June 1990

UILU-ENG-90-2218
DAC-19

COORDINATED SCIENCE LABORATORY
College of Engineering

SOME NP-COMPLETE PROBLEMS IN THE PHYSICAL DESIGN OF DIGITAL INTEGRATED CIRCUITS

Youssef G. Saab
Vasant B. Rao

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS None	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-90-2218 (DAC-19)		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois	6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION Semiconductor Research Corporation	
6c. ADDRESS (City, State, and ZIP Code) 1101 W. Springfield Ave. Urbana, IL 61801		7b. ADDRESS (City, State, and ZIP Code) P.O. Box 12053 Research Triangle Park, NC 27709	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Semiconductor Research Corporation	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER SRC 87-DP-109HAJ	
8c. ADDRESS (City, State, and ZIP Code) P.O. Box 12053 Research Triangle Park, NC 27709		10. SOURCE OF FUNDING NUMBERS	

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Some NP-Complete Problems in the Physical Design of Digital Integrated Circuits

12. PERSONAL AUTHOR(S)
Saab, Youssef G. and Rao, Vasant B.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM ____ TO ____	1990 June	29

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Combinational optimization, NP-Complete problems, partitioning, placement.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)
In this report we formulate certain specific optimization problems commonly occurring in the physical design process of digital integrated circuits such as the problem of testable nets, exchange, feed-through, and track-reduction. We show that the decision versions of these optimization problems are NP-Complete.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

SOME NP-COMPLETE PROBLEMS IN THE PHYSICAL DESIGN OF DIGITAL INTEGRATED CIRCUITS†

Youssef G. Saab and Vasant B. Rao

Coordinated Sciences Laboratory and
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
1101 W. Springfield Avenue,
Urbana, IL 61801

ABSTRACT

In this paper we formulate certain specific optimization problems commonly occurring in the physical design process of digital integrated circuits such as the problem of testable nets, exchange, feed-through, and track-reduction. We show that the decision versions of these optimization problems are NP-Complete. The problems considered in this paper are often considered as sub-problems within more general problems such as cell-placement and wire-routing, and are not given the full attention they deserve. Hence, our major contribution is in identifying these sub-problems as separate problems, and showing that they themselves are computationally intractable, thereby justifying the use of heuristics for their solution.

† This work was supported by the Semiconductor Research Corporation under contract # SRC-87-DP-109HAJ.

1. INTRODUCTION

The main application of theory of NP-Completeness is to allow algorithm designers justify the development of heuristic approaches for solving such problems. Once a problem has been shown to be NP-Complete, it joins a large (and continually growing) class of "inherently hard" problems for which there are no known efficient solution techniques and there is already an overwhelming evidence supporting the fact that such problems can never be solved efficiently (i.e., in polynomial time). In this paper, we identify several combinatorial optimization problems commonly occurring in the physical design of integrated circuits. We then show formally that the decision versions of these optimization problems are NP-Complete. In doing so we have added some new problems to the already existing long list of NP-Complete problems thereby justifying the use of heuristics in solving these sub-problems occurring within the much harder layout problem of integrated circuits.

The decision problems in this paper will be described in the format used in [4]. Proving that a new problem Π is NP-Complete consists of the following three steps :

(1) Showing that Π is in NP.

(2) Selecting a known NP-Complete problem Π' and constructing a transformation from Π' to Π.

(3) Proving that the above transformation can be performed in polynomial time.

In this paper, we will focus only on step (2) in the proofs. Steps (1) and (3) can be easily verified by the reader if necessary for each of the problems considered in this paper and will therefore be omitted from the proofs.

In Appendix A, we list three basic NP-Complete problems, namely, 3SAT, PARTITION, and HITTING_SET, for the sake of convenience of the reader. In Appendix B, we introduce two intermediate problems n-HITTING_SET and 3-MAX_FULL_SAT and establish their NP-Completeness. In each of the remaining sections, we consider some commonly occurring sub-problems in the physical design and layout of integrated circuits and establish their NP-Completeness by transforming one of the problems in the appendices to the problem under consideration. The reader is advised to read the appendices before proceeding with the rest of the paper.

2. The problem of TESTABLE_NETS

In the packaging process of large systems a complex partitioning problem encountered is as follows. Consider a large system of cells (or modules) of different sizes interconnected by a set of nets. One wishes to partition this system of cells into N packages (or parts) of disjoint cells subject to certain size and connection constraints [1,8]. A typical size constraint is that the sum of the sizes of the cells in each package is within a prespecified size-bound, resulting in a balanced partition. A net is said to be *internal* with respect to a partition if all its cells are assigned to the same package by the partition; otherwise, the net is said to be *cut* by the partition, or is *external* with respect to the partition. A typical connection constraint is that the total number of nets cut by the partition is below a prespecified bound. In some applications, however, it may be desired that certain nets are *forced to be external nets* so that they are externally available, say, for testing purposes [1,7,8]. To enforce this requirement, at least two cells of each of these nets must be locked in different packages. This is normally done before partitioning; hence, it is desirable that the number of locked cells be as small as possible, so that their influence on the subsequent partitioning step is minimal. This motivates the following decision problem:

TESTABLE_NETS

INSTANCE: A collection C of subsets of a finite set S and two positive integers N and K.

QUESTION: Is there a collection $D = \{D_1, D_2, \ldots, D_p\}$ of p disjoint subsets of S, such that $p = |D| \leq N$, $\sum_{i=1}^{p} |D_i| \leq K$, and each subset in C intersects with at least two different subsets in D, i.e., for each $c \in C$, there exist D_i and D_j, with $i \neq j$, such that both $c \cap D_i \neq \emptyset$ and $c \cap D_j \neq \emptyset$?

In the TESTABLE_NETS problem, the set S is the union of all cells belonging to external nets, C is the collection of all external nets, and D is the collection of all cells to be locked so that the nets in C are forced to be external nets. The cells in D_i are to be locked in the i-th package (or part). The integer N is the number of packages, while K is a bound on the total number of locked cells.

As an example consider the following instance to TESTABLE_NETS:

$$S = \{1,2,3,4,5,6\}$$

$$C = \left\{ \{1,2,4,6\}, \{2,3,6\}, \{2,3,4,5\}, \{1,2,3,5\} \right\}$$

$$N = 2$$

$$K = 3$$

It can be easily verified that a solution to the above instance is given by

$$D = \left\{ \{1,3\}, \{2\} \right\}.$$

Theorem 1: TESTABLE_NETS is NP-Complete.

