A Polynomial Time Algorithm for 3SAT

LIZHI DU, College of Computer Science and Technology, Wuhan University of Science and Technology

By creating new concepts and methods—the checking tree, long unit path, direct contradiction unit pair, indirect contradiction unit pair, additional contradiction unit pair, two-unit layer and three-unit layer—we successfully transform solving a 3SAT problem to solving 2SAT problems in polynomial time. Each time, we add only one layer of the three-unit layers to the two-unit layers to calculate 2SAT paths, respectively. The key is as follows: in each 2SAT path, any two units cannot be a direct contradiction unit pair and cannot be an indirect contradiction unit pair and additional contradiction unit pair. This guarantees that all of the 2SAT paths we got, respectively, can shape at least one long path without contradictions. Thus, we proved that NP = P.

CCS Concepts: • **Theory of computation** → *Design and analysis of algorithms*; *Parameterized complexity and exact algorithms*;

Additional Key Words and Phrases: Computational complexity, computer algorithm, NP, P, 3SAT

ACM Reference format:
Lizhi Du. 2021. A Polynomial Time Algorithm for 3SAT. *ACM Trans. Comput. Theory* 13, 3, Article 15 (July 2021), 13 pages.
https://doi.org/10.1145/3460950

1 INTRODUCTION

In 1971, Cook proved the first NPC, the 3SAT problem [1]. Since then, "is NP equal to P?" as a great problem has caused a lot of interest and arguments. At present, most well-known authorities in this area tend to think that NP is not equal to P [2, 3]. It is absolutely certain that the authorities have no strong basis for this view, but this view seems to have been tacitly accepted by most people.

As a result, various academic papers often talk about NP, especially NPC, directly declaring that there can be no polynomial time algorithm. Such acquiescence is undoubtedly harmful.

Why do some experts always like to assert something? How many experts have asserted something in the past and later these assertions have been disproven by new achievements?

Also, there are famous scientists agreeing that NP = P. Hilbert, a great mathematician of the 20th century, has a famous saying: "we must know; we will know." It can be seen that Hilbert essentially agreed that NP equals P. Many mathematical problems in human history, including Hilbert's famous 23 mathematical problems, are constantly being solved. Isn't it a confirmation that NP equals P?

Authors' Address: L. Du, Room 602, door 2, building 1, Faculty dormitory of Wuhan University of science and technology, Huang Jia Hu Road, Hongshan District, Wuhan, PRC, 430065; email: dulizhi@wust.edu.cn.

1942-3454/2021/07-ART15 $15.00
https://doi.org/10.1145/3460950

From the heuristic point of view, any NPC problem can be reduced to any other NPC problem in polynomial time. That is to say, every distance between two NPC problems is polynomial. The fact itself strongly shows that NP problems have a unified solution law and difficulty, and their solution difficulty should be polynomial order of magnitude. The difference of an attribute value between any group of individuals in the objective world is usually in the same order of magnitude as the absolute value of an individual attribute. For example, one adult weighs 100 pounds, and the difference between a very fat man and a very thin man is also 100 pounds. Similarly, the weight of an ant is in grams, and the difference between a big ant and a small ant is also in grams. Of course, these are not strictly proven conclusions.

We develop a polynomial time algorithm for 3SAT. The insights of this article are based on the following concepts and methods: the checking tree, long path, direct contradiction pair, indirect contradiction pair, additional contradiction pair, contradiction property, two-unit layer and three-unit layer. We successfully transform solving a 3SAT problem to solving 2SAT problems in polynomial time. The keys are as follows: the additional contradiction pair's two properties and the proofs for Lemma 1 to Lemma 4 and Corollary 1 based on the two properties.

2 ALGORITHM AND PROOF

A 3SAT contains n variables and m clauses. Each clause contains three variables. We call each of them a unit (we call it a unit and not a literal due to the fact that we treat each one as a different one and that the algorithm is not limited in 3SAT). Thus, there are $3m$ units in all of the clauses. Now, we change this 3SAT to a path-finding problem. There are m+1 cities: c_0, c_1, ... c_{m}. From c_{i} to c_{i+1} ($i = 0, 1, \ldots m{-}1$), there are 3 different roads. We call each road a unit. Thus, there are $3m$ units. Now, we want to find a path from c_0 by $c_1, \ldots c_{m-1}$ to c_{m}. We call such a path a long path. There are 3^{m} different possible long paths. A long path contains m units. However, a lot of two units have contradictions. These two units cannot be in the same path. Apparently, any two units of the *3* roads from c_{i} to c_{i+1} ($i = 0, 1, \ldots m{-}1$) cannot be in the same path. There are a lot of other two units that cannot be in the same path. For any two units, we know they have or do not have contradictions (in 3SAT, a variable x and $-x$ have contradictions). The question now is how to find a long path from c_0 to c_{m}.

If two units have contradictions, we say that one unit destroys the other one.

If a unit destroys all of its *3* possible sons, we delete this unit. Thus, we suppose that each unit cannot destroy all of its *3* possible sons, that is, it does not destroy or only destroys one of them.

Suppose that from city c_0 to c_1, the *3* roads are a_{11}, a_{12}, a_{13}, from c_1 to c_2, they are $a_{21}, a_{22}, a_{23}, \ldots$, and from c_{m-1} to c_{m}, they are a_{m1}, a_{m2}, a_{m3}.

The checking tree contains three roots a_{11}, a_{12}, and a_{13}. The roots are in **layer 0**. We say layer 0 is the **highest layer** or the first layer in the tree. a_{21}, a_{22}, a_{23} are in layer 1 and then a_{31}, a_{32}, a_{33} are in layer 2. Layer 1 is higher than layer 2, and so on. In each layer, the three units are **brother units** and are brothers of each other.

There are m clauses, each clause containing three units. Thus, in the checking tree, finally, there are m layers and each layer contains at most three units. A layer is versus a clause. All units are $3m$. We call a lower layer a descendant layer of a higher layer. Also, the latter is the ancestor of the former.

A checking tree contains i layers. Each layer contains three possible units. Each time, we add one layer's units to the checking tree, one by one. We first add the first layer's three units a_{11}, a_{12}, a_{13} one by one. Then, we add a_{21}, a_{22}, a_{23}, and so on. We call the layer we currently calculate the **current layer**. The current layer's three units are $a_{i1}, a_{i2}, a_{i3} (i > 2)$. We call a path from any one unit of the current layer to any one unit of a_{11}, a_{12}, a_{13} the **current long path**. Each layer has one unit in this path and all units in this path do not have contradictions.

