On limitations of transformations between combinatorial problems

William Slough[†] and Karl Winklmann[‡]

CU-CS-336-86 July 1986

[†] Department of Computer Science, University of Missouri, Columbia, Missouri 65211. Work of this author was supported in part by a Chevron Fellowship while he was at Washington State University. Some of the results reported in this paper were obtained while this author was completing his PhD thesis under the direction of the second author at Washington State University.

[‡] Department of Computer Science, University of Colorado, Boulder, Colorado 80309. Work of this author was supported in part by National Science Foundation Grants MCS-80004128, DCR-8202964, and DCR-8500741 and by Grant A-0369 of the Natural Sciences and Engineering Research Council of Canada and was conducted partly at Washington State University and at the University of Alberta.

ABSTRACT

We define a class of "local transformations" that includes many transformations from the NP-completeness literature. We then prove (without assuming P≠NP) that this type of transformation is too weak to transform 3SAT or a number of other NP-complete problems into 2SAT or a number of other problems in P. The proof uses concepts related to distributed computing.

1. INTRODUCTION AND OVERVIEW

Let Π be an NP-complete problem and let Π' be a problem in P. Then P$\neq$NP is equivalent to the statement that there is no polynomial-time reduction, in the sense of Cook (1971), of Π to Π'. While a proof of P$\neq$NP, i.e., a proof of nonexistence of polynomial-time reductions of Π to Π' for such a pair of problems Π and Π', seems to be beyond the reach of current techniques, nonexistence results for interesting *subclasses* of polynomial-time reductions can be proven, as we will show.

We formally define a class of "local transformations" and then prove that 3SAT, VERTEX COVER, and a number of other NP-complete problems cannot be "locally transformed" into 2SAT or a number of other problems in P. (For definitions of these and other problems we refer the reader to Garey and Johnson (1979).) This class of "local transformations" includes many transformations from the NP-completeness literature, especially transformations that fall into the intuitive categories of "local-replacement transformations" and "component-design transformations".

This result is proven by showing that these transformations preserve what we call "surface complexity". This "surface complexity" is closely related to the "communication" complexities of Yao (1979) and of Papadimitriou and Sipser (1984). Thus, the proof of our main result demonstrates that concepts from the theory of distributed computing can be used to obtain results in the theory of NP-completeness, an observation which partly motivates this work.

The rest of this paper is organized as follows. Section 2 defines "local transformations". Section 3 proves the main result for this fairly simple class of transformations. Section 4 extends the class of transformations in ways which capture substantially more transformations but do not spoil the proof of the main result.

2. Local Transformations: Definitions

2.1. Motivation

Many of the transformations used in NP-completeness proofs in the literature are of a "local" flavor: they can be carried out without ever looking at the whole problem instance that is to be transformed; instead it suffices to "slide a window" over the instance and to transform whatever "local" piece is visible in the window at any moment. This "local" nature is especially prominent in those transformations that have been called "local-replacement transformations" or "component-design transformations". The following informal characterization of local-replacement transformations is from Garey and Johnson (1979):

> "All we do is pick some aspect of the known NP-complete problem instance to make up a collection of *basic units*, and we obtain the corresponding instance of the target problem by replacing each basic unit, in a uniform way, with a different structure." (Garey and Johnson (1979), p. 66, original italics.)

The transformations we consider apply to those combinatorial problems which involve graphs, sets, and Boolean expressions. In the rest of this section we give a formal definition of a class of "local transformations". This class includes only a handful of transformations from the NP-completeness literature, but is extended in Section 4 to include many more. This two-stage approach allows the essentials of the proof of the main result to be presented without an excessive amount of technical clutter. Afterwards, it will not be difficult to argue that the class can be extended in ways which do not spoil the proof.

The "basic units" mentioned above are vertex-induced subgraphs of the graph to be transformed. An especially simple example is the transformation of VERTEX COVER to FEEDBACK ARC SET of Karp (1972), which is illustrated in Figure 1. In this example the "basic units" are the edges of the original graph. Figure 2 shows a "local-replacement rule" that defines this transformation.[1] Figure 3 provides another example.

Any set Λ of such "local-replacement" rules defines a transformation τ_Λ on graphs. A graph G is transformed by applying each rule $L \rightarrow R$ of Λ to every vertex-induced subgraph of G which is isomorphic to the left-hand side, L, of the rule. All of this is made precise below.

A (*finite, directed*) *graph* G is a pair $<V, A>$ where V is a finite set and A is a binary relation on V. Elements of V are called *vertices* of G and elements of A are called *arcs* of G. We use V_G to denote the set of vertices of a graph G and A_G to denote its set of arcs. The *union* $G \cup H$ of two graphs G and H is the graph $<V_G \cup V_H, A_G \cup A_H>$. Two graphs G_1 and G_2 are *isomorphic*, $G_1 \cong G_2$, if there is a bijection $\psi : V_{G_1} \rightarrow V_{G_2}$ with $A_{G_2} = \{<\psi(u), \psi(v)> : <u,v> \in A_{G_1}\}$. Such a bijection ψ is called an *isomorphism from* G_1 *to* G_2. An *automorphism on* a graph G is an isomorphism from G to itself. A *path from* a vertex u *to* a vertex v *in* a graph G is a sequence $v_1, a_1, v_2, a_2, \cdots, a_k, v_{k+1}$ with $u = v_1$, $v = v_{k+1}$, $v_i \in V_G$ for $1 \leq i \leq k+1$, and $a_i \in A_G \cap \{<v_i, v_{i+1}>, <v_{i+1}, v_i>\}$ for $1 \leq i \leq k$. The *length of* such a path is k.

[1] Strictly speaking, the transformation defined by this rule differs slightly from the one given in Karp (1972) by ignoring isolated vertices. A second rule, with a single vertex as left-hand side, would make the two transformations agree in their treatment of isolated vertices. For the correctness of the transformation, this is irrelevant.

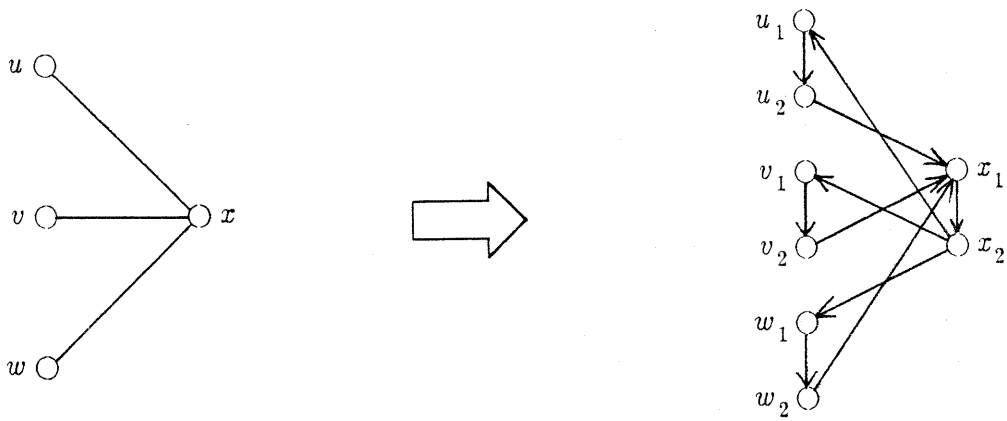

Figure 1.
Transforming VERTEX COVER to FEEDBACK ARC SET (Karp 1972).

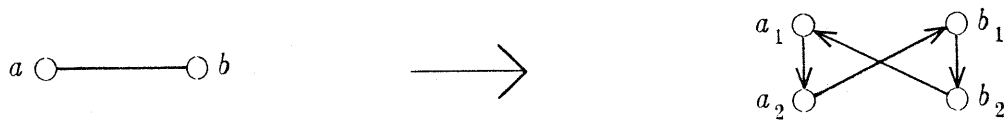

Figure 2.
A local-replacement rule for transforming VERTEX COVER to FEEDBACK ARC SET.

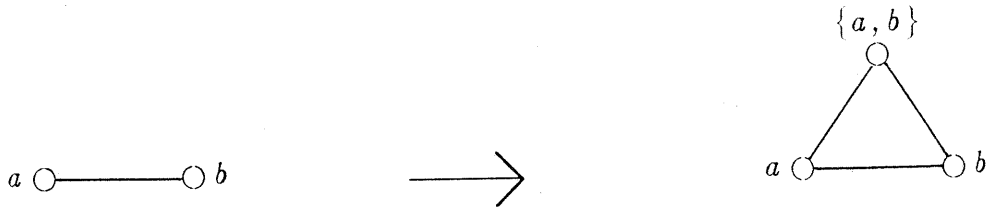

Figure 3.
A local-replacement rule for transforming VERTEX COVER to DOMINATING SET.

Note that the direction of an arc is of no concern in this definition of a path. The *distance between* two vertices of a graph is the length of a shortest path between the vertices. If no path exists the distance is regarded as infinite. The *diameter of* a graph J is defined to be the largest distance between any two vertices of J. The diameter of a disconnected graph J is infinite.

The set V_{comp} of *composite labels over* an alphabet V is built up inductively from V by forming sets. Formally, V_{comp} is the minimal set satisfying

(1) $V \subseteq V_{comp}$; and

(2) if $S \subseteq V_{comp}$ then $S \in V_{comp}$.

Since sets can be used to encode other mathematical constructs, we may pretend that, e.g., if $V=\{a,b,c,\cdots\}$ then V_{comp} contains not only a, $\{a\}$, $\{a,c\}$, $\varnothing$, $\{\varnothing\}$, $\{\{a,b,c\},\varnothing\}$, $\cdots$, but also a_1, $<a,b>$, 0, 1, $a \vee b \vee \bar{c}$, and any other type of standard or non-standard mathematical construct and notation that we might find useful. The function $components: V_{comp} \longrightarrow 2^V$ is defined inductively by

(1) $components(v) = \{v\}$ for all $v \in V$; and

(2) $components(S) = \bigcup_{s \in S} components(s)$ for all $S \in V_{comp} - V$.

