A Formal and Computationally Illustrated Proof that SUCCINCT-TRIANGLE is NP-Complete

Alwin
University of Indonesia

***Abstract*—This paper gives a self-contained proof that SUCCINCT-TRIANGLE is NP-complete. A succinct graph is represented by a Boolean circuit that decides adjacency between two vertex labels. To avoid any hidden symmetry assumption on the input circuit, the undirected edge relation is defined by applying the circuit to the canonical lexicographic ordering of each unordered pair of vertices. Membership in NP follows because a triangle has a polynomial-size certificate consisting of three vertex labels. NP-hardness is proved by a polynomial-time many-one reduction from CIRCUIT-SAT. The reduction creates a triangle exactly when the original Boolean circuit has a satisfying assignment. The paper also includes computer-generated visualizations and finite brute-force checks of the reduction. These computations illustrate and validate the implementation of the construction on small instances, but the NP-completeness claim rests on the formal proof.**

***Index Terms*—NP-completeness, succinct graphs, Boolean circuits, Circuit-SAT, triangle, computational verification.**

I. INTRODUCTION

A succinct representation stores a large combinatorial object implicitly, typically by means of a Boolean circuit. For graphs, the number of represented vertices may be exponential in the length of the representation, while adjacency between two vertices can still be computed by evaluating a circuit. Such representations are standard in complexity theory and were studied, for example, by Galperin and Wigderson in their work on succinct graph representations [2].

This paper proves that the problem of deciding whether a succinctly represented graph contains a triangle is NP-complete. The important point is that the represented graph may be exponentially large, but a triangle can still be certified by only three vertex labels. Therefore, the problem remains in NP.

In addition to the proof, this paper includes a small computer-generated illustration of the reduction and brute-force checks on finite instances. These computations are not a substitute for the proof, since NP-completeness is an asymptotic statement over infinitely many input sizes. They are included to make the construction concrete and to check that the implemented reduction satisfies the equivalence

$$B \text{ is satisfiable} \iff G_{C_B} \text{ contains a triangle}$$

on the tested instances.

II. PROBLEM DEFINITION

We first define the succinct graph representation carefully.

Definition 1 (Succinct graph)**.** *Let*

$$C : \{0,1\}^{2m} \to \{0,1\}$$

be a Boolean circuit. The circuit takes two vertex labels

$$u, v \in \{0,1\}^m.$$

It represents an undirected graph

$$G_C = (V, E)$$

where

$$V = \{0,1\}^m.$$

For two distinct vertices u and v, define

$$\mathrm{lo}(u,v) = \min_{\mathrm{lex}}\{u,v\}$$

and

$$\mathrm{hi}(u,v) = \max_{\mathrm{lex}}\{u,v\}.$$

The edge relation is

$$\{u,v\} \in E \iff u \neq v \wedge C(\mathrm{lo}(u,v), \mathrm{hi}(u,v)) = 1.$$

Thus the input circuit is not required to be symmetric.

Definition 2 (SUCCINCT-TRIANGLE)**.** *The input is a Boolean circuit*

$$C : \{0,1\}^{2m} \to \{0,1\}.$$

The question is whether G_C contains three distinct vertices

$$a, b, c \in \{0,1\}^m$$

such that

$$\{a,b\}, \{b,c\}, \{a,c\} \in E.$$

Equivalently, the question is whether there exist distinct a, b, c such that

$$C(\mathrm{lo}(a,b), \mathrm{hi}(a,b)) = 1,$$

$$C(\mathrm{lo}(b,c), \mathrm{hi}(b,c)) = 1,$$

and

$$C(\mathrm{lo}(a,c), \mathrm{hi}(a,c)) = 1.$$

III. MAIN THEOREM

Theorem 1. SUCCINCT-TRIANGLE *is NP-complete.*

We prove the theorem in two parts. First, we show that SUCCINCT-TRIANGLE belongs to NP. Second, we prove NP-hardness by giving a polynomial-time many-one reduction from CIRCUIT-SAT, which is known to be NP-complete [1].

IV. Membership in NP

Lemma 1. Succinct-Triangle *belongs to NP.*

Proof. Let the input be a circuit

$$C : \{0,1\}^{2m} \to \{0,1\}.$$

The input size is the encoding size of the circuit, denoted by $|C|$. Since the circuit explicitly has $2m$ input wires, we have

$$m \le |C|$$

under any standard reasonable circuit encoding.