Proof: We reduce 2-HS (described in Appendix B) to TESTABLE_NETS. Let a collection C of subsets of a finite set S and a positive integer $K \le |S|$ denote an arbitrary instance of 2-HS. We construct an instance of TESTABLE_NETS given by a collection $\hat{C}$ of subsets of a finite set $\hat{S}$ and two positive integers N and $\hat{K}$ as follows :

$$\hat{S} = S$$

$$\hat{C} = C$$

$$\hat{K} = K$$

$$N = K$$

Let $S' = \{s_1, s_2, \ldots, s_p\} \subseteq S$ be a solution for the given instance of 2-HS, i.e., $p = |S'| \le K$ and S' contains at least 2 elements from each subset in C. Then the collection $D = \left\{ \{s_1\}, \{s_2\}, \ldots, \{s_p\} \right\}$ of the singletons of S' is clearly a solution to the constructed instance of TESTABLE_NETS. Conversely, if $D = \{D_1, D_2, \ldots, D_p\}$ is a solution to the constructed instance of TESTABLE_NETS, then $S' = \bigcup_{i=1}^{p} D_i$ can be easily verified to be a solution to the given instance of 2-HS. $\square$

3. Exchange problems

The *bisectioning problem* is a special case of the general partitioning problem wherein one wishes to partition a given system of cells interconnected by nets into *two parts* such that the number of nets cut is minimized, subject to a balance in size constraint. There are several iterative improvement algorithms for bisectioning [2,3] that essentially start with an initial bisection and try to improve on it by exchanging subsets of cells between the two parts. This exchange step needs to account for the balance in size constraint. Suppose that the initial bisection consists of two parts P_1 and P_2. Suppose also that P_1 exceeds its size bound by a value $B > 0$. In this case the exchange procedure attempts to find $S_1 \subseteq P_1$ and $S_2 \subseteq P_2$ such that $|\text{size}(S_1) - \text{size}(S_2) - B|$ is as small as possible. Here, the size of a set is the sum of sizes of its elements. The decision version of this optimization problem can now be stated as:

2_EXCHANGE

INSTANCE: Two finite disjoint sets P_1 and P_2, a positive integer size $s(p)$ for each $p \in P_1 \cup P_2$, and two nonnegative integers B and K.

QUESTION: Are there subsets $S_1 \subseteq P_1$ and $S_2 \subseteq P_2$ such that

$$\left| \sum_{p \in S_1} s(p) - \sum_{p \in S_2} s(p) - B \right| \leq K \text{ ?}$$

Theorem 2: 2_EXCHANGE is NP-Complete.

Proof: We reduce PARTITION (described in Appendix A) to 2_EXCHANGE. Let a set A with a positive integer size $s(a)$ for each $a \in A$ denote an arbitrary instance of PARTITION. We now construct an instance of 2_EXCHANGE given by two sets P_1 and P_2, a positive integer size $\rho(p)$ for each $p \in P_1 \cup P_2$, and two nonnegative integers B and K as follows:

$P_2 = A$,
$\rho(p) = s(p)$ for each $p \in P_2$,
$T = \sum_{a \in A} s(a)$ the total size of all the elements in A,
$B = T+1$,
$K = 0$, and
$P_1 = \{y\}$ is a singleton set with a new element y which has size $\rho(y) = \frac{3T}{2} + 1$.

If $A' \subseteq A$ is a solution for the given instance of PARTITION, then, by construction, $S_1 = P_1$ and $S_2 = A'$ form a solution for the constructed instance of 2_EXCHANGE. Conversely, let S_1 and S_2 be a solution for the constructed instance of 2_EXCHANGE. Since P_1 is a singleton, by construction, there are only two possible choices for the subset S_1, namely, $S_1 = P_1$ or $S_1 = \varnothing$. But $S_1 = \varnothing$ means

$$\left| \sum_{p \in S_1} \rho(p) - \sum_{p \in S_2} \rho(p) - B \right| = \left| \sum_{p \in S_2} \rho(p) + B \right| > 0 = K$$

for any choice of subset $S_2 \subseteq P_2$. Therefore, we must have $S_1 = P_1$. Since S_1 and S_2 solve the constructed instance of 2_EXCHANGE, we get

$$\left| \sum_{p \in S_1} \rho(p) - \sum_{p \in S_2} \rho(p) - B \right| = \left| \frac{3T}{2} + 1 - \sum_{p \in S_2} \rho(p) - T - 1 \right| = \left| \sum_{p \in S_2} \rho(p) - \frac{T}{2} \right| = 0$$

Therefore, $A' = S_2 \subseteq P_2 = A$ is a solution to the given instance of PARTITION. □

The 2_EXCHANGE problem remains NP-Complete even if one of the two sets P_1 or P_2 is empty. The statement of the problem then becomes:

1_EXCHANGE

INSTANCE: A finite set P, a positive integer size $s(p)$ for each $p \in P$, and two nonnegative integers B and K.

QUESTION: Is there a subset $S \subseteq P$ such that

$$\left| \sum_{p \in S} s(p) - B \right| \leq K \text{ ?}$$

The reduction of PARTITION to 1_EXCHANGE can be easily done by setting $P = A$, $K = 0$, and $B = \frac{T}{2}$, where $T = \sum_{a \in A} s(a)$ is the total size of all the elements in A.

4. The feed-through problem

The standard cell approach is commonly used in the physical design of VLSI circuits [7]. In this approach a set of standard cells have been previously handcrafted, laid out, tested, characterized, and stored in a technology library. The individual standard cells, after layout, have a common height and may differ only in length. A standard cell interacts with other cells through *pins* which are located only at the top and bottom sides of the cell. In the layout of standard cells, the cells are assigned to rows and the interconnection nets (or wires) among the pins of these cells are routed in channels between rows. An example of a layout of 9 standard cells and 10 nets is shown in Figure 1.

Whenever a net has to cross a given row a *feed-through cell* has to be inserted in that row to connect the two parts of the net above and below the row. For example, in Figure 1, net d crosses the middle row while net j crosses the bottom row. Hence, two feed-through cells, one between cells 4 and 5, and another between cells 7 and 8, have to be inserted to provide the required interconnection. Since these feed-through cells increase the area, we would like the total number of such cells minimized. Sometimes, by flipping the cells of a given row we can reduce the number of feed-through cells in that row as illustrated in Figure 2. In Figure 2(a), nets c, d, and e cross the middle row. Hence, three feed-through cells are needed in this case. However, if cell 2 is flipped around its horizontal axis, as shown in Figure 2(b), we find that only nets a and b cross the middle row, thereby reducing the number of feed-through cells in this row by one.

The above discussion motivates the following feed-through optimization problem which is performed *after* the placement of the standard cells into rows is completed and *before* the routing phase begins. In the feed-through optimization problem, one attempts to find the set of cells to be *flipped*

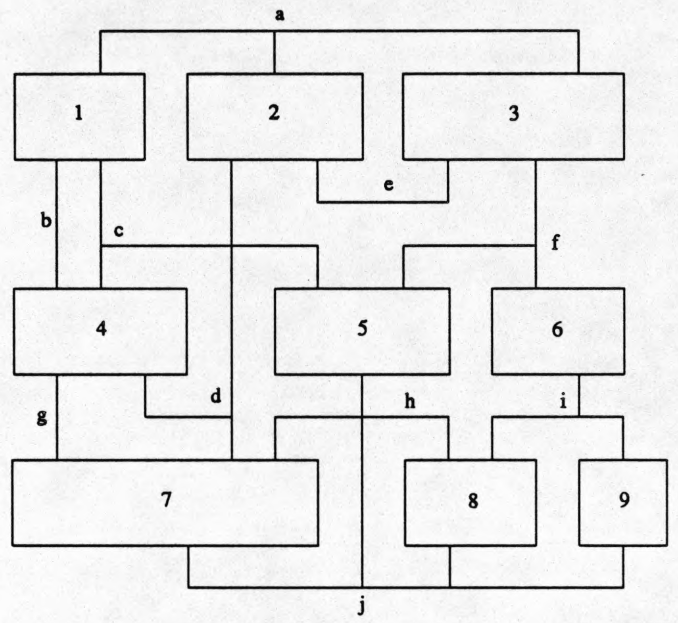


Figure 1 : An example of a standard cell layout.

around their horizontal axis, so that the total number of feed-through cells is minimized. It is important to note that the number of feed-through cells in a given row is not affected by the orientation of cells in the other rows. Hence, the feed-through optimization problem can be solved independently for each row. For the remainder of this section, therefore, we consider only one row.