If all possible current long paths do not contain a unit u, we call u an **utterly destroyed unit**. It will always be an utterly destroyed unit later.

If two units u_1 and u_2 are the same variable, but one is positive and the other one is negative, we say that u_1 **directly destroys** u_2 (also u_2 destroys u_1). If any one current long path cannot contain both u_1 and u_2 at the same time, we say that u_1 **indirectly destroys** (if does not directly destroy) u_2. For both cases, we also say that one destroys the other one. If u_1 destroys u_2, it will always destroy u_2 later.

Each time, we add one layer and then calculate. We add and calculate each unit of this layer one by one. We call it **the new added unit**. For each layer, we have to calculate all possible new utterly destroyed units. For every unit pair—that is, for any two units in different layers—we have to calculate whether one indirectly destroys the other one. For a unit pair, if one destroys the other one, we call it a **contradiction unit pair** or **a contradiction pair**. For direct destroying, it is a **direct contradiction pair** and for indirect destroying, it is an **indirect contradiction pair**.

In the checking tree, for k consecutive layers, we take one unit from each layer. If they do not destroy each other, we call these k units a **part path,** or a path. In any part path, any two units cannot have contradictions.

Thus, a checking tree contains i layers. Each layer contains at most three units. If a unit is an utterly destroyed unit, we do not keep it in the checking tree. For the checking tree, we have to remember all utterly destroyed units and all indirect contradiction unit pairs.

Now, the current layer is the ith layer. Its three units are a_{i1}, a_{i2}, a_{i3}. We have got all utterly destroyed units and all contradiction unit pairs. We now add and calculate the $i+1$th layer. Its three units are a_{i+11}, a_{i+12}, a_{i+13}. We add each one to the checking tree, respectively. We first add the unit a_{i+11} to the checking tree.

Now, the old utterly destroyed units are still utterly destroyed units and the old contradiction unit pairs are also contradiction unit pairs. We have to calculate the new utterly destroyed units and the new indirect contradiction unit pairs after the a_{i+11} is added.

In the first layer (highest layer), there are three units and there are also three units in the lowest layer. Thus, there are 9 (3 times 3) kinds of **long unit paths**, or **long paths** (current long paths). Some of them may be destroyed. For each kind of long path, we calculate each unit pair and remember whether there is a path of this kind that contains the unit pair, that is, whether one unit destroys the other one for this long path. Any consecutive part of a long path is a part path.

If the checking tree does not contain a unit that is the same variable as a_{i+11} but with a different sign, that is, if a_{i+11} is the variable v, any other unit is not the variable $-v$, then there are no new utterly destroyed units and no new contradiction unit pairs (for a_{i+11}). If the checking tree contains that unit, then for a new long path that contains $a_{\mathrm{i}+11}$, any unit whose variable is $-v$ cannot be in this path.

If a layer contains only one unit (others are destroyed and deleted), we call it a **one-unit layer**. In this way, we have a **two-unit layer, three-unit layer** and **zero-unit layer**. We only consider two-unit layers and three-unit layers, so we call all layers which contains two units or one unit two-unit layers. Other layers are three-unit layers.

2.1 Algorithm 1: Calculating New Contradiction Pairs

We want to calculate new contradiction pairs after we add a new unit to the checking tree. Suppose that the new added unit is $a_{\mathrm{i}+11}$ and its variable is v.

Now, we copy the checking tree to a temporary checking tree and we will delete some units in the temporary checking tree. For Algorithm 1, we need the temporary checking tree.

For two units x and y, the unit pair x and y is not a contradiction pair before a_{i+11} is added. u_1, u_2, u_3 are three units in a layer. u_1 is the variable $-v$. We now want to know whether x, y can be in the same long path again after we delete all of the $-v$. Suppose that x, y and u_1 are in the same old long path. This algorithm calculates whether x and y are a new contradiction pair after a_{i+11} is added. For each unit pair that is not a contradiction pair before a_{i+11} is added, we apply this algorithm one time.

For the temporary checking tree, we delete the layer that contains x and the layer that contains y. We find all layers that contain $-x$ and delete all $-x$. We call these layers the first kind of layers. We find all layers that contain $-y$ and delete all $-y$. We call these layers the second kind of layers. We find all layers that contain $-v$ and delete all $-v$. We call these layers the third kind of layers. We call all of the three kinds of layers two-unit layers. They may contain one-unit layers. Then, we do 2SAT for the three kinds of layers. We have to get these kinds of **part path** only in the three kinds of layers: any two units in the part path are a unit pair that do not destroy each other for a long path, that is, the two units can be in this long path at the same time. In each two-unit layer, there is one unit in this part path. We call this part path a **2SAT path** and call this property **the 2SAT path property**. If we cannot get such a 2SAT path, x and y destroy each other and we end this algorithm.

Now, we add each three-unit layer to the bottom of the two-unit layers and then calculate the 2SAT, respectively. Because there is only one three-unit layer, this does not affect the 2SAT calculation in polynomial time. In this calculation, each unit of a three-unit layer must be in at least one 2SAT path that has the 2SAT path property. Otherwise, we delete this unit temporarily. If some units in a three-unit layer are deleted, then we put this layer in the two-unit layers. If we cannot get such a 2SAT path, x destroys y and we end this algorithm. In the two-unit layers, each unit pair in a 2SAT path must be always in a 2SAT path when we add every three-unit layer, that is, if when we add one three-unit layer, we cannot find such a 2SAT path that contains this unit pair and when we add another three-unit layer, there is such a 2SAT path that contains the pair, then we do not think it is a useful 2SAT path. For each three-unit layer, there is at least one unit which is in the same 2SAT path with such a unit pair. We call all of these the **three-unit layer property**. Thus, for any one three-unit layer, if a unit pair cannot be in a 2SAT path with any one of the three units in this layer, we call this unit pair an additional contradiction unit pair or **an additional contradiction pair**. For any three-unit layer, a 2SAT path cannot contain an additional contradiction pair.

Thus, a 2SAT path cannot contain any one direct contradiction pair, indirect contradiction pair, or additional contradiction pair. We call this the **contradiction property**, including **direct contradiction property, indirect contradiction property**, and **additional contradiction property**.