(To prevent confusion, we will avoid letting a and a_1 denote two elements from V, since this would make a_1 denote two different objects: the element of V that is denoted by a_1 as well as the element of $V_{comp} - V$ that is obtained by putting a subscript 1 on the element of V that is denoted by a.)

Any function ψ from an alphabet V to an alphabet V' naturally induces a function $\psi_{comp}: V_{comp} \longrightarrow V'_{comp}$ defined by

(1) $\psi_{comp}(v) = \psi(v)$ for all $v \in V$; and

(2) $\psi_{comp}(S) = \{\psi_{comp}(s) : s \in S\}$ for all $S \in V_{comp} - V$.

A *local-replacement rule* is a pair $L \longrightarrow R$ of graphs such that L is connected and $V_R \subseteq (V_L)_{comp}$. A local-replacement rule $L \longrightarrow R$ is *applicable to* a graph J if $L \cong J$. The *result of applying* a local-replacement rule $L \longrightarrow R$ *to* a graph J *with respect to* an isomorphism ψ from L *to* J is the graph K defined by

$$V_K = \{\psi_{comp}(v) : v \in V_R\}$$

and

$$A_K = \{<\psi_{comp}(u), \psi_{comp}(v)> : <u,v> \in A_R\}.$$

Since we want transformations to be well-defined functions, the result of the application of a rule $L \longrightarrow R$ to a graph J should be independent of the choice of isomorphism ψ from L to J with respect to which the rule is applied. Local-replacement rules that have this property are called *deterministic.* The following observation characterizes them.

OBSERVATION 1 (Slough 1984). *A local-replacement rule $L \longrightarrow R$ is deterministic if and only if for every automorphism ψ on L, ψ_{comp} is an automorphism on R.*

A set Λ of local-replacement rules is *deterministic* if each rule in Λ is deterministic and no two rules in Λ have isomorphic left-hand sides. The

transformation induced by a deterministic set Λ of local-replacement rules is the function τ_Λ defined by

$$\tau_\Lambda(G) = \bigcup_J r_\Lambda(J)$$

where the union is taken over all vertex-induced subgraphs J of G and where $r_\Lambda(J)$, for any graph J, denotes the (unique) result of applying a rule of Λ to J. (Since Λ is deterministic, at most one rule of Λ can be applicable to any given graph J. If no rule of Λ is applicable to J we let $r_\Lambda(J)$ be the empty graph.)

OBSERVATION 2. *Let G be a graph, let Λ be a deterministic set of local-replacement rules, and let v be a vertex of $\tau_\Lambda(G)$. Then components$(v) \subseteq V_G$.*

A transformation τ of one graph problem into another is a *local transformation* if $\tau = \tau_\Lambda$ for some finite deterministic set Λ of local-replacement rules with the property that the left-hand sides of all rules are connected graphs. (This connectivity gives τ_Λ its local character.) We write $\Pi \leq_{local} \Pi'$ to indicate that a local transformation of Π into Π' exists.

OBSERVATION 3 (Slough 1984). *Let G and G' be two isomorphic graphs, let $\psi: V_G \rightarrow V_{G'}$ be an isomorphism from G to G', and let τ be a local transformation. Then ψ_{comp}, restricted to $V_{\tau(G)}$, is an isomorphism from $\tau(G)$ to $\tau(G')$.*

In various graph problems, an instance of the problem consists of a graph and an integer, as in VERTEX COVER and GRAPH k-COLORABILITY, for example. We extend our notion of local transformations to such problems by insisting that the "graph portion" of the transformation be a local transformation in the

sense defined so far and by allowing the integer in the transformed instance to be determined by any function of the original integer (if present) and of the number of vertices and the number of edges in the original graph.

2.3. Local transformations involving set problems

3-DIMENSIONAL MATCHING (3DM) is one of the six "basic" NP-complete problems considered in Garey and Johnson (1979) and is frequently used to prove NP-completeness results. The class of local transformations, as defined in Section 2.2, does not include transformations involving 3DM for the superficial reason that it is not a graph problem. However, an instance of 3DM, which consists of three sets W, X, and Y of equal cardinality and a set $M \subseteq W \times X \times Y$, may be viewed as a graph problem, not an uncommon view (see Dyer and Frieze (1986), for example). Given such a representation, the definition of local transformations between graph problems apply directly. To be specific, let us choose the representation illustrated in Figure 4. For this to work, it is necessary to make a distinction between vertices which represent elements from each of W, X, Y, and M. This could be achieved by "tagging" the four types of vertices in different ways with additional vertices and edges. Instead, we allow different *types* of vertices, shown in Figure 4 as circles, triangles, diamonds, and boxes. Naturally, isomorphisms, which govern the application of local-replacement rules, must not map vertices of one type to vertices of another.

Instances of other kinds of set problems can be given a graph representation in an obvious way. For example, EXACT COVER BY 3-SETS (X3C), a generalization of 3DM, has a graph representation similar to that just described with the exception that there are just two types of vertices, boxes and

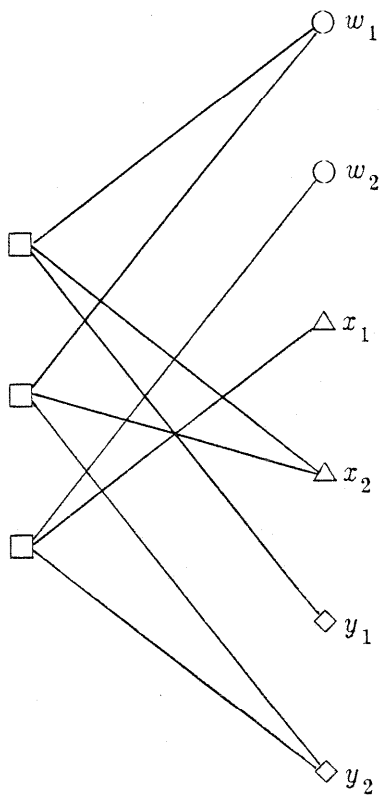

Figure 4.

A graph representation for an instance of 3DM where $W=\{w_1, w_2\}$, $X=\{x_1, x_2\}$, $Y=\{y_1, y_2\}$, and $M=\{(w_1, x_2, y_1), (w_1, x_2, y_2), (w_2, x_1, y_2)\}$.

circles, corresponding to sets and elements, since there is no distinction between the elements to be covered.

The transformation from X3C to PARTITION INTO TRIANGLES, as described in Garey and Johnson (1979), pp. 68-69, is a prototypical example of a local replacement transformation. The basic units involved in the transformation are the 3-element subsets in an X3C instance. This transformation may be described using a single local-replacement rule, shown in Figure 5.

As was noted in Section 2.2, on occasion a transformation must produce an integer in the transformed instance. When this is necessary, a local transformation involving set problems is free to use any function of the original integer (if present) and the numbers of vertices of the different types and the number of edges in the original instance.

2.4. Local transformations involving Boolean expressions

Transformations of 3SAT into other combinatorial problems typically are of the "component-design" variety: they construct a "truth-setting" component for each variable and a "satisfaction-testing" component for each clause. As an example, consider the transformation of 3SAT into VERTEX COVER from Garey and Johnson (1979), pp. 54-56, illustrated in Figure 6. (This is not the same transformation as the one in Karp (1972) via the CLIQUE problem.) In this transformation, each variable x of a Boolean expression E gets transformed into two adjacent vertices labeled x and $\bar{x}$, and each clause $C=(l_1 \vee l_2 \vee l_3)$ gets transformed into a 3-clique of vertices labeled l_1^C, l_2^C, and l_3^C. Finally, for every variable x and every clause C, the vertex labeled x gets connected with every vertex labeled x^C, and similarly the vertex labeled $\bar{x}$ gets connected with every vertex labeled $\bar{x}^C$. The Boolean expression E is satisfiable

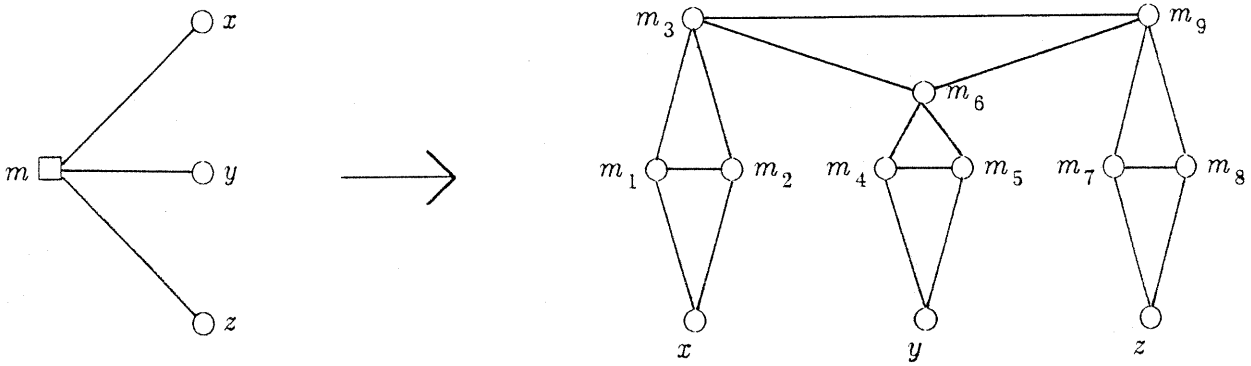

Figure 5.

A local replacement rule for transforming X3C to PARTITION INTO TRIANGLES.