In order to give the abstract formulation of the problem, we need an adequate representation. Consider a particular row of p standard cells. Suppose the cells are labeled by positive integers from 1 through p. Let $U = \{u_1, u_2, \ldots, u_p\}$ be a collection of p Boolean variables and let

$$V = \{u_1, \bar{u}_1, u_2, \bar{u}_2, \ldots, u_p, \bar{u}_p\}$$

denote the collection of the $2p$ literals over U. Let us arbitrarily label the two horizontal sides (the top

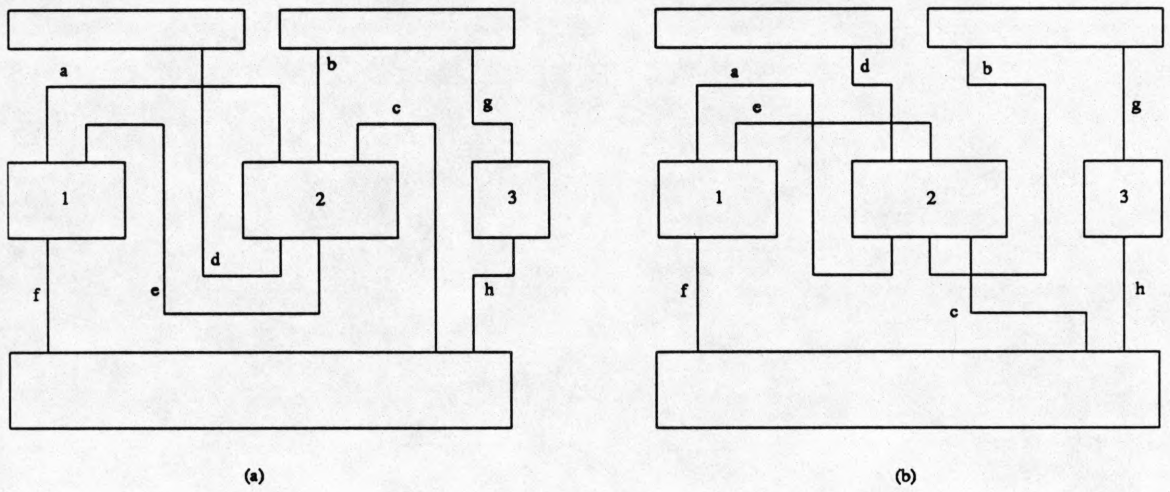


Figure 2 : An illustration of feed-through reduction.

and bottom sides containing the pins) of the cell i by Boolean literals u_i and $\overline{u}_i$ respectively. We then consider the cell i oriented with its side labeled u_i on the top whenever $u_i = 1$ and on the bottom whenever $u_i = 0$. Thus, each truth assignment over U corresponds to a particular orientation (with respect to the horizontal axis) of the cells in the particular row under consideration. In this setting, a net is a collection of pins to be electrically connected. Consider the set of nets connecting pins on the cells of the row under consideration. Such a net is represented by a *clause* of Boolean literals drawn from $V \cup \{0,1\}$ as follows:

(1) If a net contains a pin on the side labeled u_i ($\overline{u}_i$) of cell i, then include the literal u_i ($\overline{u}_i$) in its representation-clause.

(2) If a net contains a pin on a cell in a row above (below) the row under consideration, then include the literal 1 (0) in its representation-clause.

Here, 1 is treated as a literal that is always assigned a true value of 1 by any truth assignment. Similarly, 0 is treated as a literal that is always assigned a false value of 0 by any truth assignment. For example, consider the middle row of cells in the layout of Figure 2(a). This row has 3 cells and we label the *top* side of cell i by u_i (the choice is arbitrary). There are eight 2-terminal nets connecting pins on the cells of this row and their representation-clauses are given in Table 1 below.

It can be easily seen that for a particular orientation of cells of a row, a net *will not cross* the given row if and only if *all* the literals in its representation-clause are assigned the *same* value by the corresponding truth assignment. This gives rise to the following:

Definition : A clause of literals is *fed-through* by a truth assignment over U if the conjunction (AND) of its literals is 0 and the disjunction (OR) of its literals is 1 under that assignment.

In other words, a clause is fed-through if any two of its literals are assigned different values by a truth assignment. Going back to the above example, the truth assignment $(u_1,u_2,u_3) = (1,1,1)$ corresponding to the orientation of cells as shown in Figure 2(a) results in three feed-through clauses, namely, the ones representing nets c, d, and e. However, the truth assignment $(u_1,u_2,u_3) = (1,0,1)$ corresponding to the orientation of cells as shown in Figure 2(b) results in only two feed-through clauses, namely, the ones representing nets a and b. In fact, it can be easily verified (by exhausting all possibilities) that the assignment (1,0,1) results in the minimum number of feed-through clauses; hence, the layout of Figure 2(b) will result in the smallest number of feed-through cells (2 in this case) that need to be inserted in the middle row of the layout. The abstract decision version of our feed-through optimization problem can now be stated as :

FEED_THROUGH

INSTANCE: A set $U = \{u_1, u_2, \ldots, u_p\}$ of Boolean variables, a collection C of clauses where each clause is a subset of literals from $\{0, 1, u_1, \bar{u}_1, u_2, \bar{u}_2, \ldots, u_p, \bar{u}_p\}$, and a non-negative integer K.

Table 1 : A representation of nets of Figure 2(a).

Net	Representation
a	$\{u_1,u_2\}$
b	$\{1,u_2\}$
c	$\{0,u_2\}$
d	$\{1,\bar{u}_2\}$
e	$\{u_1,\bar{u}_2\}$
f	$\{0,\bar{u}_1\}$
g	$\{1,u_3\}$
h	$\{0,\bar{u}_3\}$

QUESTION: Is there a truth assignment for U such that the number of clauses in C that are fed-through is at most K ?