We call a unit pair that is not a contradiction pair **a useful pair**, that is, for any two units, if each three-unit layer has at least one 2SAT path that contains these two units, we call the two units **a useful pair**. For a unit's 2SAT path, if any two units in it are a useful pair, we call it **a useful 2SAT path**. In this algorithm, every 2SAT path must be a useful 2SAT path.

When we put a three-unit layer to the two-unit layers to calculate, we call all 2SAT paths that contain one unit of the three-unit layer this **unit's 2SAT paths** (or this **unit has these 2SAT paths**) and call all the 2SAT paths of the three units together this three-unit **layer's 2SAT paths** (or this **layer has these 2SAT paths**).

Finally, if each three-unit layer has at least one useful 2SAT path, then any two layers' useful 2SAT paths contain the same useful pairs, that is, for each unit pair in any one 2SAT path of a three-unit layer, we always can find a 2SAT path of any other one three-unit layer, which also contains the unit pair.

For the additional contradiction pair, we have **two properties**:

Property 1. Considering two units in the two-unit layers that are not a direct contradiction pair or an indirect contradiction pair. If there is one (or more) three-unit layer that does not have a 2SAT path containing the two units, we call the two units an additional contradiction pair. In a useful 2SAT path, any two units in it can be in the same long path and cannot be an additional contradiction pair. Also, for a unit t's 2SAT paths, the unit pair t with any one unit in these 2SAT paths can be in the same long path.

Property 2. Considering a unit pair in two-unit layers that is not a contradiction pair, that is, which is a useful pair. For each unit in three-unit layers, we remember whether this unit has a 2SAT path containing the unit pair. Then, for calculating a unit u's one useful 2SAT path, it cannot contain any contradiction pair. Also, we have to check every other old three-unit layer whose 2SAT paths have been calculated. Considering a unit pair p in the 2SAT path, if in another three-unit layer, each unit that has the 2SAT paths to contain the unit pair p destroys u, that is, this unit and u are a direct or indirect contradiction pair, then for the unit u, p is not a useful pair, that is, u's useful 2SAT paths cannot contain p. Of course, after calculating this layer, if there are new additional contradiction pairs, we have to check and calculate each old three-unit layer's 2SAT paths again.

Finally, we can determine that x and y do not destroy each other, that is, they are a unit pair that can be in the same long path.

We now prove why this algorithm is correct.

First, we define some concepts to be used in the following lemmas and proofs. Let p_1 and p_2 be two part paths in the same two-unit layers. The two units in a two-unit layer are brother units. The brother unit of each unit in p_1 is in p_2 and vice versa. We call p_1, p_2 **two brother part paths** in some two-unit layers. p_3 and p_4 are also two brother part paths in some two-unit layers—so are p_5 and p_6. These layers are all the two-unit layers. (We suppose the two-unit layers are more than 5. If they are less than 5, we can calculate all 2SAT paths in the two-unit layers. Then, each time, we check one path for each of the three-unit layers.) t_1, t_2, t_3 are three units in a three-unit layer—say, layer 1. So are t_4, t_5, t_6 (layer 2), t_7, t_8, t_9 (layer 3), and t_{10}, t_{11}, t_{12} (layer 4). Each time, we add one three-unit layer to the two-unit layers to calculate 2SAT paths.

Lemma 1. *For two three-unit layers, layer 1 and layer 2, if t_1 has a useful 2SAT path p and t_1 destroys t_4, then at least one unit of t_5 and t_6 must also have the 2SAT path p and must not destroy t_1. When t_1 has a useful 2SAT path p, there are two ways to let layer 2's 2SAT paths contain all unit pairs in p: (1) one unit in layer 2 also has this 2SAT path and does not destroy t_1; and (2) we separate this 2SAT path into three parts—each of t_4's, t_5's, and t_6's one 2SAT path contains different two of the three parts and t_1 does not destroy t_4, t_5, and t_6. Recursively, in order to let a unit's 2SAT paths contain all unit pairs in a part path, we can separate the part path into three parts so that each of the unit's three 2SAT paths contain different two of the three parts.*

Proof. First, if t_1 also destroys t_5, then t_6 must have the 2SAT path p and must not destroy t_1. By the property 2, t_1 cannot also destroy t_6; otherwise, t_1 cannot have a useful 2SAT path. Also, any unit pair in p must be in t_6's 2SAT paths. Thus, the lemma holds. If t_1 destroys only t_4, then all unit pairs in p must be in the 2SAT paths of t_5 and t_6. If both t_5 and t_6 destroy some units in p, then some unit pairs in p cannot be in the 2SAT paths of t_5 and t_6. Thus, at least one of t_5 and t_6 cannot destroy any unit in p. If layer 2 does not have the 2SAT path p, then the only way that layer 2's 2SAT paths contain all unit pairs in p is as follows: we separate this 2SAT path p into three parts, each of t_4's, t_5's, and t_6's one 2SAT path contains different two parts and t_1 does not destroy t_4, t_5, and t_6. Thus, the lemma holds. □

Lemma 2. *If t_1 has a useful 2SAT path $p_1p_3p_5$, in order to let layer 2's 2SAT paths contain all unit pairs in $p_1p_3p_5$, there are three possible kinds: kind 1—one (or more) unit in layer 2 also has the 2SAT path $p_1p_3p_5$ and does not destroy t_1; kind 2—t_1 does not destroy any one of t_4, t_5, t_6, and t_1 destroys at least two of the three part paths p_2, p_4, p_6 or t_1 does not destroy any one of p_2, p_4, p_6; kind 3—t_1 does not destroy any one of t_4, t_5, and t_6, and t_1 destroys only one of the three part paths p_2, p_4, p_6. For kind 2, layer 2 and layer 1 must have at least the same one 2SAT path. Also, for different layers, we have* ***the same case property*** *for kind 2 as stated as follows.*

Proof. We prove only kind 2. For kind 3, it may lead to an exceptional case that is very special, which we will discuss in Lemma 4. □

We discuss kind 2 in several cases.

We first suppose that t_1 destroys p_2, p_4, and p_6 and layer 1 does not have the 2SAT path $p_2p_3p_5$.