$C_1 \wedge C_2 \wedge C_3 =$
$(x_1 \vee x_2 \vee \bar{x}_3) \wedge$
$(\bar{x}_1 \vee \bar{x}_3 \vee x_4) \wedge$
$(\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4)$

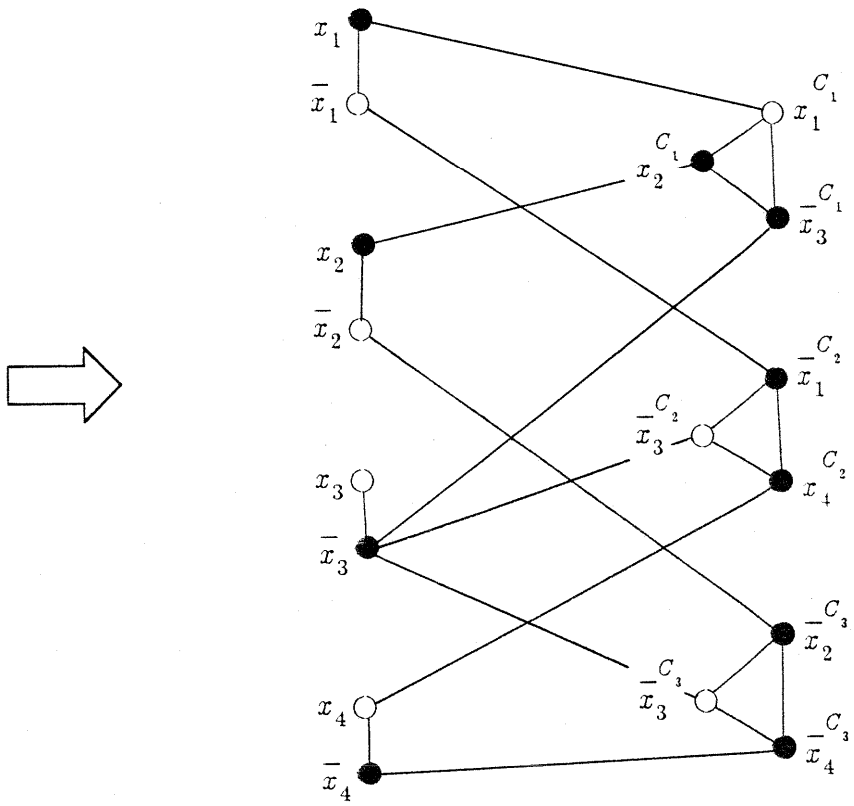

Figure 6.

Transforming 3SAT to VERTEX COVER (Garey and Johnson 1979). The dark vertices form a vertex cover that corresponds to the satisfying assignment $(x_1, x_2, x_3, x_4) = (true, true, false, false)$.

if and only if the graph thus constructed has a vertex cover whose size is the number of variables in E plus twice the number of clauses. Choosing to make variable x of E *true* corresponds to choosing the vertex labeled x to belong to the vertex cover of the graph.

In a fashion analogous to the graph representation of set problems discussed above, we choose some straightforward representation of Boolean expressions in conjunctive normal form as graphs, illustrated in Figure 7. It is convenient to use different types of vertices to represent variables and clauses, and different types of edges to indicate whether a variable occurs negated or nonnegated in a clause. Naturally, isomorphisms, which govern the application of local-replacement rules, must preserve types of edges as well as types of vertices. With this graph representation of Boolean expressions in conjunctive normal form, the transformation of 3SAT to VERTEX COVER that was illustrated in Figure 6 can be defined by a set of four local-replacement rules, which is shown in Figure 8. (We are assuming that each clause has exactly three literals and that no variable appears twice in a single clause. The four rules handle the cases of 0, 1, 2, and 3 negations in a clause.)

If it is necessary for a local transformation to produce an integer in the transformed instance, it is free to use any function of the original integer (if present) and the numbers of vertices of each type and the numbers of edges of each type in the original graph.

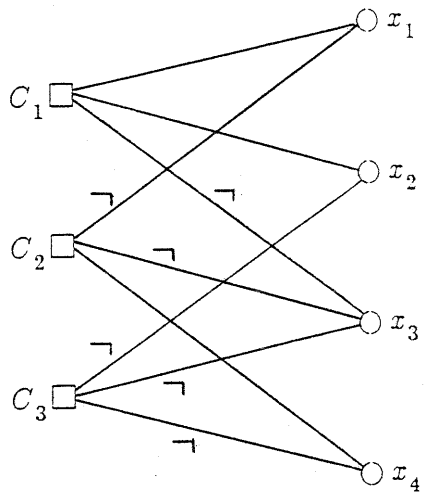

Figure 7.

A graph representation of the Boolean expression $C_1 \wedge C_2 \wedge C_3 = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4)$.

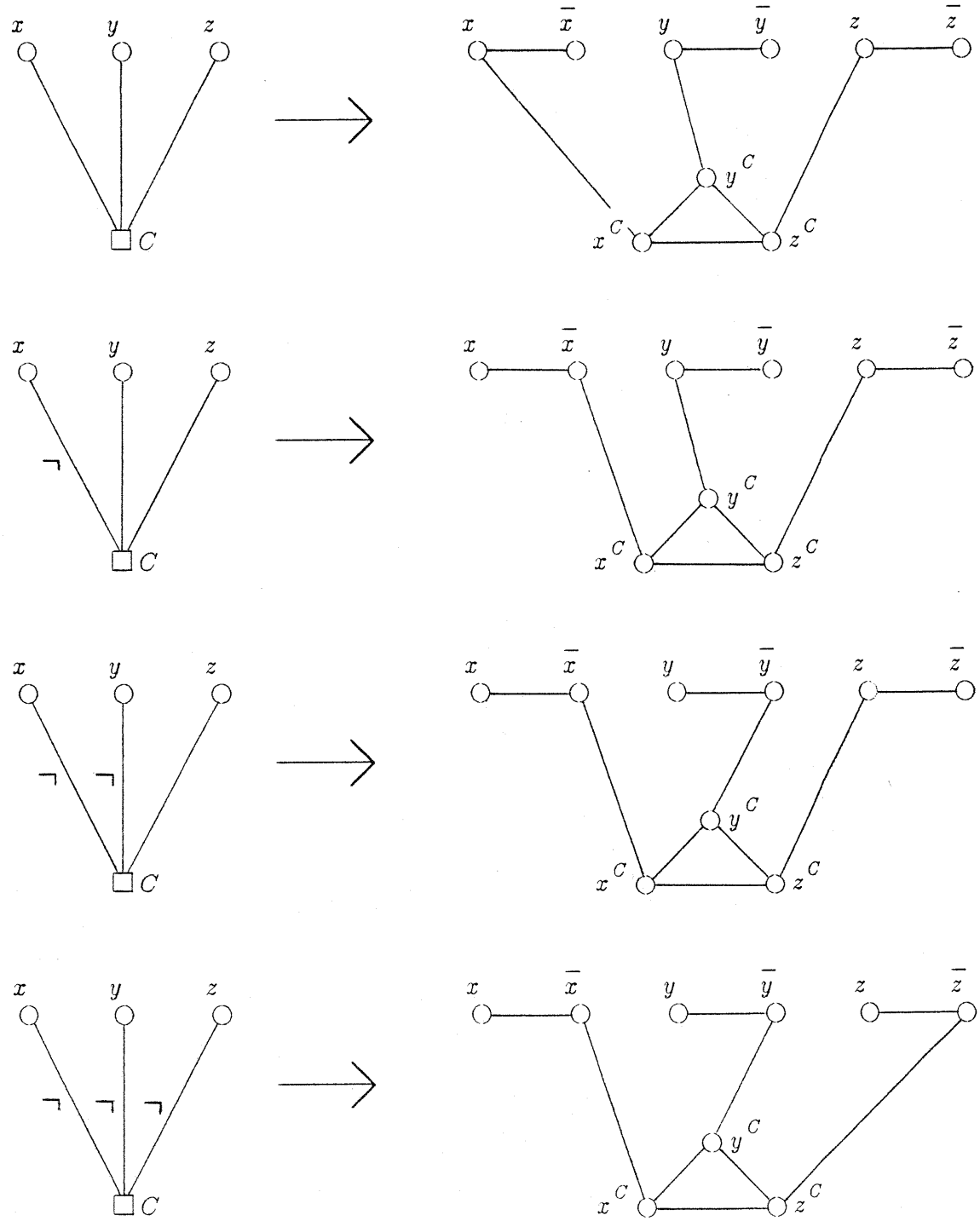

Figure 8.
Local-replacement rules for transforming 3SAT to VERTEX COVER.