We now distinguish between three types of values of a clause of Boolean literals. The OR-value of a clause is simply the disjunction (OR), the AND-value is the conjunction (AND), while the EXOR-value is the EXCLUSIVE-OR of its AND-value and OR-value. In the 3SAT problem we seek a truth assignment such that each clause has an OR-value of 1. In the optimization version of the 3-MAX_FULL_SAT problem we seek a truth assignment that *maximizes* the number of clauses having an AND-value of 1, while in the optimization version of the FEED_THROUGH problem we seek a truth assignment that *minimizes* the number of clauses having an EXOR-value of 1. This is essentially the difference between the three problems. In Appendix B, the reduction of 3SAT to 3-MAX_FULL_SAT is given. We now consider the following result.

Theorem 3: FEED_THROUGH is NP-Complete.

Proof: We reduce 3-MAX_FULL_SAT to FEED_THROUGH. Let $U = \{u_1, u_2, \ldots, u_p\}$, $C = \{c_1, c_2, \ldots, c_m\}$ of 3-literal clauses over U, and a positive integer $k \leq m$, denote an arbitrary instance of 3-MAX_FULL_SAT. For convenience, suppose each clause $c_j = \{c_{j,1}, c_{j,2}, c_{j,3}\}$, where $c_{j,q}$ is a literal over U for each $1 \leq q \leq 3$, and $1 \leq j \leq m$. We need to construct an instance $\hat{U}$ of Boolean variables, $\hat{C}$ of clauses of literals of $\hat{U}$ together with fixed Boolean constants 0 and 1, and a positive integer K, such that there exists a truth assignment over U that fully-satisfies at least k clauses in C if and only if there is a truth assignment over $\hat{U}$ such that at most K clauses of $\hat{C}$ are fed-through. The construction is as follows:

Define the set $\hat{U} = U$.

Define the integer $K = |C| - k = m - k$. Note that $K \geq 0$.

For each $1 \leq j \leq m$, define $\hat{c}_j = \{1, c_{j,1}, c_{j,2}, c_{j,3}\}$, and set $\hat{C} = \{\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_m\}$.

Note that a clause $c_j \in C$ is *fully-satisfied* if and only if the clause $\hat{c}_i \in \hat{C}$ is *not fed-through*. Hence, a truth assignment for U solves the given instance of 3-MAX_FULL_SAT if and only if the same truth assignment for $\hat{U} = U$ solves the constructed instance of FEED_THROUGH. □

5. The track reduction problem

Consider, once again, the standard cell layout problem. After the cells have been placed in rows, and the cells have been flipped around the horizontal axis, if necessary, to minimize the number of feed-through cells, the routing of wires in the channels created between the rows remains to be performed. The height of a channel is proportional to the number of horizontal *tracks* needed to perform the required routing. Before the routing phase actually begins, it may be possible to minimize the number of tracks used for routing as well as the total-wire-length by performing appropriate flips of the cells around their *vertical axis*. A simple example in Figure 3(a) shows two nets that cannot be routed on the same track because they overlap horizontally. However, as shown in Figure 3(b), the flipping of the second cell around its vertical axis permits us not only to route the two nets on the same track but also decreases the wire-length necessary to route these two nets.

Let us define the *horizontal span of a net* to be the length of the horizontal side of the smallest rectangle that encloses all its pins. Then the track-reduction problem can be stated as finding the cells to be flipped around their vertical axis such that the sum of the spans of all the nets is minimized.

In order to give the abstract formulation of the decision version of the above optimization problem, we need a suitable model for the representation of an instance of the problem. Consider a placement of p standard cells on horizontal rows of a 2-dimensional plane. Let the x-axis and y-axis denote the horizontal and vertical axes respectively. Let $P = \{1,2,3, \ldots ,p\}$ denote the set of cells. Let $U = \{u_1,u_2, \ldots , u_p\}$ be a collection of p Boolean variables. Let us arbitrarily label one of the vertical sides (left or right) of the cell i by a Boolean literal u_i and the other side by $\overline{u}_i$. We then consider the cell i oriented with its side labeled u_i on the *left* whenever $u_i = 1$ and on the *right* whenever $u_i = 0$. Thus, each truth assignment over U corresponds to a particular orientation (with respect to the *vertical* axis) of the cells in P. In some applications, it may be necessary to include a *dummy* cell labeled 0 for which $u_0 = 1$ always and this cell is never flipped.

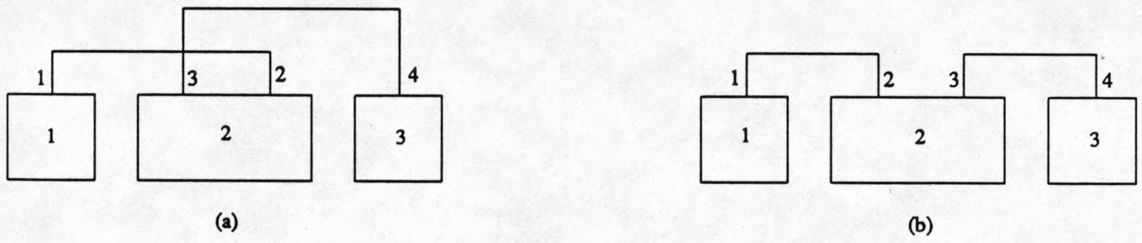


Figure 3 : An illustration of the track-reduction problem.

We now consider the set of pins on the top and bottom faces of the cells in P. Label these pins arbitrarily by positive integers and let $R = \{1,2, \dots, r\}$ denote the set of all pins. Define a function $\sigma : R \to P$ such that $\sigma(s) \in P$ denotes the cell on which pin $s \in R$ is located. Also define a real-valued function $f : \{0,1\} \times R \to \mathbb{R}$, such that $f(0,s)$ is the x-coordinate of pin s when $u_{\sigma(s)} = 0$ and $f(1,s)$ is the x-coordinate of pin s when $u_{\sigma(s)} = 1$. A net can now be specified as a subset of pins to be electrically connected.

Definition : The span of a net $c \subseteq R$ under a truth assignment over U is defined as the maximum of the absolute value of the difference in the x-coordinates among all pairs of its pins specified by that assignment. Mathematically, given a truth assignment over U, functions σ and f, and a net $c \subseteq R$, define

$$Span(c) = \max_{s,t \in c} \mid f(u_{\sigma(s)},s) - f(u_{\sigma(t)},t) \mid$$

The above definitions can easily be modified to include the dummy cell labeled 0 if necessary. We can now state the decision version of the track-reduction problem as follows:

TRACK_REDUCTION

INSTANCE: A set $P = \{1,2,3, \dots, p\}$ of cells, a set $U = \{u_1, u_2, \dots, u_p\}$ of Boolean variables, a set $R = \{1,2, \dots, r\}$ of pins, a function $\sigma : R \to P \cup \{0\}$ associating a pin with a cell (here 0 is a dummy cell), a real-valued function $f : \{0,1\} \times R \to \mathbb{R}$ providing the x-coordinates of pin locations on the cell boundary, a collection C of nets which are subsets of R, and a real number $\alpha \geq 0$.

QUESTION: Is there a truth assignment for U such that the sum of spans of all nets in C is at most α, i.e.,

$$\sum_{c \in C} \max_{s,t \in c} \mid f(u_{\sigma(s)},s) - f(u_{\sigma(t)},t) \mid \ \leq \alpha \, ?$$

Here, we assume that $u_0 = 1$ if required.