Case 1: Now we suppose that t_1's one 2SAT path is $p_1p_3p_5$. In order to let layer 2 not have such a 2SAT path but contain all of the unit pairs in this path, one way is t_4's 2SAT path $p_2p_3p_5$ (t_4 destroys p_1), t_5's 2SAT path $p_1p_4p_5$ (t_5 destroys p_3), and t_6's 2SAT path $p_1p_3p_6$ (t_6 destroys p_5).

Now, t_1 destroy some units (for convenience, suppose that it destroys all units) in p_2, in p_4, and in p_6. t_2 and t_3 cannot destroy units in p_2 and p_4 or p_6 because, if so, layer 1's 2SAT paths cannot contain such a unit pair whose one unit is in p_2 and the other one is in p_4 or p_6. In order to let layer 1 not have the same 2SAT path $p_2p_3p_5$ but contain the unit pairs one in p_2 and one in p_3p_5, one way is t_2 has the 2SAT path $p_3p_6p_2$ and t_3 has $p_4p_5p_2$. In order to let layer 1's 2SAT paths contain unit pairs in p_1p_4 and in p_1p_6, t_2 and t_3 cannot destroy p_1.

In order to let layer 2's 2SAT paths contain unit pairs that t_2's and t_3's 2SAT paths contain, t_5's and t_6's 2SAT paths must contain p_2 or part of p_2 (or t_4's 2SAT paths contain p_4, p_6). In this case, suppose that they contain p_2. Then, both layers would have the 2SAT paths $p_2p_4p_5$, $p_2p_3p_6$, $p_1p_4p_5$, $p_1p_3p_6$.

Case 2: When we said that t_5 has the 2SAT path $p_1p_4p_5$ and t_3 also has this 2SAT path, it means that t_5 and t_3 do not destroy units in $p_1p_4p_5$, but $p_1p_4p_5$ may not be a 2SAT path. Thus, by Lemma 1, t_5 or t_3 may have three or more 2SAT paths containing all unit pairs in $p_1p_4p_5$. **Note that t_5's such 2SAT paths must be the same as t_3's such 2SAT paths**. This is because we suppose that t_5 destroys p_3 and it does not have the 2SAT path $p_1p_3p_5$, but t_5's 2SAT paths must contain all unit pairs in p_1p_5; thus, t_5 cannot destroy p_4. If $p_1p_4p_5$ is not a 2SAT path, by Lemma 1, we can separate p_1p_5 into three parts and each of t_5's three 2SAT paths contain different two of the three parts together with p_4. Based on the above two properties, t_3 also has these three 2SAT paths.

Case 3: If t_2 and t_3 destroy some units in p_2, in order to let their 2SAT paths contain every unit pair of which one unit is in p_2 and the other in p_3p_5, t_2 and t_3 must have the 2SAT paths $p_{21}p_{12}p_3p_5$ and $p_{11}p_{22}p_3p_5$, respectively. Considering t_5's and t_6's 2SAT paths, t_2 and t_3 must not destroy p_4, p_6. Also, t_5 and t_6 cannot destroy p_2. Thus, t_5 and t_6 must also have the 2SAT paths $p_{21}p_{12}p_4p_5$ and $p_{11}p_{22}p_3p_6$, respectively. t_2 and t_3 also have the two 2SAT paths. P_{11} and p_{12} together is p_1. P_{21} and p_{22} together is p_2. p_{11} and p_{21} are brother part paths, as are p_{12} and p_{22}.

Case 4: In order to let layer 2 not have the 2SAT path $p_1p_3p_5$ but contain all the unit pairs in this path, another way is as follows: t_4's 2SAT path $p_{21}p_{12}p_3p_5$ (t_4 destroys p_{11}), t_5's 2SAT path $p_1p_{41}p_{32}p_5$ (t_5 destroys p_{31}) and t_6's 2SAT path $p_1p_3p_{62}p_{51}$ (t_6 destroys p_{52}). p_{31} and p_{32} together is p_3. p_{41} and p_{42} together is p_4. p_{51} and p_{52} together is p_5. p_{61} and p_{62} together is p_6. p_{31} and p_{41} are brother part paths, as are p_{32} and p_{42}, p_{51} and p_{61}, p_{52} and p_{62}. In order to let layer 1 not have the same 2SAT path $p_{21}p_{12}p_3p_5$ but contain the unit pairs that the path contains, one way is as follows: t_2 has the 2SAT path $p_3p_6p_{21}p_{12}$ and t_3 has $p_4p_5p_{21}p_{12}$. In order to let layer 1's 2SAT paths contain unit pairs in p_1p_{41} and in p_1p_{62}, t_2 and t_3 cannot destroy p_1. Also, t_2 cannot destroy p_{41}, p_{51} and t_3 cannot destroy p_{32}, p_{62}. Then, the two layers have the same 2SAT paths: $p_1p_{41}p_{32}p_5$, $p_1p_3p_{62}p_{51}$.

Case 5: for this case, we consider three three-unit layers layer 1, layer 2 and layer 3. For more than three three-unit layers, we can separate them into three sets: set 1 is like layer 1, set 2 is like layer 2 and set 3 is like layer 3. Suppose t_1 has the useful 2SAT path $p_1p_3p_5$. t_4 also has this useful 2SAT path and t_4 does not destroy t_1. t_1 destroys t_7 and t_4 destroys t_8. t_7 and t_8 do not destroy any unit in $p_1p_3p_5$, but they do not have the useful 2SAT path $p_1p_3p_5$. In order to let the useful 2SAT paths of t_7 and t_9 contain all unit pairs in $p_1p_3p_5$, the way is: t_7 has two useful 2SAT paths $p_2p_3p_5$ and $p_1p_3p_6$. t_9 has a useful 2SAT path $p_1p_4p_5$. Because t_7 destroys t_1, t_2 or t_3 must have the 2SAT paths $p_2p_3p_5$ and $p_1p_3p_6$. If t_2 has the useful 2SAT paths $p_2p_3p_5$ and $p_1p_3p_6$, then t_2 and t_7 must also have the useful 2SAT path $p_1p_3p_5$. Anyway, for this case, we also can get that each 3 unit layer has at least the same one 2SAT path (also see the proof of lemma 3).