3. LOCAL TRANSFORMATIONS: RESULTS

3.1. Surface complexity

Let Π be a set of graphs and let S be a set of vertices. We call two graphs G and G' *S-equivalent (with respect to* Π*)*, $G \equiv_{\Pi}^{S} G'$, if for all graphs H with $V_G \cap V_H \subseteq S$ and $V_{G'} \cap V_H \subseteq S$,

$$G \cup H \in \Pi \text{ iff } G' \cup H \in \Pi.$$

A graph H is said to *demonstrate the inequivalence* $G \not\equiv_{\Pi}^{S} G'$ if $V_G \cap V_H \subseteq S$ and $V_{G'} \cap V_H \subseteq S$ and exactly one of $G \cup H$ and $G' \cup H$ is in Π. Figure 9 provides an illustration. If the instances of Π include an integer, then two graphs G and G' are *S-equivalent (with respect to* Π*)*, $G \equiv_{\Pi}^{S} G'$, if for all graphs H with $V_G \cap V_H \subseteq S$ and $V_{G'} \cap V_H \subseteq S$ and all integers k,

$$\langle G \cup H, k \rangle \in \Pi \text{ iff } \langle G' \cup H, k \rangle \in \Pi.$$

With this type of graph problem, a graph H and integer k are said to *demonstrate the inequivalence* $G \not\equiv_{\Pi}^{S} G'$ if $V_G \cap V_H \subseteq S$ and $V_{G'} \cap V_H \subseteq S$ and exactly one of $\langle G \cup H, k \rangle$ and $\langle G' \cup H, k \rangle$ is in Π. For either type of graph problem Π, the *surface complexity* $\sigma_{\Pi}(s)$ *of* Π is the binary logarithm of the number of equivalence classes induced by the relation $\equiv_{\Pi}^{S}$, where S is any fixed set of s vertices. This also defines surface complexity for any problem for which a graph representation has been defined, as was done for families of sets and for Boolean expressions in conjunctive normal form in Section 2. (There are problems Π for which the relation $\equiv_{\Pi}^{S}$ induces infinitely many equivalence classes. One such problem is GRAPH AUTOMORPHISM. For all of the specific problems

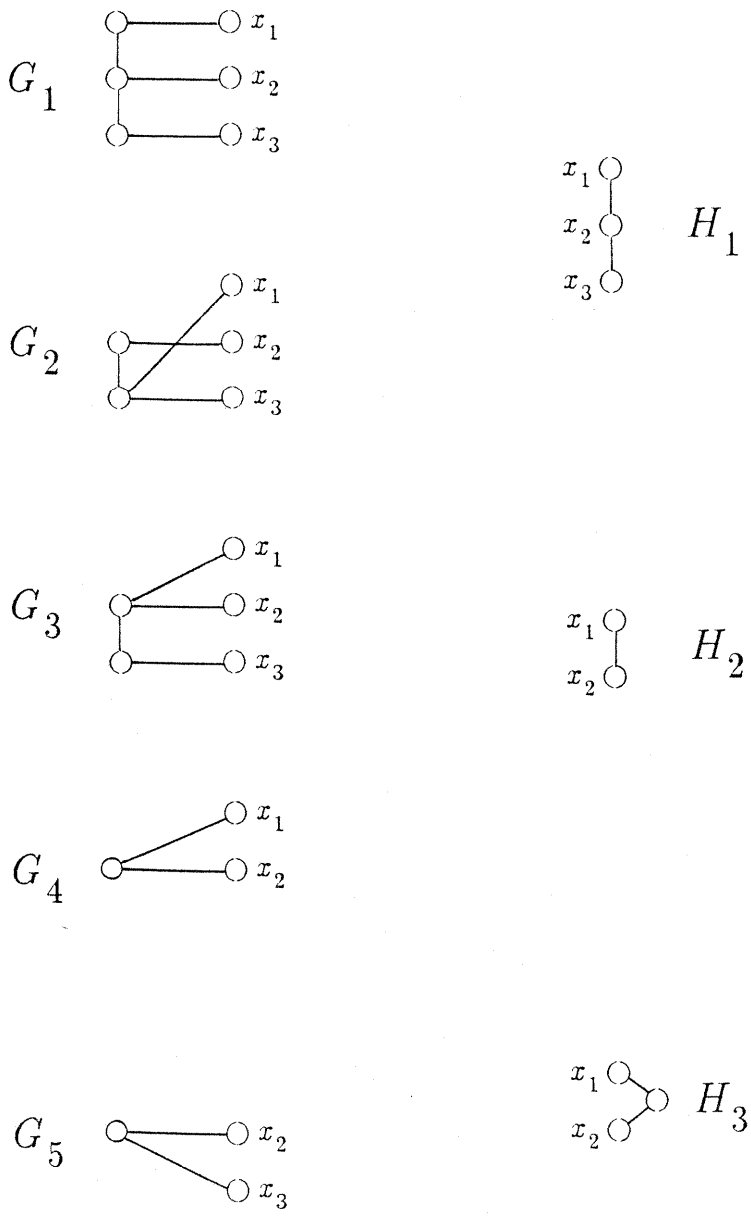

Figure 9.

Illustrating the definition of $\equiv_{||}^{S}$ for Π = GRAPH 2-COLORABILITY and $S = \{x_1, x_2, x_3\}$. The only equivalence among G_1 through G_5 is $G_1 \equiv_{||}^{S} G_2$. H_1 demonstrates the inequivalences $G_1 \not\equiv_{||}^{S} G_3$, $G_1 \not\equiv_{||}^{S} G_4$, $G_1 \not\equiv_{||}^{S} G_5$, $G_2 \not\equiv_{||}^{S} G_3$, $G_2 \not\equiv_{||}^{S} G_4$, and $G_2 \not\equiv_{||}^{S} G_5$. H_2 demonstrates the inequivalences $G_1 \not\equiv_{||}^{S} G_3$, $G_1 \not\equiv_{||}^{S} G_4$, $G_2 \not\equiv_{||}^{S} G_3$, $G_2 \not\equiv_{||}^{S} G_4$, $G_3 \not\equiv_{||}^{S} G_5$, and $G_4 \not\equiv_{||}^{S} G_5$. H_3 demonstrates the same inequivalences among G_1 through G_5 as H_1.

Π mentioned in our results, the number of equivalence classes of $\equiv_{\Pi}^{S}$ and hence $\sigma_{\Pi}(s)$ are always finite.)

Surface complexity can be given the following computational interpretation, which shows its connections to ideas from the theory of distributed computing. Assume that a pipeline of two processes, process A followed by process B, solves problem Π for graphs which have a vertex cutset that is a subset of $\{v_1, \cdots, v_s\}$ and assume that the graphs are cut along this cutset into two graphs, G and H, with process A receiving (an encoding of) G as input and process B receiving (an encoding of) H (as well as an integer if the instances of Π involve integers). Figure 10 provides an illustration. Process A *cannot* afford to send the same message to process B for two inequivalent graphs G and G' because B might receive on its other input line a graph H which demonstrates the inequivalence of G and G'. Conversely, process A *can* afford to send the same message to B for any two equivalent graphs G and G'. Thus, an optimal process A, i.e., one that minimizes the number of bits sent in a worst case, would tell B which equivalence class G belongs to, but nothing more. Hence, the number of bits that process A has to pass on to process B, in a worst-case instance with cutset $\{v_1, \cdots, v_s\}$, is the surface complexity $\sigma_{\Pi}(s)$, rounded up to the nearest integer. Thus, our surface complexity σ_{Π} is related to the "deterministic one-way" complexity of Yao (1979) and the one-way "communication complexity" of Papadimitriou and Sipser (1984). A major difference between σ_{Π} and both of these "one-way complexities" is that σ_{Π} is a function of *surface* size, not of *problem* size. This is crucial because only when surface complexity is defined as a function of surface size do there exist exponential differences such as the ones illustrated in Lemmas 2 and 3 below, which will allow us to prove limitations of local transformations. Lakshmipathy and Winklmann (to appear) investigate a complexity measure that is essentially

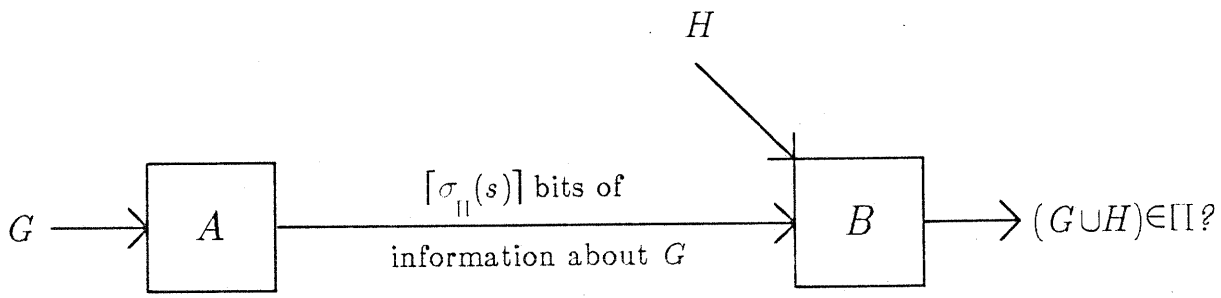

Figure 10.
A computational interpretation of $\sigma_{II}(s)$.

a two-way version of σ_Π.

LEMMA 1. *For all graph problems* Π *and all* $s \geq 0$, $\sigma_\Pi(s+1) \geq \sigma_\Pi(s)$.

PROOF. $G \not\equiv_\Pi^S G'$ always implies $G \not\equiv_\Pi^{S \cup \{v\}} G'$ because any H which demonstrates the inequivalence $G \not\equiv_\Pi^S G'$ also demonstrates the inequivalence $G \not\equiv_\Pi^{S \cup \{v\}} G'$. (This can easily be verified from the definition of $\equiv_\Pi^S$: $V_G \cap V_H \subseteq S$ and $V_{G'} \cap V_H \subseteq S$ trivially imply $V_G \cap V_H \subseteq S \cup \{v\}$ and $V_{G'} \cap V_H \subseteq S \cup \{v\}$ and the statement $G \cup H \in \Pi$ *iff* $G' \cup H \in \Pi$ does not involve S.) Hence the number of equivalence classes of $\equiv_\Pi^S$ cannot be smaller than the number of equivalence classes of $\equiv_\Pi^{S \cup \{v\}}$. □

In the following, 2-COLOR is an abbreviation for GRAPH 2-COLORABILITY.

LEMMA 2. $\sigma_{\text{2-COLOR}}(s) = O(s \log s)$.