A certificate for a yes-instance is a triple of vertex labels

$$a, b, c \in \{0,1\}^m.$$

The certificate length is

$$|a| + |b| + |c| = 3m \le 3|C|,$$

which is polynomial in the input size.

The verifier first checks that

$$a \ne b, \qquad b \ne c, \qquad a \ne c.$$

It then computes the three canonical ordered pairs

$$(\mathrm{lo}(a,b), \mathrm{hi}(a,b)),$$

$$(\mathrm{lo}(b,c), \mathrm{hi}(b,c)),$$

and

$$(\mathrm{lo}(a,c), \mathrm{hi}(a,c)).$$

Finally, it evaluates C on these three pairs and accepts if all three outputs are 1.

Lexicographic comparison of two m-bit strings takes polynomial time in m, and evaluating a Boolean circuit takes polynomial time in the circuit size. Since the verifier evaluates the circuit only three times, the whole verification procedure runs in polynomial time in $|C|$. Therefore, Succinct-Triangle is in NP. □

V. NP-Hardness

Lemma 2. Succinct-Triangle *is NP-hard.*

Proof. We give a polynomial-time many-one reduction from Circuit-SAT to Succinct-Triangle.

Let the input instance of Circuit-SAT be an arbitrary Boolean circuit

$$B(x_1, x_2, \ldots, x_n).$$

We construct an adjacency circuit C_B such that

$$B \text{ is satisfiable} \iff G_{C_B} \text{ contains a triangle.}$$

A. Vertex Labels

Each vertex of G_{C_B} has a label of the form

$$(t, x),$$

where

$$t \in \{00, 01, 10, 11\}$$

and

$$x \in \{0,1\}^n.$$

The first component t is a two-bit type, and the second component x is an assignment to the variables of B. Hence each vertex label has length

$$m = n + 2.$$

B. The Adjacency Circuit

Define the type predicate $T(t,s)$ by

$$T(t,s) = 1$$

if and only if

$$\{t,s\} \in \big\{\{00,01\}, \{01,10\}, \{00,10\}\big\}.$$

Equivalently, $T(t,s) = 1$ exactly when t and s are two distinct elements of

$$\{00, 01, 10\}.$$

The type 11 is a dummy type.

Now define the circuit

$$C_B((t,x),(s,y))$$

by

$$C_B((t,x),(s,y)) = 1$$

if and only if all of the following hold:

$$x = y,$$

$$B(x) = 1,$$

and

$$T(t,s) = 1.$$

Figure 1 shows the circuit-level structure of this adjacency predicate.

This circuit can be constructed in polynomial time from B. The equality test $x = y$ has size $O(n)$, the test $B(x) = 1$ uses one copy of B, and the type predicate $T(t,s)$ has constant size. Since n is at most the input length of B up to a standard encoding constant, we have

$$|C_B| \le |B| + O(n) + O(1) = \mathrm{poly}(|B|).$$

Thus $B \mapsto C_B$ is a polynomial-time computable transformation.

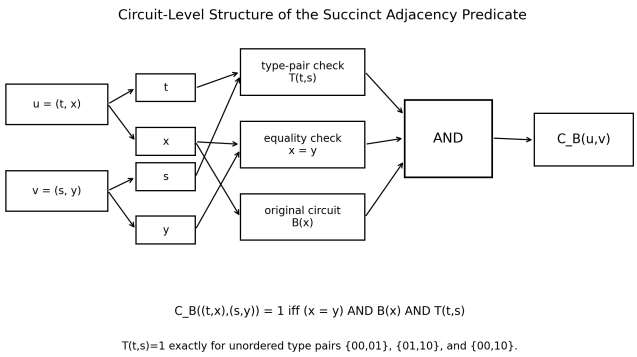

Fig. 1. Structure of the succinct adjacency circuit C_B. The circuit accepts a pair of vertex labels only when their assignment components are equal, the assignment satisfies B, and the two type components form one of the three allowed type pairs.

C. Correctness: Forward Direction

Assume that B is satisfiable. Then there exists an assignment

$$\alpha \in \{0,1\}^n$$

such that

$$B(\alpha) = 1.$$

Consider the three vertices

$$(00, \alpha), \qquad (01, \alpha), \qquad (10, \alpha).$$

They are distinct because their type components are distinct.

For every pair among these three vertices, the assignment component is the same, namely α; the circuit B outputs 1 on α; and the pair of types is one of

$$\{00, 01\}, \qquad \{01, 10\}, \qquad \{00, 10\}.$$

Therefore the three vertices are pairwise adjacent in G_{C_B}. Hence G_{C_B} contains a triangle.