We now explain why this lemma holds. We consider only case 1 and case 3. Other all possible cases are logically the same. In order to let layer 2 not have the 2SAT path $p_1p_3p_5$ but contain all the unit pairs in this path, there are two different ways: case 1 and case 4. We explain only case 1. Case 4 is in the same logic. In order to let layer 1 not have the same 2SAT path $p_2p_3p_5$ but contain the unit pairs one in p_2 and one in p_3p_5, there are two different ways: case 1 and case 3. For case 1, in order to let layer 1's 2SAT paths contain unit pairs in p_1p_4 and p_1p_6, t_2 and t_3 cannot destroy p_1. For case 3, because p_1 is separated into two parts in t_2's and t_3's 2SAT paths and so is p_2, in order to let layer 1's 2SAT paths contain unit pairs in t_5's and t_6's 2SAT paths, both t_2 and t_3 cannot destroy p_4, p_6. Also, t_5's and t_6's 2SAT paths must contain unit pairs in p_2p_4 and in p_2p_6. Then, for the two different cases, two layers must have the same 2SAT paths.

Note that for the above four cases, we suppose that t_1 destroys p_2, p_4 and p_6 and then the two layers have the same two 2SAT paths. If t_1 only destroys p_2, p_4, in the same way, we can see that the two layers have at least the same one 2SAT path. We suppose that layer 1 does not have the 2SAT path $p_2p_3p_5$. In the same way, we also can suppose that layer 1 does not have the 2SAT path $p_1p_4p_5$ or $p_1p_3p_6$. If t_1 does not destroy any one of p_2, p_4, p_6, t_1 would have the 2SAT paths that layer 2 has. We call such a layer an unaffected layer.

We now suppose that t_7 in layer 3 also has the 2SAT path $p_1p_3p_5$. We consider layer 3 and layer 2. Note that for kind 2, in order to let all unit pairs in any one three-unit layer's useful 2SAT paths are also contained in each other layer's 2SAT paths, when layer 1 and layer 2 is in any case of the above cases, layer 3 and layer 2 must also in the same case (except the unaffected layer). We call this **the same case property. For different layers, the same case property is the key**.

Lemma 3. *For k (k >= 2) three-unit layers, if each layer has the useful 2SAT path p, then there is a part path in the k layers (i.e., the part path contain k units which do not destroy each other and each unit is in each of the k layers) and each unit in the part path has the same one 2SAT path (may or may not be the same as p).*

Proof. Let $p_1p_3p_5$ and $p_2p_4p_6$ be the two-unit layers. p_1 and p_2 are brother part paths. So are p_3 and p_4 as well as p_5 and p_6. We call $p_1p_3p_5$ the first kind 2SAT path and units in this path are the first kind units. We call $p_2p_4p_6$ the second kind 2SAT path and units in this path are the second kind units. Any one unit in the first kind 2SAT path has a brother unit in the second kind 2SAT path. The two brother units are in the same layer. Any one part path of the first kind 2SAT path has a brother part path in the second kind 2SAT path. The three-unit layers have k layers and each layer contains three units: the first one, the second one and the third one. t_1 is a unit in a three-unit layer.

Suppose that t_1 has the first kind 2SAT path and t_1's 2SAT paths also contain p_2 (only p_2, i.e., they do not contain other units in the second kind 2SAT path. Suppose that any other unit which has the first kind 2SAT path has the 2SAT paths which contain p_2 (only p_2), or, contain p_2 (or part of p_2, or non of p_2) and some other units in the second kind 2SAT path.

Now if there is a long path which contains t_1 and which only contains such units in three-unit layers each of these units has the first kind 2SAT path, this means that the units which are in the long path and also in the three-unit layers have the same one 2SAT path, the first kind 2SAT path.

If there is a long path which contains t_1 and also contains some units that do not have the first kind 2SAT path and for each of such units, its 2SAT paths do not contain such units which are in the second kind 2SAT path but are not in p_2, and also each of such units has the same one 2SAT path, then the units in the long path and also in the 3 unit layers have the same one 2SAT path.

If each of all long paths which contains t_1 contains one (or more) unit which does not have the first kind 2SAT path and whose 2SAT paths always contain a unit v in the second kind 2SAT path which is not in p_2 (we suppose the 2SAT paths do not contain the unit v's brother unit), this case does not fulfil the rule: the two units t_1 with any one unit in the first kind 2SAT path can be in the same long path. So this case cannot happen.

Suppose that there are two long paths. One contains t_1 and t_i, the other one contains t_1 and t_k. t_i and t_k are two units in three-unit layers. ti and t_k do not have the first kind 2SAT path. t_i's 2SAT paths contain p_{41} (part of p_4) and t_k's 2SAT paths contain p_{61} (part of p_6). They do not contain other units in the second 2SAT paths. Considering these two long paths, we can see it is possible that the unit pair of t_1 with any one unit in the first kind 2SAT path can be in the same long path.

Then in order to let each layer's 2SAT paths contain the unit pairs related to p_{41} and p_{61} (i.e., at least one unit of each pair is in p_{41} or p_{61}), each layer's 2SAT paths must contain p_{41} and p_{61}. **Please note: we cannot separate** p_{41} (or p_{61}) into three parts and let each part (suppose that their brother units are not in) be in three different unit's 2SAT paths, because if so, for the above two long paths, we cannot get that: the unit pair t_1 with any one unit in the first kind 2SAT path can be in the same long path. Thus in each 3 unit layer, at least one unit's 2SAT paths contain p_{41} and at least one unit's 2SAT paths contain p_{61}. If one unit's 2SAT paths contain both p_{41} and p_{61}, they must also contain the brother part path of p_{41} and the brother part path of p_{61}.

We can suppose that each 2SAT path does not contain both p_{61} and p_{41} together and that p_{61} or p_{41} does not separate into three parts in three different 2SAT paths. If a 2SAT path contains both p_{61} and p_{41} (or part of p_{61} or p_{41}), each layer's 2SAT paths must also contain unit pairs in it and then we still can prove in the same way as following.

Now for each three-unit layer, at least one unit's 2SAT paths contain p_{61} and another one unit's 2SAT paths contain p_{41}. If for each layer, only one unit's 2SAT paths contain p_{61} and another one unit's 2SAT paths contain p_{41}, then any two units whose 2SAT paths contain p_{61} must not destroy each other. So we can get one part path for all three-unit layers in which each unit is in each different layer of the three-unit layers and each unit has the same 2SAT path which contains p_{61}.