PROOF. Consider the computational interpretation of $\sigma_\Pi(s)$ given above. How many bits of information are sufficient to determine the equivalence class of $\equiv_{\text{2-COLOR}}^S$ to which a graph G belongs? It is sufficient to know if G is 2-colorable and, if so, which pairs of vertices in $S \cap V_G$ are connected in G by paths of even lengths, which ones by paths of odd lengths, and which ones by no paths at all. If S, and hence $S \cap V_G$, contains at most s vertices, this information about paths can be packed into $O(s \log s)$ bits in the following fashion. Being connected via an even-length path is an equivalence relation on $S \cap V_G$ whose equivalence classes can be listed using $O(s \log s)$ bits. Since vertices from any one of these equivalence classes can be connected via an odd-length path to vertices in at most one other class, another $O(s \log s)$ bits suffice to indicate which classes are thus connected. The lemma follows. □

LEMMA 3. $\sigma_{\text{3SAT}}(s) = 2^s$.

PROOF. For any Boolean expression E let V_E denote the set of variables occurring in E and define an assignment $\alpha_S : S \longrightarrow \{true, false\}$ to be *acceptable to* E if there is an assignment $\alpha_I : V_E - S \longrightarrow \{true, false\}$ (I for "interior") such that $\alpha_{S \cup I} : V_E \longrightarrow \{true, false\}$ defined by

$$\alpha_{S \cup I}(v) = \begin{cases} \alpha_S(v) \text{ if } v \in V_E \cap S, \\ \alpha_I(v) \text{ if } v \in V_E - S \end{cases}$$

satisfies E. Then two Boolean expressions E and E' are S-equivalent (with respect to satisfiability) if for all $\alpha_S : S \longrightarrow \{true, false\}$, α_S is acceptable to E if and only if it is acceptable to E'. For every set A of assignments α_S there is a Boolean expression E_A in 3-cnf to which exactly the assignments in A are acceptable. E_A can, for example, be obtained by first constructing a Boolean expression in *disjunctive* normal form which has one clause for each assignment α_S in A and to which exactly the assignments in A are acceptable, and then converting this expression to 3-cnf with the same satisfiability properties, using standard techniques (see, e.g., Hopcroft and Ullman (1979), Theorems 13.2 and 13.3). Note that the size of the expression E_A is of no concern since σ_Π is not a function of size. Since there are 2^{2^s} different such sets A, the lemma follows. □

Lemmas 2 and 3 show that there is an exponential gap between the surface complexities of 3SAT and GRAPH 2-COLORABILITY. We will show that local transformations cannot bridge such gaps. It will be helpful to further restrict 3SAT to instances in which each variable occurs in at most d clauses, for some

constant d. Such a "degree-bound" does not affect the surface complexity of 3SAT. This is stated in the following lemma.

LEMMA 4. $\sigma_{3SAT,d=9}(s) = 2^s$.

PROOF. In the graph representation of a Boolean expression in 3-cnf each clause is represented by a vertex of degree 3, and each variable is represented by a vertex whose degree is equal to the number of occurrences of that variable in the expression. By introducing new variables, plus new clauses that force these new variables to be equivalent to old ones, the number of occurrences of any one variable, and hence the degree of any vertex in the graph representation, can be brought down to 9 (and even down to 5 if one allows clauses with fewer than three literals). The surface complexity is not affected by this. □

3.2. Limitations of local transformations

A function $\sigma : \mathbb{N} \longrightarrow \mathbb{N}$ is *at least exponential* if there is a constant $c > 1$ such that $\sigma(s) \geq c^s$ for all but finitely many values of s.

LEMMA 5. *Let Π be a degree-constrained graph problem whose surface complexity σ_Π is at least exponential and let Π' be a graph problem with $\Pi \leq_{local} \Pi'$. Then $\sigma_{\Pi'}$ is at least exponential.*

PROOF. To describe the idea of the proof it helps to assume for a moment,

falsely, that $\tau_\Lambda(G \cup H) = \tau_\Lambda(G) \cup \tau_\Lambda(H)$ for all graphs G and H.[2] Claim 1 will show that this assumption is, in fact, quite close to being true. Consider two graphs G and G' with $G \not\equiv^{S}_{\Pi} G'$. There must be a graph H which demonstrates this inequivalence of G and G'. From our (false) assumption, it follows easily that $\tau_\Lambda(H)$ demonstrates the inequivalence $\tau_\Lambda(G) \not\equiv^{S_\Lambda}_{\Pi'} \tau_\Lambda(G')$, for some set S_Λ which is not too much larger than S and which depends on S and Λ but not on G or G'. Thus, nonequivalence is preserved by τ_Λ, which implies that the number of equivalence classes does not decrease in the transformation τ_Λ. A weaker (but true) version of this will be stated in Claim 3. We also need to show that S_Λ is not too much larger than S. This is stated in Claim 2.

As defined earlier, the *diameter of* a connected graph J is the largest distance between any two vertices of J. The *k-neighborhood of* a vertex v *in* a graph G, denoted by $N^k_G(v)$, is the subgraph of G that is induced by the set of those vertices of G whose distance to v is k or less. The *k-neighborhood of* a set S of vertices *in* a graph G, denoted by $N^k_G(S)$, is defined to be $\bigcup_{v \in S} N^k_G(v)$.

Let Λ be a deterministic set of local-replacement rules such that τ_Λ is a transformation of Π to Π'. Let k be the largest diameter of any left-hand side of a rule in Λ. Let $s \geq 1$, let $S = \{v_1, \cdots, v_s\}$ and let $\mathbf{G}$ be a collection of graphs with $\sigma_\Pi(s) = \log_2 |\mathbf{G}|$ such that $G \not\equiv^{S}_{\Pi} G'$ for all $G, G' \in \mathbf{G}$ with $G \neq G'$. (In other words, $\mathbf{G}$ contains exactly one representative from each equivalence class of $\equiv^{S}_{\Pi}$.)

[2] This assumption is false in general because some rule of Λ may be applicable to a subgraph J of $G \cup H$ which is neither a subgraph of G nor a subgraph of H. It is true for local transformations Λ in which all left-hand sides of rules are complete graphs, e.g., transformations which use a single edge as left-hand side of their only rule such as the transformations of Figures 2 and 3.

CLAIM 1. *Let G and H be two graphs with $V_G \cap V_H \subseteq S$. Then*

$$\tau_\Lambda(G \cup H) = \tau_\Lambda(G) \cup \tau_\Lambda(N_G^{k-1}(S) \cup H).$$

PROOF OF CLAIM 1. Let J be a vertex-induced subgraph of $G \cup H$ to which a rule of Λ is applicable. We need to show that J is a vertex-induced subgraph of at least one of the graphs G and $N_G^{k-1}(S) \cup H$. Assume otherwise. Since J is not a vertex-induced subgraph of G, it must have a vertex $u \in V_{G \cup H} - V_G$, and similarly, since J is not a vertex-induced subgraph of $N_G^{k-1}(S) \cup H$, it must have a vertex $v \in V_{G \cup H} - V_{N_G^{k-1}(S) \cup H}$. The distance between two such vertices u and v is greater than k, i.e. greater than the diameter of any left-hand side of a rule of Λ, which contradicts the assumption that some rule of Λ was applicable to J. Figure 11 provides an illustration. □

Let d be the bound on the degrees of vertices in instances of Π and let $s' = s \times d^k$. Then $|V_{N_G^{k-1}(S)}| \leq s'$ for all graphs G in $\mathbf{G}$. By renaming vertices if necessary, we may assume that the vertices of $N_G^{k-1}(S)$ all belong to the set $S' = \{v_1, \cdots, v_{s'}\}$ for all graphs $G \in \mathbf{G}$. Define $S_\Lambda = \{v : v \in V_{\tau_\Lambda(J)}$ for some graph J and $components(v) \subseteq S'\}$.

OBSERVATION 4. *$|S_\Lambda| \leq c \times |S'| = c \times s'$, where c is a constant that depends only on Λ.*

CLAIM 2. *Let $G \in \mathbf{G}$ and let H be a graph with $V_G \cap V_H \subseteq S$. Then $V_{\tau_\Lambda(G)} \cap V_{\tau_\Lambda(N_G^{k-1}(S) \cup H)} \subseteq S_\Lambda$.*

PROOF OF CLAIM 2. To arrive at a contradiction, assume that there exists a

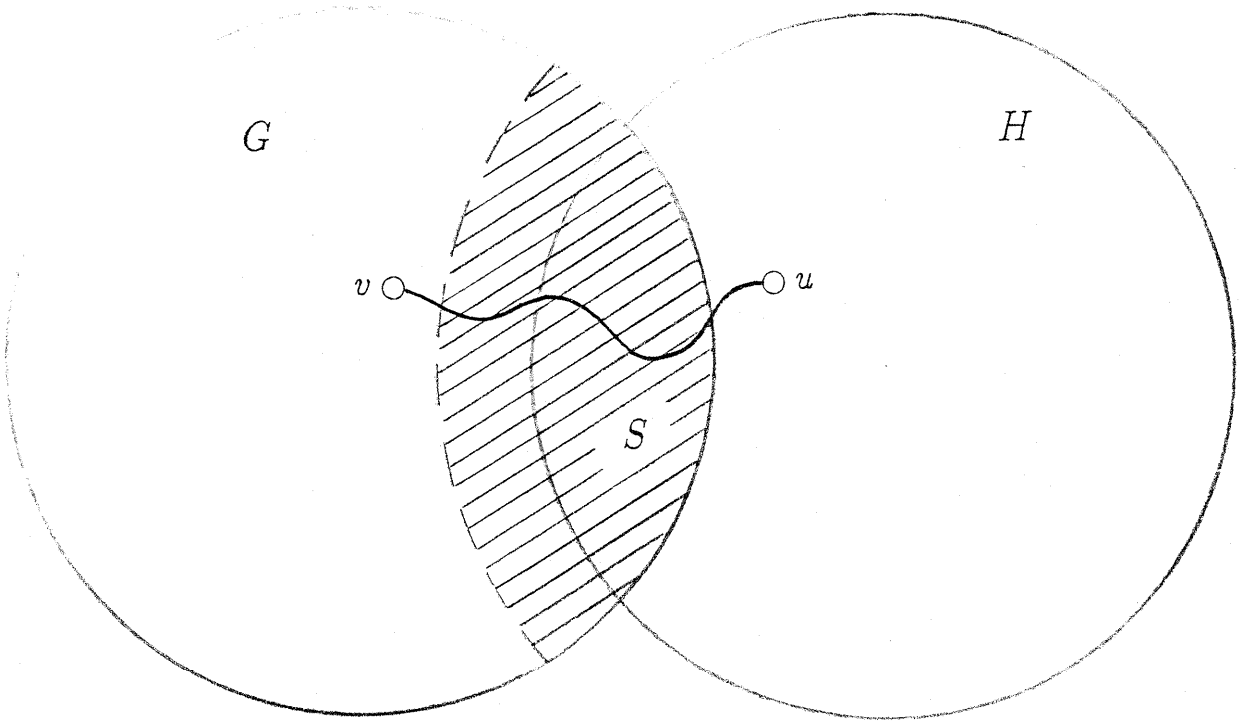

Figure 11.
Illustrating the proof of Claim 1. The shaded area is $N_G^{k-1}(S)$.