Thus

$$B \text{ is satisfiable} \implies G_{C_B} \text{ contains a triangle.}$$

D. Correctness: Reverse Direction

Conversely, assume that G_{C_B} contains a triangle. Then there exist three distinct vertices

$$(t_1, x_1), \qquad (t_2, x_2), \qquad (t_3, x_3)$$

that are pairwise adjacent.

Because the first and second vertices are adjacent, the definition of C_B implies

$$x_1 = x_2.$$

Because the second and third vertices are adjacent, the definition of C_B implies

$$x_2 = x_3.$$

Hence

$$x_1 = x_2 = x_3.$$

Let this common assignment be α.

Furthermore, every edge produced by the construction exists only if the associated assignment makes B output 1. From any one of the three edges in the triangle, we therefore obtain

$$B(\alpha) = 1.$$

Thus B has a satisfying assignment, so B is satisfiable.

Therefore

$$G_{C_B} \text{ contains a triangle} \implies B \text{ is satisfiable.}$$

E. Conclusion of the Reduction

Combining both directions gives

$$B \text{ is satisfiable} \iff G_{C_B} \text{ contains a triangle.}$$

Since $B \mapsto C_B$ is computable in polynomial time, SUCCINCT-TRIANGLE is NP-hard. □

VI. Concrete Example of the Reduction

Consider the Boolean circuit

$$B(x_0, x_1) = x_0 \wedge \neg x_1.$$

The satisfying assignments are exactly

$$x = 10.$$

Since $n = 2$, the constructed graph has

$$4 \cdot 2^n = 16$$

vertices, one for each pair (t, x) with $t \in \{00, 01, 10, 11\}$ and $x \in \{0,1\}^2$. The construction produces edges only in the assignment block $x = 10$, and exactly among the three type labels 00, 01, and 10. Thus the graph contains exactly one triangle:

$$(00, 10), \qquad (01, 10), \qquad (10, 10).$$

The explicit graph is shown in Fig. 2.

VII. Computational Verification

The formal proof above is sufficient for NP-completeness. Nevertheless, the reduction was also checked computationally on small finite instances. The purpose of this section is not to prove NP-completeness by experiment, but to verify that the implemented construction agrees with the mathematical statement.

For a Boolean function $B : \{0,1\}^n \to \{0,1\}$, the program explicitly constructed G_{C_B} with

$$4 \cdot 2^n$$

vertices. It then enumerated all unordered triples of vertices and tested whether any triple forms a triangle. The result was compared with the direct satisfiability test

$$\exists x \in \{0,1\}^n \; B(x) = 1.$$

For exhaustive tests, B was represented by its truth table, allowing all Boolean functions on n variables to be enumerated.

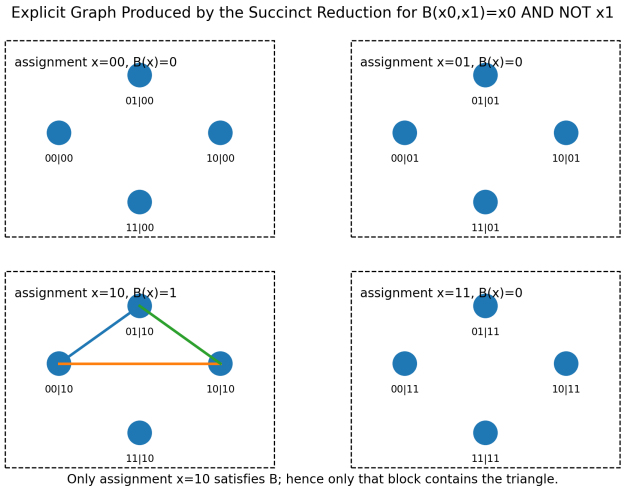

Fig. 2. Explicit small graph produced by the reduction for $B(x_0, x_1) = x_0 \wedge \neg x_1$. Each block corresponds to an assignment x. Only $x = 10$ satisfies B, so only that block contains the triangle on types 00, 01, and 10.

TABLE I
FINITE COMPUTATIONAL CHECKS OF THE EQUIVALENCE $SAT(B) \iff Triangle(G_{C_B})$.