If in some layers, two units' 2SAT paths contain p_{61} and the third one unit's 2SAT paths contain p_{41}, then we still can get a part path in which each unit has the same 2SAT path which contains p_{41}. We also can get a part path in which each unit has the same 2SAT path which contains p_{61}, because if not so, then for a unit x whose 2SAT paths contain p_{61}, each long path which contains x must contain such a unit whose 2SAT path contains p_{41}, then we cannot get that every unit pair of x and one unit in the 2SAT path which contains p_{61} can be in the same long path. We call these the property of a part path with the same 2SAT path.

Please note: the case 5 in lemma 2 does not affect this proof, because in case 5, u_7 and u_8 do not destroy any unit in $p_1p_3p_5$, i.e., u_7 or u_8 with any one unit in $p_1p_3p_5$ can be in the same long path.

In the same way, for all other possible cases, we still can get this result. □

Corollary 1. *If without the exceptional case caused by kind 3 in Lemma 2, for all three-unit layers, if they have useful 2SAT paths, then there is a part path in all three-unit layers (i.e., each unit of the* *part path is exactly in each of all three-unit layers and they do not destroy each other) and each unit in the part path has the same one 2SAT path.*

Proof. We prove it by induction. Suppose that for r ($r \geq 2$) layers, the corollary holds. We call these r layers holding layers and call the part path in these layers the holding part path. Each time, we add one other layer l_1 in the three-unit layers to the holding layers. Each time we consider two layers, one is in the holding layers and one is l_1. By Lemma 2 and **by the same case property** in Lemma 2, we can immediately get that for r+1 layers, each layer has at least the same one 2SAT path. Then, by Lemma 3, the corollary also holds. □

Lemma 4. *For the kind 3 in Lemma 2, it may lead to an exceptional case. For this case, even if each three-unit layer has one or more useful 2SAT paths, we still may not get a long path. This is a very special case. We can solve this case in polynomial time.*

Proof. Let a_1, a_2 be in the same two-unit layer and b_1, b_2 be in the same two-unit layer. p_1 and p_2 are brother part paths in two-unit layers. p_3 and p_4 are also two brother part paths in two-unit layers, as are p_5 and p_6. These layers are all two-unit layers. t_1, t_2, t_3 are three units in a three-unit layer—say, layer 1. So are t_4, t_5, t_6 (layer 2), t_7, t_8, t_9 (layer 3), and t_{10}, t_{11}, t_{12} (layer 4). □

Each time, we add one three-unit layer to the two-unit layers to calculate 2SAT paths. Let a_1, b_1, p_1, p_3, p_5 be a 2SAT path of t_1. It is layer 1's one 2SAT path. We analyze the exceptional case.

Suppose that t_1's 2SAT path is $a_1b_1p_1p_3p_5$, t_2's 2SAT path is $a_1b_1p_2p_4p_6$, t_4's 2SAT path is $a_1b_1p_1p_4p_6$, and t_5's 2SATpath is $a_1b_1p_2p_3p_5$, but t_3 and t_6 destroy a_1, b_1. Each 2SAT path of t_3 and t_6 contains a_2, b_2 and does not contain a_1, b_1. Each 2SAT path of t_9 and t_{12} contains a_1, b_1 and does not contain a_2, b_2. Suppose t_7's 2SAT path $a_2b_2p_1p_4p_6$, t_8 $a_2b_2p_2p_3p_5$, t_{10} $a_2b_2p_1p_3p_6$, and t_{11} $a_2b_2p_2p_4p_5$. t_3, t_6, t_9, t_{12} contain all needed useful pairs (units in p_1 to p_6) so that each unit pair in a 2SAT path is a useful pair.

We separate all three-unit layers into two parts (can be more than two parts, but this does not affect our algorithm): part 1: each layer is like layer 1 (layer 2); part 2: each layer is like layer 3 (or layer 4). In this case, t_3's as well as t_6's 2SAT paths contains all useful pairs which the 2SAT paths of each layer's first two units of part 2 contain. t_9's as well as t_{12}'s 2SAT paths contains all useful pairs which the 2SAT paths of each layer's first two units of part 1 contain.

For the above exceptional problem, if each 2SAT path of t_9 and t_{12} contains a part path like a_1, b_1 and each 2SAT path of t_1, t_2, t_4 and t_5 also contains this part path. We call this the exceptional problem 1. If this part path is separated into three parts and for each other layer, each unit's 2SAT paths contain one of these three parts, we call this the exceptional problem 2.

After we calculated the 2SAT paths of every three-unit layer, in each 2SAT path, any two units are a useful pair. All other units which are not in useful pairs are temporarily deleted. The exceptional case is a very special case. Now we have to solve the exceptional problem 1 and exceptional problem 2. The condition for an exceptional problem happening is that in each of the three-unit layers, there is at least one unit which does not destroy all units of the same two-unit layers.

If a unit x in a 3 unit layer does not destroy both units in one two-unit layer, we say that this two-unit layer is x's one exceptional layer. For the above example, in addition to the shared exceptional layers, t_9's as well as t_{12}'s 2SAT paths may contain some units which t_6's or t_3's 2SAT paths also contain, but this does not affect our algorithm.

Now we can do the following steps to **solve the exceptional problem** discussed above:

(1) For each layer of the three-unit layers, we try to choose a unit which has exceptional layers. We call it a **chosen unit**. For each of the rest three-unit layers, choose such one unit which shares the most exceptional layers with former chosen units. We also call this unit a chosen unit. All chosen units, each of them is in each of the three-unit layers, must

share some exceptional layers. They are the final exceptional layers which each unit of the chosen units has. We call these chosen units a chosen unit set. We do not have to mind whether the units in a chosen unit set destroy each other, due to the reasons: 1) if the exceptional problems happen, all units in a chosen unit set do not destroy each other and a chosen unit does not destroy most other units; 2) if in one layer, more than one unit has the shared exceptional layers, choosing any one does not affect our method (choose the one which share the most exceptional layers). We may have to get more than one different chosen unit sets. Considering that we do not have to mind whether the units in a chosen unit set destroy each other and that for the three-unit layers, all the third units **share the most number of exceptional layers** (for some fixed layers), the number of exceptional layer sets is O(1) and the job to find these layer sets is not hard. The time for this job is $O(m^2)$. We first choose layer 1. Its three units are t_1, t_2, t_3. We choose t_3 and get t_3's exceptional layers. We temporarily delete such exceptional layers: in one (or more) three-unit layer, no unit has such exceptional layers. Then for the rest exceptional layers of t_3, each time we add one such layer to try to get a final exceptional layer set. The layer sets may be more than one but in O(1). For each of such exceptional layers, we only have to consider O(1) layer sets which may contain it. We do not have to calculate a lot of combinations. At last, for each chosen unit set (each unit in each three-unit layer, if we can get their shared exceptional layers), we do the following steps. If we cannot get a long path, we try other chosen unit sets. If we still cannot get a long path, we try t_1 and t_2 in layer 1 in the same way. For each three-unit layer, we say the chosen unit is the third unit. Other two units are the first and the second units.