vertex $v \in V_{\tau_\Lambda(G)} \cap V_{\tau_\Lambda(N_G^{k-1}(S) \cup H)}$ with $v \notin S_\Lambda$. By the definition of S_Λ this implies that $components(v) \not\subseteq S'$. Let x be an element of $components(v) - S'$. Since $v \in V_{\tau_\Lambda(G)}$, we have $x \in V_G$ (by Observation 2), and similarly, since $v \in V_{\tau_\Lambda(N_G^{k-1}(S) \cup H)}$, we have $x \in V_{N_G^{k-1}(S) \cup H}$. Hence $x \in (V_G \cap V_{N_G^{k-1}(S) \cup H}) - S' = (V_G \cap V_H) - S' \subseteq (V_G \cap V_H) - S$, contradicting the assumption that $V_G \cap V_H \subseteq S$. □

CLAIM 3. *Let* $G, G' \in \mathbf{G}$ *with* $G \neq G'$ *and* $N_G^{k-1}(S) = N_{G'}^{k-1}(S)$. *Then* $\tau_\Lambda(G) \not\equiv_{\Pi'}^{S_\Lambda} \tau_\Lambda(G')$.

PROOF OF CLAIM 3. Let H be a graph which demonstrates the inequivalence $G \not\equiv_{\Pi}^{S} G'$. Claim 1 and the fact that $N_G^{k-1}(S) = N_{G'}^{k-1}(S)$ together imply that

$$\tau_\Lambda(G \cup H) = \tau_\Lambda(G) \cup \tau_\Lambda(N_G^{k-1}(S) \cup H)$$

and

$$\tau_\Lambda(G' \cup H) = \tau_\Lambda(G') \cup \tau_\Lambda(N_G^{k-1}(S) \cup H).$$

Since τ_Λ is a transformation from Π to Π', exactly one of these two graphs is in Π'. By Claim 2,

$$V_{\tau_\Lambda(G)} \cap V_{\tau_\Lambda(N_G^{k-1}(S) \cup H)} \subseteq S_\Lambda$$

and

$$V_{\tau_\Lambda(G')} \cap V_{\tau_\Lambda(N_G^{k-1}(S) \cup H)} \subseteq S_\Lambda.$$

Hence, by definition, $\tau_\Lambda(G) \not\equiv_{\Pi'}^{S_\Lambda} \tau_\Lambda(G')$, as demonstrated by the graph $\tau_\Lambda(N_G^{k-1}(S) \cup H)$. □

Since the vertices of $N_G^{k-1}(S)$ all belong to $S' = \{v_1, \cdots, v_{s'}\}$ for all graphs G in $\mathbf{G}$, there are at most

$$\sum_{n=0}^{s'} \binom{s'}{n} \times 2^{n^2} \;\le\; 2^{(s')^2} \times \sum_{n=0}^{s'} \binom{s'}{n} \;=\; 2^{(s')^2+s'}$$

different graphs among $\{N_G^{k-1}(S) : G \in \mathbf{G}\}$ and hence at least $|\mathbf{G}| / (2^{(s')^2+s'})$ graphs G in $\mathbf{G}$ with $N_G^{k-1}(S) = N_0$ for some fixed N_0. Combined with Observation 4 and Claim 3 this implies that

$$\sigma_{\Pi'}(c \times s') \;\ge\; \log_2(|\mathbf{G}| / (2^{(s')^2+s'})) \;=\; \sigma_{\Pi}(s) - O(s^2),$$

where c is a constant that depends only on Λ. This, combined with the fact that $\sigma_{\Pi'}$ is nondecreasing (Lemma 1), makes $\sigma_{\Pi'}$ at least exponential if, as assumed in this lemma, σ_{Π} is. This ends the proof of Lemma 5. □

MAIN RESULT, basic version. *There are no local transformations of any problem appearing in Table 1 into any problem among (d is a bound on the degree of vertices):*

2SAT,
GRAPH 2-COLORABILITY,
HAMILTONIAN CIRCUIT with d=2,
EULERIAN CIRCUIT,
VERTEX COVER with d=2,
GRAPH 3-COLORABILITY with d=3, *and*
FEEDBACK VERTEX SET with d=2.

PROOF. From Lemma 5 we know that local transformations preserve surface complexity when applied to degree-constrained graph problems. Therefore it suffices to show that all problems appearing in Table 1 have at least exponential surface complexities even when restricted to instances with some fixed bound on the degrees of vertices, and that 2SAT and the other polynomial-time

From	To	Reference
3SAT	VERTEX COVER	Garey and Johnson (1979)
3SAT	INDEPENDENT SET	Garey and Johnson (1979)
3SAT	GRAPH 3-COLORABILITY	Garey, Johnson, and Stockmeyer (1976)
3SAT	MAXIMUM 2SAT	Garey, Johnson, and Stockmeyer (1976)
VERTEX COVER	FEEDBACK ARC SET	Karp (1972)
VERTEX COVER	FEEDBACK VERTEX SET	Karp (1972)
VERTEX COVER	DOMINATING SET	Garey and Johnson (1979) [a]
VERTEX COVER	VERTEX DELETION FOR PROPERTY Π [b]	Krishnamoorthy and Deo (1979)
VERTEX COVER	MINIMUM COVER	Karp (1972)

Table 1.
Local transformations used in the basic version of the main result.

a) appears only as an exercise.

b) for Π such as being a tree, being planar, acyclic, bipartite, outerplanar, transitively orientable, chordal, and being an interval subgraph.

solvable problems listed in the theorem have at most polynomial surface complexities.

Degree-bounded 3SAT has exponential surface complexity by Lemma 4. The other problems in the first group have at least exponential surface complexities because they can be reached from 3SAT by one or more local transformations which happen to preserve "degree-boundedness", meaning that if a set of instances of an original problem has a bound on the degree of vertices then so does the corresponding set of transformed instances, although the two bounds need not be the same. (This is true of all local transformations which do not use any vertices v with $components(v)=\varnothing$ in their right-hand sides. The degree of a vertex v with $components(v)\neq\varnothing$ is bounded by $b\times 2^{d^{D+1}}$ where d is the degree bound of the original problem, D is the largest diameter of any left-hand side of a rule in Λ, and b is the largest vertex degree in any right-hand side in Λ.) These transformations are listed in Table 1.

GRAPH 2-COLORABILITY has polynomial surface complexity by Lemma 2. The other problems in that group have polynomial surface complexities by arguments similar to the proof of Lemma 2. To convince oneself of this fact, it helps to think of the computational interpretation of surface complexity given earlier and to consider adapting algorithms for these problems to a pipeline of two processes. We omit further details. □

4. Extensions: Dependence on Ordering and Other Layout Features

4.1. Motivation

Unlike the local transformations of the preceding sections, many transformations of graph problems in the NP-completeness literature are not functions on graphs. Instead, they are defined on graphs augmented by additional information such as an ordering of the vertices or the placement of crossovers in a layout of the graph. As long as this additional information does not spoil the inherent local nature of these transformations, we should be able to adapt our definitions to capture these transformations without spoiling the proof of their limitations. Technically, these extensions are captured by letting the left-hand sides of replacement rules refer to such ordering or other layout-dependent information.

4.2. Dependence on orderings of vertices and edges

The ordering of the vertices of an input graph can be represented by augmenting the graph with directed edges that are distinguished from the edges already present in the graph. These "ordering" edges hook up the vertices of the graph into a single chain. The first and the last vertex in the chain are labeled *first* and *last*. In graphs with different types of vertices, each type of vertex is ordered separately. Replacement rules may include vertex ordering information by including such ordering edges. For all examples in this section, these edges will be drawn with dotted lines. The labels *first* and *last* may also appear within the left-hand side of a replacement rule. A rule is applicable to a graph J if J is isomorphic to the left-hand side, L, of the rule where

isomorphisms must preserve all labels and all types of vertices and edges. No further changes in the definitions are necessary. Ordering of edges is handled analogously. There is a separate ordering for each type of vertex and there is, for each vertex v, a separate ordering for each type of edge incident to v.

An example of a transformation which uses ordering of edges is the transformation of GRAPH 3-COLORABILITY to GRAPH 3-COLORABILITY with d=4 of Garey, Johnson, and Stockmeyer (1976). This transformation works by replacing each vertex of degree d greater than four with an appropriate "vertex substitute" containing d "outlet" vertices. Figure 12 illustrates local-replacement rules for this transformation.