Theorem 4: TRACK_REDUCTION is NP-Complete.

Proof: We reduce FEED_THROUGH to TRACK_REDUCTION. We will first informally illustrate the construction of an instance of TRACK_REDUCTION from an instance of FEED_THROUGH. Consider an instance of FEED_THROUGH shown in Figure 4(a). The abstract representation of this instance is denoted by a set of clauses $C = \{c_1, c_2, c_3\}$ where

$$c_1 = \{1, u_1, \bar{u}_2\}$$

$$c_2 = \{1, u_2, u_3\}$$

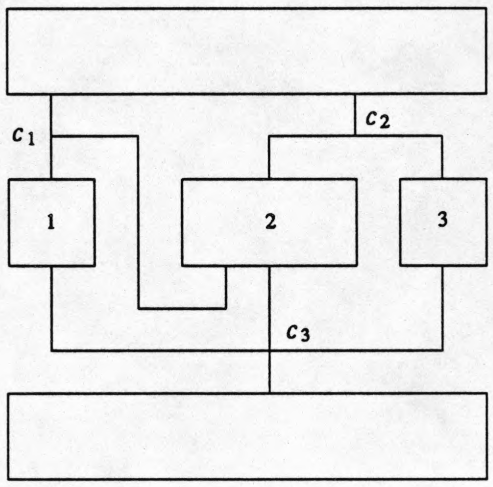
$$c_3 = \{0, \bar{u}_1, \bar{u}_2, \bar{u}_3\}$$

The corresponding instance of TRACK_REDUCTION is illustrated in Figure 4(b). Here, we have 3 cells centered around a common vertical axis and placed one below the other. Since the literal 1 or 0 is present in a clause of FEED_THROUGH, a *dummy cell* 0 is introduced and is placed on top of cell 1. Each row in the constructed instance of TRACK_REDUCTION will contain only one cell of length $2(m^2+m)$, where m is the number of clauses (nets) in the instance of FEED_THROUGH (here, $m=3$). For each clause c_j in the instance of FEED_THROUGH, a net $\hat{c}_j$ is created in the instance of TRACK_REDUCTION as follows. If $u_i \in c_j$, then the net $\hat{c}_j$ contains a pin on the top of cell i located at a distance of m^2+j-1 units to the *left* of the vertical axis. However, if $\overline{u_i} \in c_j$, then the net $\hat{c}_j$ contains a pin on the top of cell i located at a distance of m^2+j-1 units to the *right* of the vertical axis. With regards to the dummy cell 0, we always assume that $u_0 = 1$.

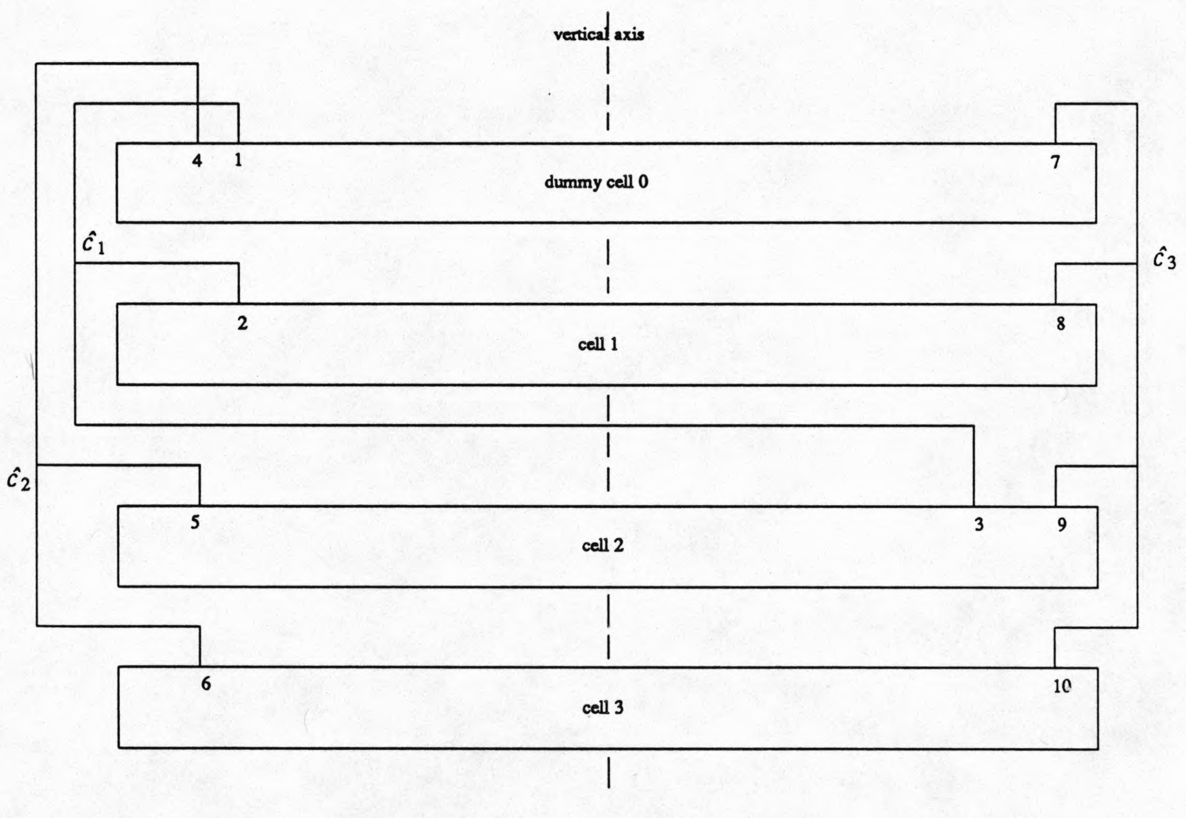
We will now give a formal reduction. Let an arbitrary instance of FEED_THROUGH be denoted by a set $U = \{u_1, u_2, \ldots, u_p\}$ of Boolean variables, a collection $C = \{c_1, c_2, \ldots, c_m\}$ of m clauses where each clause

$$c_j = \{c_{j,1}, c_{j,2}, \ldots, c_{j,\gamma_j}\}$$

is a subset of γ_j literals drawn from the set $\{0, 1, u_1, \bar{u}_1, u_2, \bar{u}_2, \ldots, u_p, \bar{u}_p\}$, and a non-negative integer K. We now construct a set $P = \{1, 2, 3, \ldots, p\}$ of cells, a set $\hat{U} = \{\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_p\}$ of Boolean



(a) FEED_THROUGH instance.



(b) TRACK_REDUCTION instance.

Figure 4 : A transformation of FEED_THROUGH to TRACK_REDUCTION.

variables, a set $R = \{1,2,\ldots,r\}$ of pins, a function $\sigma : R \rightarrow P \cup \{0\}$ associating a pin with a cell, a real-valued function $f : \{0,1\} \times R \rightarrow \mathbb{R}$, a collection of nets $\hat{C}$ of subsets of R, and a real number $\alpha \geq 0$, representing an instance of TRACK_REDUCTION as follows.