(2) Choose layer 1 of the three-unit layers. Only consider the first and second units in this layer. Choose another three-unit layer and also only consider the first two units. If they can share a 2SAT path, we call it together with layer 1 a **layer set**. We call layer 1 the **start layer** of this set. For the layer set (each layer two units) together with all two-unit layers, we calculate 2SAT. We have to get at least one 2SAT path on them. Then, each time we add one other three-unit layer (only the first two units) to them (also to the layer set if we successfully get a shared 2SAT path) to calculate 2SAT and try to get at least one 2SAT path. For the rest three-unit layers which cannot be added into this layer set, i.e., each of them (only for the first two units) and the layer set cannot share the same one 2SAT path, we calculate another layer set in the same way. We call this layer set **the first kind** layer set. Please note: layers in different layer sets may overlap, but each start layer is a new layer which is not in former layer sets. For each layer set, find the rest three-unit layers and temporarily delete the first two units in these layers. We call these layers **the second kind** layer set. We put this second layer set (each layer one unit), and the two-unit layers together to calculate 2SAT. If we cannot get a shared 2SAT path, there is no shared 2SAT path for this case and we can try other first kind layer sets. If we can get a 2SAT path and all units in the second layer set (only the third unit) do not destroy each other, do the next step 3).

(3) Put the layers of this second layer set (each layer one unit, the third one), all other three-unit layers (each layer two units, the first two units), and the two-unit layers together to calculate 2SAT.

(4) For the second layer set, we add all other possible three-unit layers (only the third unit) to this second kind layer set which can share the same one 2SAT path. We call it the **final second kind layer set**. Then we put the layers of this final second layer set (each layer one unit), all other three-unit layers (each layer two units), and the two-unit layers together to calculate 2SAT.

Now we explain why the above steps can solve the exceptional problem. Units in a first kind layer set share a 2SAT path. If the number of the rest three-unit layers is 0, the result is enough. Now suppose it is not 0. If a third unit t_3 has a 2SAT path $p_1p_3p_5$, then for each of other three-unit layers, case 1: it also has the 2SAT path $p_1p_3p_5$; case 2: its three units have the 2SAT paths $p_1p_3p_6$, $p_1p_5p_4$ and $p_3p_5p_2$ respectively and t_3's 2SAT paths also contain p_2. p_1 and p_2, p_3 and p_4, p_5 and p_6 are brother part paths; case 3: t_3's 2SAT paths do not contain p_2 but both t_1's and t_2's 2SAT paths contain p_2. t_1, t_2 and t_3 are three units in the same layer. This layer also must have the 2SAT paths $p_1p_3p_6$ and $p_1p_5p_4$. For case 2 and case 3, each layer may have different 2SAT paths, but due to the above property 1 and property 2, at last, each layer must have the same two or more 2SAT paths. For the final second layer set, if it is like the case 1, we can get a shared 2SAT path for all the three-unit layers by step 4). If it is like the case 2, we can get a shared 2SAT path for all the third units by the above steps. If it is like the case 3, we can get a shared 2SAT path for the first two units in all 3 unit layers. Or, there is no such a shared 2SAT path due to the exceptional problem. If so and if part of the final second layer set have another 2SAT path which is like the case 1 and there is no exceptional problem for this case, then for the case 3, in order to let each layer's 2SAT paths contain the same unit pairs and also in order to let the exceptional problem happen, the first unit's 2SAT paths must contain $p_1p_3p_6$, p_2, p_4 and the second unit's 2SAT paths must contain $p_1p_5p_4$, p_2, p_6. Suppose part of the final second kind layer set (we call them the part layers) have another 2SAT path (say h_1), and the first or second unit of each other three-unit layer also has the 2SAT path h_1. For the case 3, we have two different kinds. Kind 1: h_1 contains p_6 (or p_4). If in the part layers, the first unit's and the second unit's 2SAT paths also contain p_6 (or p_4), then when we put the first two units of the first kind layer set, the third unit of the second kind layer set, and the two-unit layers together, we can get a long path. If in the part layers, the first unit's and the second unit's 2SAT paths do not contain p_6 and p_4, then when we choose p_1p_2, p_5p_6 as the exceptional layer set, i.e., all third units (different third units, i.e., another case's third units) have these exceptional layers, if we put all the first two units (another case's first two units) and the two-unit layers together (i.e., for this case, the first kind layer set includes all three-unit layers), we can get a long path. Kind 2: h_1 does not contain the whole p_6 or the whole p_4 or the whole p_2. Suppose h_1 contains $p_6p_1p_{52}$. p_{61} and p_{62} together is p_6. p_{51} and p_{52} together is p_5. If the third units in the part layers and the first unit in other layers share the 2SAT path h_1, then when we choose p_1p_2, and p_5p_6 as the exceptional layer set, if we put all the first two units (another case's first two units) and the two-unit layers together, we can get a long path. If in some layers, the first (or second) unit also shares the final exceptional layers and then the first (or second) unit and the third unit exchange, this does not affect our method, because we do the step 2), 3), 4). Suppose that for k_1 layers, at last, the first (or second) unit and the third unit exchange positions, we call such third units k_1 layers wrong third units. All three-unit layers are k layers. For the new first two units, if at most there are k_2 (k-$k_1 < k_2 < k$) layers which share another different 2SAT path, then the k_1 layers wrong third units must also share this 2SAT path. So we can get a long path by above steps. For a exceptional layer set, we only have to consider it one time. If in step 1), there are no exceptional layer sets, we do the step 2), 3), 4), and we still can get a long path.