An example of a transformation that uses both vertex and edge orderings is the transformation of 3SAT into DIRECTED HAMILTONIAN PATH of Machtey and Young (1978), pp. 244-247. Figures 13a and 13b show some of the local-replacement rules that define this transformation. Each clause is replaced by what Machtey and Young call a "three-lane carriageway (TLC)". The replacement rule shown in Figure 13a is one of four rules needed to describe the clause replacement. (The right-hand side of this rule is a representative TLC.) The four rules handle the cases of 0, 1, 2, or 3 variables appearing negated within a clause. Each TLC has three "inputs" and three "outputs". For a Boolean expression in conjunctive normal form with m clauses and n variables, a directed graph with m TLC's and $n+1$ additional vertices, x_i, $1 \leq i \leq n+1$, is constructed, as follows. As indicated above, each clause is replaced by a TLC with three inputs and three outputs, labeled according to the literals which appear in the clause. For each variable x_i, two paths are constructed: one from x_i to x_{i+1} which connects all TLC's produced by clauses containing the literal x_i, and, similarly, one from x_i to x_{i+1} which connects all TLC's produced

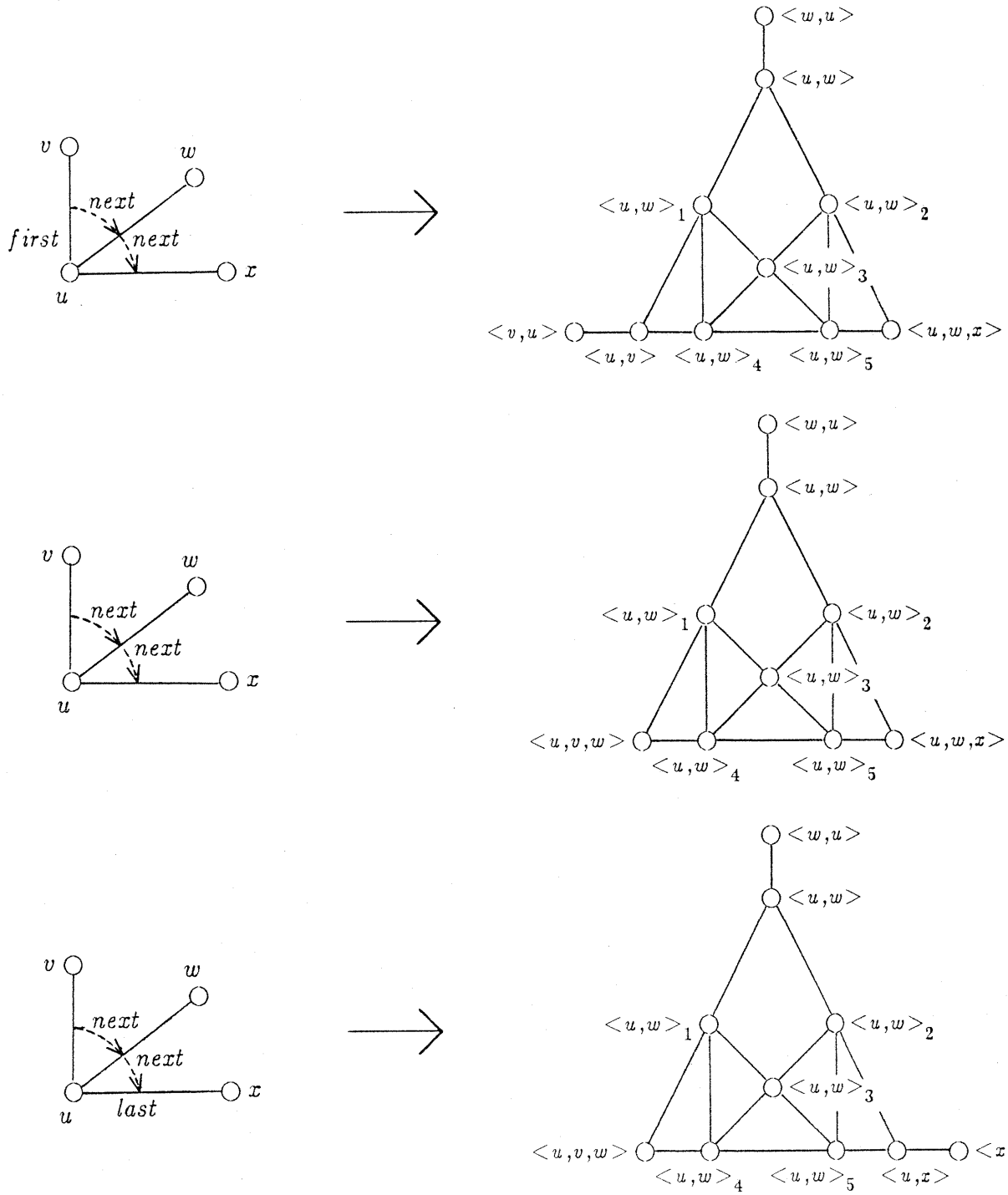

Figure 12.

Local-replacement rules for transforming GRAPH 3-COLORABILITY to GRAPH 3-COLORABILITY with d=4. The given set of three rules is sufficient for all instances in which every vertex has degree greater than 4. More rules are needed to handle instances which contain vertices of lower degree.

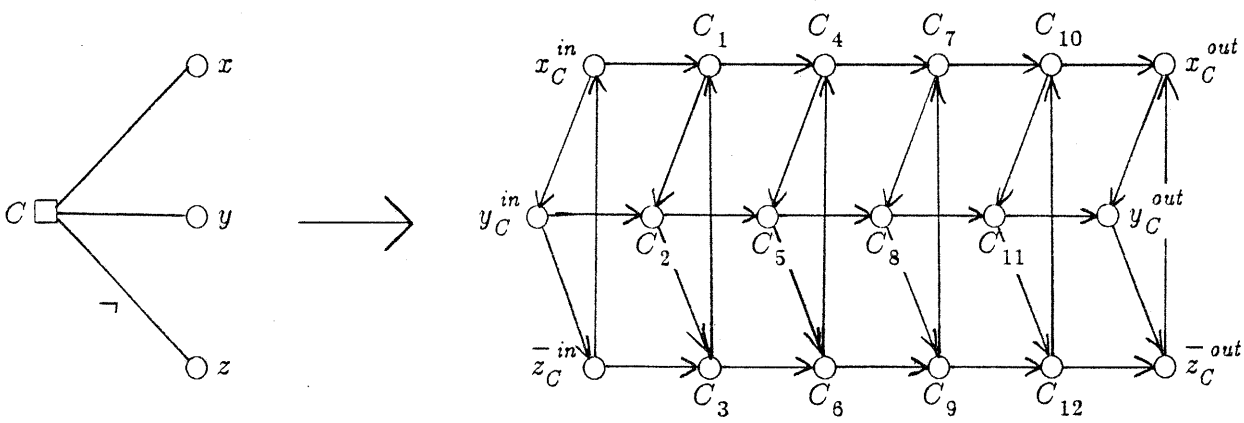

Figure 13a.

A local-replacement rule for transforming 3SAT to DIRECTED HAMILTONIAN PATH. This is one of four similar rules for transforming clauses. Which one of the four rules applies depends on the number of negations in a clause.

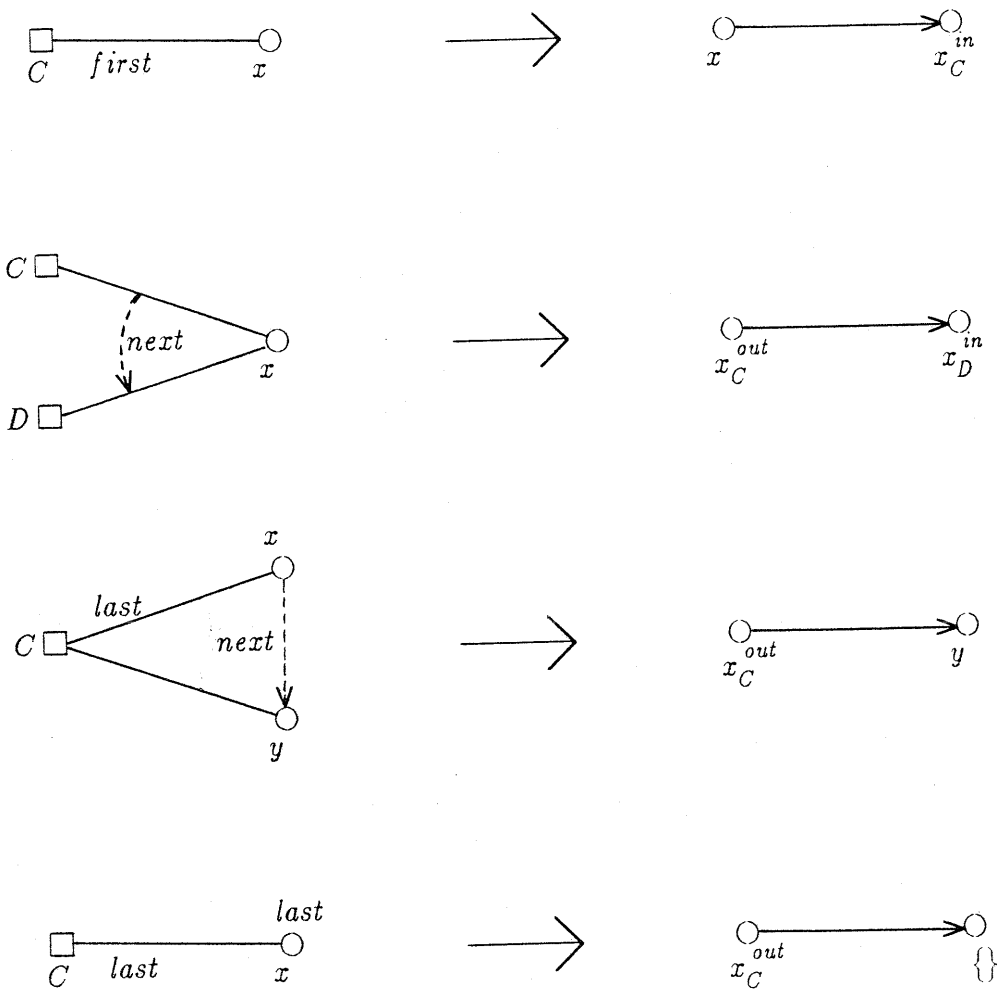

Figure 13b.