Define $p = |U|$, $\hat{U} = U$, $\hat{u}_i = u_i$ for each $1 \leq i \leq p$.

Define $r = \sum_{j=1}^{m} \gamma_j$ as the total number of pins. Let $\eta_j = \sum_{q=1}^{j-1} \gamma_q$ denote the cumulative sums of the cardinalities of the clauses. For each element $c_{j,l}$ of clause c_j create a pin $s = l + \eta_j$. If $c_{j,l} \in \{u_i, \bar{u}_i\}$, then set $\sigma(s) = i$ (i.e., pin s belongs to cell i). However, if $c_{j,l} \in \{0, 1\}$, then set $\sigma(s) = 0$ (i.e., pin s belongs to the dummy cell 0). Also, define $\beta(s) = j$ and $\delta(s) = l$.

For each pin s, define

$$f(1,s) = \begin{cases} -(m^2+\beta(s)-1) & \text{if } c_{\beta(s),\delta(s)} = u_{\sigma(s)} \\ +(m^2+\beta(s)-1) & \text{if } c_{\beta(s),\delta(s)} = \bar{u}_{\sigma(s)} \end{cases}$$

and

$$f(0,s) = -f(1,s)$$

Here we assume that $u_0 = 1$ and $\bar{u}_0 = 0$.

For each clause c_j, define a net

$$\hat{c}_j = \{s : \eta_j < s \leq \eta_{j+1}\}$$

as a collection of pins and define

$$\hat{C} = \{\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_m\}.$$

as a collection of nets.

Finally, define $\alpha = 2K(m^2+m-1)$.

We now illustrate the formal reduction by considering, once again, the example in Figure 4. Here the set $U = \{u_1, u_2, u_3\}$ of Boolean variables, the set of clauses $C = \{c_1, c_2, c_3\}$ where

$$c_1 = \{1, u_1, \bar{u}_2\},$$

$$c_2 = \{1, u_2, u_3\},$$

$$c_3 = \{0, \bar{u}_1, \bar{u}_2, \bar{u}_3\},$$

and integer $K = 1$ represents an instance of FEED_THROUGH. Clearly, p=3, m=3, $\gamma_1 = \gamma_2 = 3$, and

$\gamma_3 = 4$. Therefore, $r = \gamma_1+\gamma_2+\gamma_3 = 10$ is the total number of pins, and $\eta_1 = 0$, $\eta_2 = 3$, $\eta_3 = 6$, and $\eta_4 = 10$. This gives us $R = \{1,2, \ldots, 10\}$ as the set of pins. The values of the functions β, δ, and σ for each of the pins are given by the following table.

s	$\beta(s)$	$\delta(s)$	$\sigma(s)$
1	1	1	0
2	1	2	1
3	1	3	2
4	2	1	0
5	2	2	2
6	2	3	3
7	3	1	0
8	3	2	1
9	3	3	2
10	3	4	3

The values of the function $f(1,s)$ are given in the following table.

s	$f(1,s)$
1	-9
2	-9
3	+9
4	-10
5	-10
6	-10
7	+11
8	+11
9	+11
10	+11

and $f(0,s) = -f(1,s)$ for each pin $s \in R$. The set of nets is given by

$$\hat{C} = \left\{ \{1,2,3\}, \{4,5,6\}, \{7,8,9,10\} \right\}.$$

as shown in Figure 4(b). Finally, the integer $\alpha = 22$.

To complete the proof we make the following simple observations.

(1) The *span of a net* $\hat{c}_j$ in the constructed instance of TRACK_REDUCTION is either 0 or $2(m^2+j-1)$.

(2) The clause c_j in the instance of FEED_THROUGH is fed-through by a truth assignment over U *if and only if* the corresponding net $\hat{c}_j$ in the constructed instance of TRACK_REDUCTION has a span of $2(m^2+j-1)$ under the same truth assignment over $\hat{U} = U$.

Now suppose there is a truth assignment for U that solves FEED_THROUGH, i.e., there are at most K clauses fed-through in the given instance of FEED_THROUGH. By Observation (2), there are at most K nets with nonzero span in the constructed instance of TRACK_REDUCTION under the same truth assignment. But the maximum span of a net in the constructed instance of TRACK_REDUCTION is $2(m^2+m-1)$. Consequently, the sum of spans of all nets for the instance of TRACK_REDUCTION is at most $\alpha = 2K(m^2+m-1)$; hence, the same truth assignment for $\hat{U} = U$ solves the constructed instance of TRACK_REDUCTION.

Conversely, assume that there is a truth assignment for $\hat{U}$ that solves the constructed instance of TRACK_REDUCTION. We will then show that the same truth assignment for $U = \hat{U}$ indeed solves the given instance of FEED_THROUGH using a proof by contradiction. Suppose that under the assumed truth assignment, there are at least $K+1$ clauses fed-through in the given instance of FEED_THROUGH. Then, by Observation (2), there are at least $K+1$ nets in the constructed instance of TRACK_REDUCTION with nonzero span. But, by Observation (1), the minimum nonzero span of a net in the constructed instance of TRACK_REDUCTION is $2m^2$. Therefore, we must have

$$2(K+1)m^2 \leq \alpha = 2K(m^2+m-1),$$

which implies

$$m^2 \leq (m-1)K. \tag{**}$$

There are two possibilities for the value of the integer K in the instance of FEED_THROUGH, namely, $K > m$ or $K \leq m$. If $K > m$ then, clearly, any truth assignment for U solves the given instance of

FEED_THROUGH. On the other hand, if $K \leq m$ then it is impossible to satisfy (**). Therefore, the number of clauses fed-through in the instance of FEED_THROUGH under the assumed truth assignment must be at most K, thus proving that the assumed truth assignment also solves the given instance of FEED_THROUGH. □

6. Conclusions

In this paper, we have considered several problems such as TESTABLE_NETS, 2-EXCHANGE, FEED_THROUGH, and TRACK_REDUCTION, that often occur as sub-problems in the physical design phase of integrated circuits. We have shown that each of these problems is NP-Complete by reducing