Please note: by Lemma 1, Lemma 2, Lemma 3 and Corollary 1, if each three-unit layer has one or more useful 2SAT paths, then for any two three-unit layers, there is at least one unit in each layer that has the same one 2SAT path and do not destroy each other directly or indirectly. However, for three or more three-unit layers, there is a special case, as above. Due to the above property 1 and property 2, this is the only one kind of exceptional case. They are not exactly the same, but they are in this style and we can solve them in the same way. Except for this kind of exceptional case, if each three-unit layer has one or more useful 2SAT paths, then there is at least one unit in every layer that has the same one 2SAT path and these units do not destroy each other directly or indirectly.

If there are no three-unit layers because for all layers we can get a 2SAT path that does not contain $-x$ and $-y$, that is, which can contain x and y, the result is correct. If there is only one three-unit layer, it is obviously also correct. Suppose that there are more than one three-unit layers. In the algorithm, we first calculate 2SAT on the two-unit layers. Then, we add each three-unit layer to the bottom of the two-unit layers and calculate the 2SAT, respectively, as stated above.

Foe the proof, we supposed the worst cases. Even if the suppositions do not happen, it still does not affect our proof.

A unit in a three-unit layer cannot destroy both units in a two-unit layer because, if so, this unit would not have any 2SAT paths and, thus, we temporarily delete it.

In summary of the above lemmas and corollaries, we can get that the algorithm is correct.

The key for the polynomial is as follows: all possible 2SAT paths may be exponential, but all possible two-unit combinations are polynomial ($O(m^2)$). We do not have to remember all possible 2SAT paths but only remember all two-unit contradiction pairs, including direct contradiction pairs, indirect contradiction pairs, and additional contradiction pairs. To calculate a 2SAT path, the time is $O(m)$. The problem is that when calculating 2SAT paths for one three-unit layer, we may get new additional contradiction pairs (every such unit pair cannot be in the same useful 2SAT path). Then, we have to calculate former three-unit layers again and do not let each of these new contradiction pairs in the same useful 2SAT path. Thus, for each unit in three-unit layers, we consider calculating its 2SAT paths $O(m)$ times on average. Note that for each unit in three-unit layers, we may calculate more than one 2SAT path containing different unit pairs. However, on average, we still think that the time to calculate each unit's 2SAT paths each time is $O(m)$.

In the same way, we also calculate whether x destroys y when we add the unit a_{i+12}, a_{i+13}. For each of 9 kinds of long paths, we calculate each unit pair and remember whether there is such a path of this kind containing the unit pair.

Note that in this algorithm and proof, the indirect destroying is based on the old checking tree, that is, before the unit $a_{\mathrm{i}+11}$ is added. This is also for the concept long unit path.

2.2 Algorithm 2: Calculating New Utterly Destroyed Units

Let u_1, $u_2 u_3$ be three brother units and are not utterly destroyed units. u_1 is the variable $-v$. $a_{\mathrm{i}+11}$ is the variable v and is the new added unit. If a unit is destroyed by both u_2 and u_3, then this unit is a new utterly destroyed unit for $a_{\mathrm{i}+11}$. If more than one layer contains the variable $-v$ because after Algorithm 1, we have got all of the new contradiction pairs, each unit that is destroyed by both two units (except the $-v$) in such a layer is a new utterly destroyed unit.

In this way, we also calculate new utterly destroyed units for $a_{\mathrm{i}+12}$ and for $a_{\mathrm{i}+13}$. Only new utterly destroyed units for $a_{\mathrm{i}+11}$ and for $a_{\mathrm{i}+12}$, $a_{\mathrm{i}+13}$ are the final new utterly destroyed units (for this new added layer), but we calculate and remember utterly destroyed units for each of 9 kinds of long paths, respectively.

After we added the units a_{m1}, a_{m2}, a_{m3} and calculated all utterly destroyed units and all contradiction unit pairs, if some units are not utterly destroyed units, then there is at least a long path from the first layer to the mth layer. Otherwise, there is no such a path.

2.3 Algorithm 3: Calculating A Long Unit Path From The First Layer To The *M*th Layer

For 9 kinds of long paths, we try to calculate one. For any long path of 9 kinds, if we proved that it exists, we can calculate such a long path. Delete all utterly destroyed units for this long path. Then, each unit can be in some one long path. We first choose any one unit from the first layer and it is the root of the long unit path to calculate. We delete its brother units in the first layer. Then, from the second layer to the lowest layer, we delete all units that are destroyed by the unit we keep in the first layer. Then, we calculate all new contradiction unit pairs due to deleting the brother units by calling Algorithm 1. In each layer, there is at least one unit; we continue for the second layer. We choose one unit from the second layer and delete its brother units in this layer. We then delete all units destroyed by the unit that we keep in the second layer. Then, we calculate all new contradiction unit pairs due to deleting the brother units by calling Algorithm 1, continuing for the third layer, and so on. Finally, we can get a long unit path from the first layer to the lowest layer.

Theorem 1. *The algorithm can solve the 3SAT in $O(m^6)$.*

Proof. Suppose that the number of clauses is m. All units are $3m$. The main time is to calculate the new contradiction unit pairs. For each $O(m^2)$ unit pair, for each unit in three-unit layers, we have to calculate $O(m)$ times 2SAT as stated above. The time for each 2SAT is $O(m)$. The units in three-unit layers is $O(m)$. Thus, the time for each unit pair, that is, for Algorithm 1, is $O(m) * O(m) * O(m) = O(m^3)$. We have to do this for each of 9 kinds of long paths. The time for all pairs is $O(m^2) * O(m^3) * 9 = O(m^5)$. There are $O(m)$ new added units in turn. The time for the Algorithm 3 is the same. Thus, the entire time complexity is $O(m^5) * O(m) = O(m^6)$. □

ACKNOWLEDGMENTS

I would like to thank the editor-in-chief, the editors, and the reviewers of TOCT. They gave my paper a very patient and very good review. Thanks a lot. I would also like to thank my parents and grandparents. When I was very young, they gave me a lot of love and encouragement. This paper is dedicated to them.

REFERENCES

[1] S. A. Cook. 1971. The complexity of theorem proving procedures. In *Proceedings of 3rd Annual ACM Symposium on Theory of Computing*, ACM, New York2021, 151–158.
[2] S. Arora and B. Barak. 2009. *Complexity Theory: A Modern Approach.* Cambridge: Cambridge University Press2021.
[3] Lance Fortnow. 2010. The status of the P versus NP problem. *Communications of the ACM* 52, 9 (2010), 78–86.

Received December 2020; revised January 2021; accepted January 2021