical spaces, protein spaces, treatment-combination spaces, and patient-specific treatment spaces in ways that are currently impossible.

Some diseases would still remain hard because the biology is poorly understood.

But for diseases where the bottleneck is search, the impact could be enormous.

19. Not a Magic Wand, but a Search Engine for Biology

Perhaps the best analogy is not a magic wand.

It is a search engine.

Not a search engine for web pages, but for biological possibilities.

You give it a disease model, a set of constraints, and a definition of what counts as a promising candidate. It searches the enormous space of possible interventions and returns candidates that satisfy the constraints.

Of course, the analogy has limits.

A web search engine can show you a web page. A $P = NP$ drug-search algorithm would show you a hypothesis.

That hypothesis would still need to be tested.

But good hypotheses are precious.

In drug discovery, a good candidate can save years. A better search process can redirect entire research programs. A candidate that humans would never have guessed can open a new therapeutic path.

This is why the computational breakthrough would matter.

Not because it replaces medicine.

Because it gives medicine a sharper way to ask:

> *What should we try?*

References

[1] Clay Mathematics Institute. *P vs NP*. claymath.org/millennium/p-vs-np.

[2] U.S. Food and Drug Administration. *Drug Development Process*. fda.gov.