Four local-replacement rules for transforming 3SAT to DIRECTED HAMILTONIAN PATH. These four rules handle the TLC connections for the case of a nonnegated variable; four others are needed for negated variables.

by clauses containing the literal $\bar{x}_i$. Two TLC's are connected by introducing an edge directed from an output to a corresponding input. (As a minor detail, if some literal, say either x_i or $\bar{x}_i$, does not appear in any clause, the path from x_i to x_{i+1} just described connects *no* TLC's; i.e., the path is just the single edge $<x_i, x_{i+1}>$. For the correctness of the transformation, it doesn't matter whether or not such a degenerate path is present. The rules in Figure 13b do not produce such a degenerate path.)

4.3. Dependence on embedding

Transforming graph problems II into PLANAR II is commonly done by picking an arbitrary embedding of the graph in the plane and replacing each crossover by a suitable planar graph. These transformations are about as local in character as possible. In order to capture them, we let left-hand sides of local-replacement rules have access to information about crossovers. This can be achieved by replacing each crossover with a vertex of a new type and by ordering the edges incident to that new vertex as they were ordered, say, clockwise, in the embedding.

4.4. Further extensions

Further extensions are clearly possible. For example, any kind of "local preprocessing" of the input is likely to leave the proof of the main result intact. As one specific example, consider the transformation of GRAPH 3-COLORABILITY into PLANAR GRAPH 3-COLORABILITY described in Garey and Johnson (1979), pp. 85-86. Like many "II-into-PLANAR II"-transformations, it

works by replacing crossovers but it also requires an arbitrary orientation of the edges of the input graph. Adding such an orientation is a perfect example of "local preprocessing" and poses no problem for the main proof. As another example, consider transforming instances of graph problems in which each vertex has degree at most d into instances in which each vertex has exactly degree d, e.g. transforming VERTEX COVER with a degree bound of 3 into 3-REGULAR VERTEX COVER. Such a transformation is achieved by attaching copies of some suitable graph to vertices whose degree needs to be increased, as shown in Figure 14 for VERTEX COVER, and is an intrinsically local transformation. The difficulty in capturing such a transformation in our framework is that any rule that is applicable to a vertex of degree, say, 2, will also be applicable to any vertex of degree higher than 2. This problem is easily taken care of, by labeling each vertex with its degree, for example. Again, this is a local kind of preprocessing which does not spoil the proof of the main result.

It is worth noting that our notion of "local" is based on the graph-theoretic notion of "distance" between two vertices. In other words, local replacements act on subgraphs of bounded diameter. There are transformations that are local in a different sense. Consider the transformation of VERTEX COVER into INDEPENDENT SET (and vice versa) which replaces a graph by its complement. This is not a graph-theoretically local transformation because it puts edges between pairs of vertices which were at unbounded (even infinite) distances from each other in the original graph. But it is very much a local transformation in the sense that it works on small submatrices (1-by-1 submatrices, to be exact) of the adjacency matrix of the original graph. Still, it would take a substantial change in our approach to capture this type of transformation. A similar situation arises in transformations that use "garbage collection". An example is the transformation of 3SAT into 3DM in Garey and Johnson (1979), pp. 50-53.

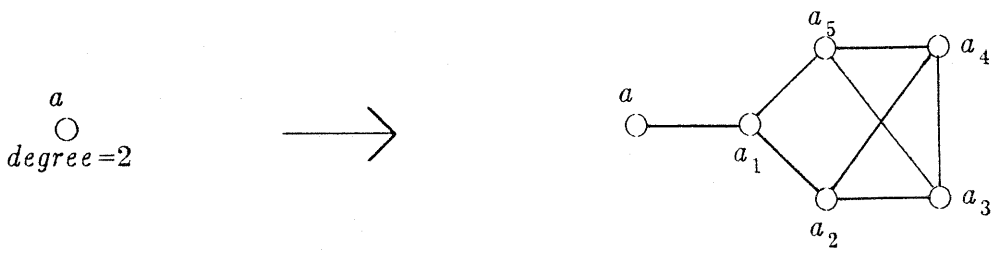

Figure 14.

One of four local-replacement rules for transforming VERTEX COVER with $d=3$ to 3-REGULAR VERTEX COVER.

From	To	Reference	Extension
3SAT	1-3SAT	Dyer and Frieze (1986)	edge-ordering
3SAT	3SAT, d=9	———	edge-ordering
GRAPH 3-COLORABILITY	PLANAR GRAPH 3-COLORABILITY	Garey, Johnson and Stockmeyer (1976)	embedding, edge-orientation
VERTEX COVER	PLANAR VERTEX COVER	Garey, Johnson and Stockmeyer (1976)	embedding
3SAT	GRAPH 3-COLORABILITY	Garey, Johnson and Stockmeyer (1976)	see footnote [a]
GRAPH 3-COLORABILITY	GRAPH 3-COLORABILITY, d=4	Garey, Johnson and Stockmeyer (1976)	edge-ordering
3SAT	GRAPH GRUNDY NUMBERING	van Leeuwen (1976) or see Monien and Sudborough (1981)	edge-ordering
1-3SAT	3-DIMENSIONAL MATCHING, d=3	Dyer and Frieze (1986)	———

Table 2.

Local transformations used in the extended version of the main result.

(Continued on next page)

[a] This transformation needs to be modified to preserve degree-boundedness, which is necessary to make Lemma 5 applicable to the next entry in the table.

From	To	Reference	Extension
1-3SAT	PLANAR 1-3SAT	Lichtenstein (1982)	embedding, edge-ordering
3-DIMENSIONAL MATCHING	EXACT 3-COVER	———	———
EXACT 3-COVER	SET PACKING	———	———
EXACT 3-COVER	HITTING SET	———	———
3SAT	DIRECTED HAMILTONIAN PATH	Machtey and Young (1978)	edge-ordering, vertex-ordering
DIRECTED HAMILTONIAN PATH	UNDIRECTED HAMILTONIAN PATH	Karp (1972)	———
VERTEX COVER	3-REGULAR VERTEX COVER	Garey and Johnson, unpublished	degree-dependent
EXACT 3-COVER	PARTITION INTO TRIANGLES	Garey and Johnson (1979)	———

Table 2.

Local transformations used in the extended version of the main result.

MAIN RESULT, extended version. *There are no local transformations of any problem appearing in Table 2 into* 2SAT *or any of the other polynomial-time solvable problems listed in the basic version of the main result.*

PROOF. All of the extensions discussed (types of vertices and edges, ordering of vertices and edges, layout-dependence, edge-orientation, degree-dependence) amount to augmenting the original instances by additional information. The results are not affected as long as this preprocessing preserves degree-boundedness and preserves surface complexity up to a polynomial, which is the case with all the extensions discussed. We leave the verification of this claim to the reader. □

5. REFERENCES

Cook, S. A. (1971), The complexity of theorem-proving procedures, *in* "Proc. 3rd Ann. ACM Symposium on Theory of Computing," Assoc. for Computing Machinery, New York, 151-158.

Dyer, M. E., and Frieze, A. M. (1986), Planar 3DM is NP-complete, *J. Algorithms* **7**, 174-184.

Garey, M. R., and Johnson, D. S. (1979), "Computers and Intractability: A Guide to the Theory of NP-Completeness," Prentice Hall, Englewood Cliffs.

Garey, M. R., Johnson, D. S., and Stockmeyer, L. (1976), Some simplified NP-complete graph problems, *Theor. Comput. Sci.* **1**, 237-267.

Hopcroft, J. E., and Ullman, J. D. (1979), "Introduction to Automata Theory, Languages, and Computation," Addison Wesley, Reading.

Karp, R. M. (1972), Reducibility among combinatorial problems, *in* "Complexity of Computer Computations" (R. E. Miller and J. W. Thatcher, Eds.), Plenum Press, New York, 85-104.

Krishnamoorthy, M. S., and Deo, N. (1979), Node-deletion NP-complete problems, *SIAM J. Comput.* **8**, 619-625.

Lakshmipathy, N., and Winklmann, K. (to appear), 'Global' graph problems tend to be intractable, *J. Comput. System Sci.*

van Leeuwen, J. (1976), Having a Grundy-numbering is NP-complete, Report No. 207, Computer Science Department, The Pennsylvania State University, University Park, Pennsylvania.

Lichtenstein, D. (1982), Planar formulae and their uses, *SIAM J. Comp.* **11**, 329-343.

Machtey, M., and Young, P. (1978), "An Introduction to the General Theory of Algorithms," Elsevier North-Holland, New York.

Monien, B., and Sudborough, I. H. (1981), Bandwidth-constrained NP-complete problems, *in* "Proc. 13th Ann. ACM Symposium on Theory of Computing," Assoc. for Computing Machinery, New York, 207-217.

Papadimitriou, C. H., and Sipser, M. (1984), Communication complexity, *J. Comput. System Sci.* **28**, 260-269.

Slough, W. A. (1984), "On the power of 'locally defined' transformations between combinatorial problems," PhD Thesis, Washington State University.

Yao, A. C. (1979), Some complexity questions related to distributive computing, *in* "Proc. 11th Ann. ACM Symposium on Theory of Computing," Assoc. for Computing Machinery, New York, 209-213.