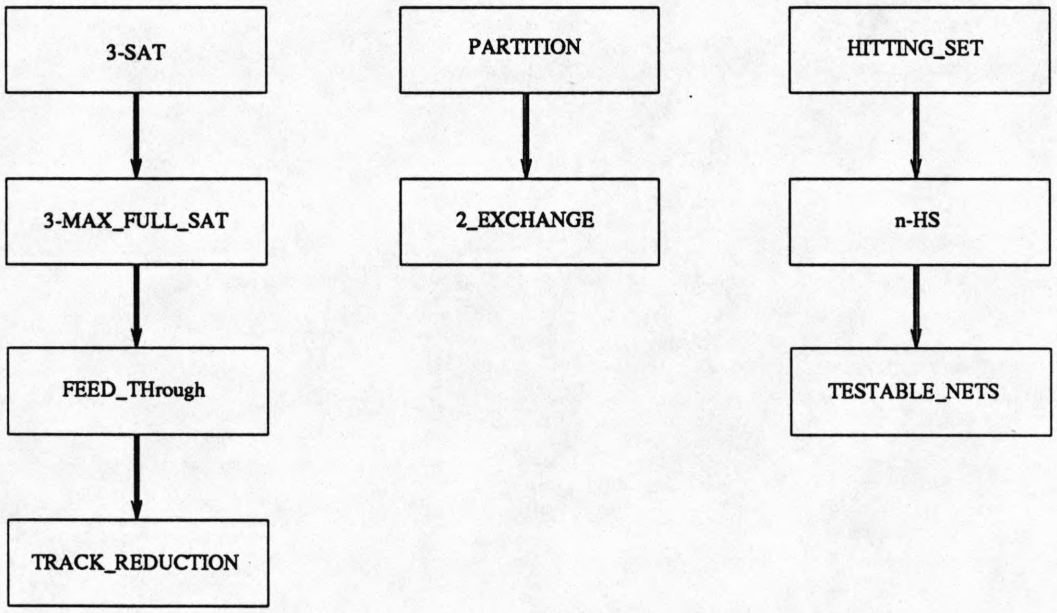


Figure 5 : An illustration of the reduction process.

three well-known and classical NP-Complete problems, namely, 3-SAT, PARTITION, and HITTING_SET. The reduction process is summarized in Figure 5. Our main purpose in establishing the NP-Completeness of these sub-problems is to justify the use of heuristics for their solution.

REFERENCES

[1] S. Goto and T. Matsuda, "Partitioning, Assignment and Placement," *Layout Design and Verification (Ed. T. Ohtsuki)*, North-Holland, 1986, pp. 55-97.

[2] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *Bell System Technical Journal*, vol. 49, pp. 291-307, February 1970,

[3] C. M. Fiduccia and R. M. Mattheyses, "A Linear-Time Heuristics for Improving Network Partitions," *Proceedings of the 19th Design Automation Conference*, pp. 175-181, January 1982.

[4] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, pp. 126, 1979.

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

[6] R. M. Karp, "Reducibility among combinatorial problems," *Complexity of Computer Computations (Eds. R. Miller and J. Thatcher)*, Plenum Press, New York, 1972, pp. 85-103.

[7] B. T. Preas, "Introduction to Physical Design Automation," *Physical Design Automation of VLSI Systems (Eds. B. Preas and M. Lorenzetti)*, The Benjamin/Cummings Publishing Company, Inc., Menlo Park, California, 1988, pp. 1-29.

[8] Y. G. Saab and V. B. Rao, "An Evolution-Based Approach to Partitioning ASIC Systems," *Proceedings of the 26th Design Automation Conference*, pp. 767-770, Las Vegas, Nevada, June 1989.

APPENDIX A

A list of three well-known NP-Complete Problems

3-SATISFIABILITY (3SAT)

The terminology used in describing this problem, borrowed from [4], is as follows. Let $U = \{u_1, u_2, \ldots, u_n\}$ be a set of Boolean *variables*. A *truth assignment for* U is an assignment of a fixed *value* 0 or 1 to each variable in U. Here a value 0 means "false", while value 1 means "true". If u is a variable in U, then u and $\bar{u}$ are *literals* over U. If u is assigned a value 0, denoted by u=0, then $\bar{u}$ gets assigned a value 1, denoted by $\bar{u}$=1, and *vice-versa*. Similarly, u=1 if and only if $\bar{u}$=0. A *clause* over U is a set of literals over U. The *value of a clause* (also called OR-value) under a truth assignment is the *disjunction* (OR) of the values of its literals under that assignment. A clause is *satisfied* by a truth assignment if its value is 1 under that assignment. Hence, a clause is satisfied if and only if *at least one* of its literals is assigned a value 1 by the truth assignment. For example, $c_1 = \{u_1, \bar{u}_2, u_4\}$ is clause over $U = \{u_1, u_2, u_3, u_4\}$. The truth assignment u_1=0,u_2=1,u_3=0,u_4=1 satisfies the above clause c_1. However, the same truth assignment *does not satisfy* the clause $c_2 = \{u_1, \bar{u}_2, u_3\}$. The 3SAT problem can now be described as follows:

INSTANCE: A set $U = \{u_1, u_2, \ldots, u_n\}$ of Boolean variables and a collection $C = \{c_1, c_2, \ldots, c_m\}$ of clauses over U such that each clause has exactly 3 literals, i.e., $|c_i| = 3$ for all $1 \le i \le m$.

QUESTION: Is there a truth assignment for U that simultaneously satisfies all the clauses in C?

PARTITION

INSTANCE: A finite set A and a positive integer size $s(a)$ for each $a \in A$.

QUESTION: Is there a subset $A' \subset A$ such that

$$\sum_{a \in A'} s(a) = \sum_{a \in A-A'} s(a) \ ?$$

HITTING_SET

INSTANCE: A collection C of subsets of a finite set S and a positive integer $K \leq |S|$.

QUESTION: Is there a subset $S' \subseteq S$ with $|S'| \leq k$ such that S' contains at least one element from each subset in C ?

Further discussion and proofs of NP-Completeness of these problems can be found in [4,5,6].

Appendix B

Two new NP-Complete Problems

n-HITTING_SET (n-HS)

INSTANCE: A collection C of subsets of a finite set S and a positive integer $K \leq |S|$.

QUESTION: Is there a subset $S' \subseteq S$ with $|S'| \leq K$ such that S' contains at least n elements from each subset in C ?

It is easily seen that 1-HS is the same as the HITTING_SET problem described in Appendix A. As an example, let

$$S = \{s_1, s_2, s_3, s_4\}$$

$$C = \left\{ \{s_1, s_2, s_3\}, \{s_2, s_3, s_4\}, \{s_1, s_3, s_4\} \right\}$$

and integer K=3 be an instance of 1-HS. Clearly, the subset $S' = \{s_3\}$ is a solution to 1-HS. The same S, C, and K can be considered instances of 2-HS and 3-HS problems. For the 2-HS problem, the solution is the set $S' = \{s_1, s_2, s_3\}$, while this instance has no solution for the 3-HS problem. However, if we choose K=4, then $S' = S$ becomes a solution to the 3-HS problem.

Theorem B1: n-HS is NP_Complete for any fixed positive integer $n \geq 1$.

Proof: We use induction on n. The basis is the case n=1 for which n-HS becomes 1-HS (or simply the HITTING_SET problem) which is known to be NP_Complete [4].

The induction step consists of assuming that n-HS is NP_Complete for some value of $n \geq 1$, and proving that $(n+1)$-HS is also NP-Complete. Let an arbitrary instance of n-HS be given by a set $S = \{s_1, s_2, \ldots, s_p\}$, a collection $C = \{c_1, c_2, \ldots, c_q\}$ of subsets of S, i.e., $c_j \subseteq S$ for each $1 \leq j \leq q$, and a positive integer $K \leq |S| = p$. We will now construct an instance of $(n+1)$-HS given by a set $\hat{S}$, a collection $\hat{C}$ of subsets of $\hat{S}$, and a positive integer $\hat{K} \leq |\hat{S}|$ as follows:

Define a new set $X = \{x_1, x_2, \ldots, x_{n+1}\}$ disjoint from S.