Test set	Cases	Matches	Mismatches
Exhaustive $n = 2$	16	16	0
Exhaustive $n = 3$	256	256	0
Random $n = 4$	500	500	0

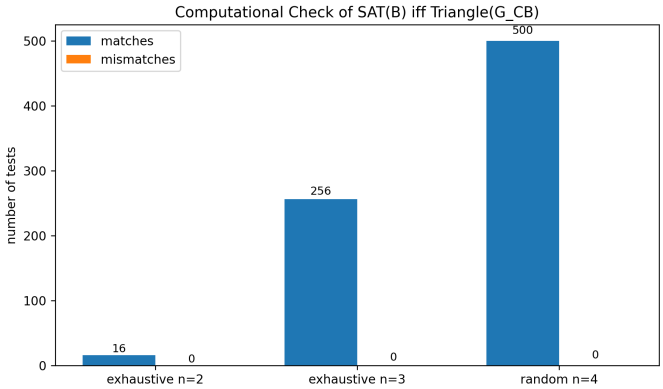

Fig. 3. Computer-generated validation of the equivalence $SAT(B) \iff Triangle(G_{C_B})$ on finite instances. The exhaustive cases cover all Boolean functions for $n = 2$ and $n = 3$; the $n = 4$ cases are random truth-table functions.

Table I gives the results. For $n = 2$, all

$$2^{2^2} = 16$$

Boolean functions were tested. For $n = 3$, all

$$2^{2^3} = 256$$

Boolean functions were tested. For $n = 4$, 500 random truth-table functions were tested. No mismatch was found. Figure 3 visualizes the same result.

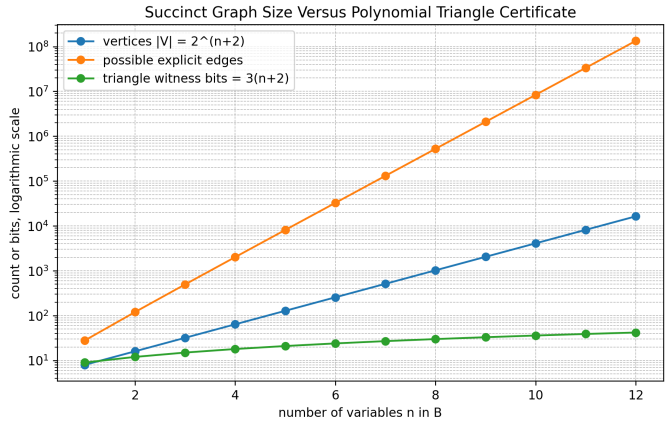

Fig. 4. The explicit graph size grows exponentially in n, while the triangle witness remains linear in the label length. This explains why the graph is succinct but SUCCINCT-TRIANGLE remains in NP.

The example in Fig. 2 was also checked by explicit enumeration. The computed values were

$$|V| = 16,$$

$$|E| = 3,$$

and

$$\#\text{triangles} = 1,$$

with the unique triangle

$$(00|10), \quad (01|10), \quad (10|10).$$

These values agree exactly with the reduction.

VIII. SIZE GROWTH AND CERTIFICATE LENGTH

Although the represented graph G_C has

$$|V| = 2^m$$

vertices, a yes-certificate for a triangle does not need to describe the whole graph. It only needs to list three vertex labels

$$a, b, c \in \{0, 1\}^m.$$

The witness length is therefore $3m$, which is polynomial in the size of the input circuit.

For the reduction above, $m = n + 2$, so the graph has

$$2^{n+2}$$

vertices, while the triangle certificate has length

$$3(n + 2).$$

Figure 4 compares the exponential growth of the explicit graph with the linear growth of the triangle certificate length.

This explains why SUCCINCT-TRIANGLE is an NP-complete succinct-graph problem rather than a problem that jumps to nondeterministic exponential time. More formally, since SUCCINCT-TRIANGLE is in NP, and NP is a proper subset of NEXP by the nondeterministic time hierarchy theorem, SUCCINCT-TRIANGLE is not NEXP-complete under polynomial-time many-one reductions.

IX. Conclusion

We have proved that SUCCINCT-TRIANGLE is in NP and is NP-hard. Membership in NP follows from the polynomial-size certificate consisting of three vertex labels. NP-hardness follows from a polynomial-time reduction from CIRCUIT-SAT that creates a triangle exactly when the original circuit has a satisfying assignment. The computational visualizations and brute-force checks support the construction on finite examples, while the formal proof establishes the asymptotic complexity result. Therefore,

$$\boxed{\text{SUCCINCT-TRIANGLE is NP-complete.}}$$

References

[1] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, 1971, pp. 151–158.

[2] H. Galperin and A. Wigderson, "Succinct representations of graphs," *Information and Control*, vol. 56, no. 3, pp. 183–198, Mar. 1983.