Set $\hat{S} = S \cup X$. Hence, $|\hat{S}| = n+1+|S| = n+1+p$.

Define $\hat{c}_j = \{x_1\} \cup c_j$ for each $1 \le j \le q$.
Define $\hat{c}_{q+1} = X = \{x_1, x_2, \ldots, x_{n+1}\}$.

Set $\hat{C} = \{\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_{q+1}\}$. Clearly, $\hat{C}$ is a collection of subsets of $\hat{S}$.

Set $\hat{K} = n+1+K$. Note that $\hat{K} \le |\hat{S}|$.

Let $S' \subseteq S$ be a solution to the above instance of n-HS, i.e., $|S'| \le K$, and $|S' \cap c_j| \ge n$ for each $1 \le j \le q$. Define $\hat{S}' = S' \cup X$. Clearly, $\hat{S}' \subseteq \hat{S}$, $|\hat{S}'| = |S'| + n+1 \le K+n+1 = \hat{K}$. Also, $\hat{S}' \cap \hat{c}_j = \{x_1\} \cup (S' \cap c_j)$ for each $1 \le j \le q$, and $\hat{S}' \cap \hat{c}_{q+1} = X$. Hence, for each $1 \le j \le q+1$ we have $|\hat{S}' \cap \hat{c}_j| \ge n+1$, since X and S are disjoint, thus proving the $\hat{S}'$ is a solution to the constructed instance of $(n+1)$-HS. Conversely, let $\hat{S}' \subseteq \hat{S}$ be a solution to the constructed instance of $(n+1)$-HS, i.e., $|\hat{S}'| \le \hat{K}$, and $|\hat{S}' \cap \hat{c}_j| \ge n+1$ for each $1 \le j \le q+1$. In particular, when $\hat{c}_{q+1} = X$, we get $|X| = n+1 \ge |\hat{S}' \cap X| \ge n+1$, thereby establishing that $X \subseteq \hat{S}'$. Consequently, the set $S' = \hat{S}' - X$ is a solution to the considered instance of n-HS. □

3-MAX_FULL_SAT

In the classical 3SAT problem, defined in Appendix A, a clause c of 3 literals of a set U of Boolean variables is said to be satisfied by a truth assignment over U if and only if at least one of the literals in the clause c is assigned a value 1 (i.e., a true value). Thus, a clause is viewed as a disjunction (OR) of the values of its literals in 3SAT. In the new problem of 3-MAX_FULL_SAT, defined below, we say that a clause c of 3 literals is *fully-satisfied* by a truth assignment over U if and only if *all* its literals are assigned a true value of 1 by that assignment. Thus, a clause is viewed as a *conjunction* (AND) of the values of its literals in 3-MAX_FULL_SAT. We now define the new problem as follows:

INSTANCE: A set $U = \{u_1, u_2, \ldots, u_n\}$ of Boolean variables, a collection $C = \{c_1, c_2, \ldots, c_m\}$ of clauses over U such that each clause has exactly 3 literals, and a positive integer $k \le |C| = m$.

QUESTION: Is there a truth assignment for U that simultaneously fully-satisfies at least k clauses in C?

Theorem B2: 3-MAX_FULL_SAT is NP-Complete.

Proof: We reduce 3SAT to 3-MAX_FULL_SAT. Let $U = \{u_1, u_2, \ldots, u_n\}$ and $C = \{c_1, c_2, \ldots, c_m\}$ of 3-literal clauses over U denote an arbitrary instance of 3SAT. For convenience, suppose each clause $c_j = \{c_{j,1}, c_{j,2}, c_{j,3}\}$, where $c_{j,p}$ is a literal over U for each $1 \le p \le 3$, and $1 \le j \le m$. We need to construct an instance $\hat{U}$ of Boolean variables, $\hat{C}$ of 3-literal clauses over $\hat{U}$, and a positive integer k, such that there exists a truth assignment over U that simultaneously satisfies all clauses in C if and only if there is a truth assignment over $\hat{U}$ that simultaneously fully-satisfies at least k clauses of $\hat{C}$. The construction is as follows:

Define positive integer $k = m = |C|$, the number of clauses in C.

Introduce $2m$ new Boolean variables x_j and y_j, $1 \le j \le m$ disjoint from the variables in U, and form the set

$X = \{x_1, y_1, x_2, y_2, \ldots, x_m, y_m\}$.

Define the set $\hat{U} = U \cup X$. Note $|\hat{U}| = n+2m$.

For each clause $c_j = \{c_{j,1}, c_{j,2}, c_{j,3}\} \in C$ of literals over U, define three new clauses

$c_j^1 = \{c_{j,1}, x_j, y_j\}$

$c_j^2 = \{c_{j,2}, x_j, \bar{y}_j\}$

$c_j^3 = \{c_{j,3}, \bar{x}_j, y_j\}$

Define the set $\hat{C} = \bigcup_{j=1}^{m} \{c_j^1, c_j^2, c_j^3\}$ of 3-literal clauses over $\hat{U}$. Note $|\hat{C}| = 3m$.

We now make the following simple observations :

(1) For each $1 \le j \le m$, *no two* of the clauses c_j^1, c_j^2, c_j^3 in $\hat{C}$ can be simultaneously fully-satisfied by *any* truth assignment over $\hat{U}$.

(2) If a truth assignment for U satisfies a clause $c_j \in C$, then there is a truth assignment for the variables x_j and y_j such that *exactly one* of the clauses c_j^1, c_j^2, c_j^3 in $\hat{C}$ is fully-satisfied. For example, if a truth assignment sets $c_{j,1} = 1$ to satisfy the clause c_j, then set $x_j = y_j = 1$ to fully-satisfy the clause c_j^1. Note that the clauses c_j^2 and c_j^3 are not fully-satisfied by this truth assignment.

(3) If there is a truth assignment for $\hat{U}$ that fully-satisfies any one of the clauses c_j^1, c_j^2, c_j^3 in $\hat{C}$ for some $1 \le j \le m$, then the restriction of this truth assignment to U will satisfy the clause $c_j \in C$.

Now suppose that there is a truth assignment for U that satisfies each of the m clauses in C. By observation (2), we can extend this truth assignment to an assignment for $\hat{U}$ such that exactly $k=m$ clauses of $\hat{C}$ are fully-satisfied. Conversely, suppose a truth assignment for $\hat{U}$ fully-satisfies at least $k=m$ clauses of $\hat{C}$. By construction of $\hat{C}$ and by observation (1), this truth assignment actually fully-satisfies exactly $k=m$ clauses in $\hat{C}$. In fact, this truth assignment fully-satisfies exactly one of the clauses c_j^1, c_j^2, c_j^3 in $\hat{C}$, for each $1 \le j \le m$. Therefore, $c_j \in C$ is satisfied by observation (3) for each $1 \le j \le m$. Consequently, the truth assignment for $\hat{U}$ when restricted to U satisfies each clause in C. $\square$