Graph-Theoretic and Linear Optimization Approaches to Boolean Logic Minimization

Isaac Wicklund
Department of Mathematics and Statistics
Minnesota State University Mankato

June 3, 2026

Abstract

Boolean function minimization is central to modern digital logic design, circuit optimization, and VLSI synthesis. Classical techniques such as the Quine–McCluskey procedure systematically enumerate prime implicants, yet the subsequent selection of a minimum subset of implicants covering the ON-set is computationally difficult. This paper develops a unified framework combining graph theory, combinatorial optimization, and mathematical programming. Boolean functions are represented on the n-dimensional hypercube, where implicants correspond to subcubes and prime implicants correspond to maximal subcubes. The implicant selection problem is shown to be equivalent to a minimum set cover over these maximal subcubes. A complete integer linear programming (ILP) formulation is developed, along with its linear relaxation, which we solve using the simplex method. We prove the LP relaxation is integral and extended the framework to dual maxterm constructions as well as Boolean functions that include "don't-care" assignments.

Contents

1 Introduction

Boolean function minimization is a foundational problem in digital logic design. Given a Boolean function expressed over several variables, the objective is to construct a logically equivalent expression that uses the fewest possible terms or literals. Minimization reduces hardware cost, power consumption, delay, and improves manufacturability in combinational logic circuits.

Classical minimization techniques range from Karnaugh maps for small-variable functions and the Quine–McCluskey (QM) algorithm for systematic minimization. While QM enumerates all prime implicants, it does not resolve the subsequent task of selecting a smallest subset of those implicants that forms a complete cover of the function. This selection step is NP-hard and represents a central challenge in logic synthesis.

In this paper, we develop a geometric and optimization-based framework for understanding and solving the implicant selection problem. Boolean functions are represented as subsets of the n-dimensional hypercube, with implicants corresponding to subcubes and prime implicants to maximal subcubes. Under this interpretation, selecting a minimal set of prime implicants is equivalent to solving a minimum set cover problem over these maximal subcubes. We formalize this viewpoint by constructing an integer linear programming (ILP) formulation of the selection problem and analyzing its linear relaxation, which we solve using the simplex method. This unified formulation connects classical logical minimization with modern combinatorial optimization techniques and establishes a foundation for further generalizations explored in later sections.

All background material on Boolean functions, including Boolean algebra, function representation, and classical simplification methods, is taken from *Fundamentals of Logic Design* by Roth and kinny [2] and notes by Shirk [5]. Foundational concepts from graph theory and combinatorial optimization, are drawn from a graduate course in combinatorial optimization by Hollignsworth [4].

2 Background

2.1 Boolean Algebra

Definition 2.1 (Boolean Algebra)**.** A *Boolean Algebra* is on the set $B = \{0, 1\}$ with two binary operators $+$ ($\vee$, OR), $\cdot$ ($\wedge$, AND), and a unary operator $\sim$ ($\bar{}$, ', NOT, INV).

This algebraic system was first introduced by English mathematician/logician George Boole in his 1847 pamphlet *The Mathematical Analysis of Logic*. In 1898 Alfred North Whitehead published *Universal Algebra*, giving the first formal axiomization with seven axioms. In 1904, American Edward V. Huntington reduced the axiomization to 4. Although even more compact axiom systems now exist, Huntington's formulation is the most convenient for our purpose and is shown in Table 1.

Axiom	Duality	
	OR $+$	AND $\bullet$
Identity	$X + 0 = X$	$X \cdot 1 = X$
Commutativity	$X + Y = Y + X$	$X \cdot Y = Y \cdot X$
Distributivity	$X + (Y \cdot Z) = (X + Y)(X + Z)$	$X(Y + Z) = XY + XZ$
Complement	$X + \bar{X} = 1$	$X \cdot \bar{X} = 0$

Table 1: Huntington's four axioms.

We note that the axioms are self-dual, that is if we replace $+$ with $\cdot$ and 0 with 1 we obtain one of the other axioms.

2.2 Boolean Functions and the Hypercube

Definition 2.2 (Boolean Function)**.** A *Boolean function* in n variables is a mapping

$$f : \{0,1\}^n \to \{0,1\}.$$

A vector $x \in \{0,1\}^n$ is called a *minterm*.

Definition 2.3 (Graph)**.** A *graph* is a pair $G = (V, E)$ where V is a set of vertices, and E is a set of unordered pairs $\{v_i, v_j\}$ called edges.

Definition 2.4 (Boolean n-Cube)**.** The *n-dimensional hypercube*, denoted Q_n, is the graph with vertices

$$V(Q_n) = \{0,1\}^n$$

and edges between vertices that differ in exactly one coordinate. This coincides with the Hamming graph $H(n, 2)$.

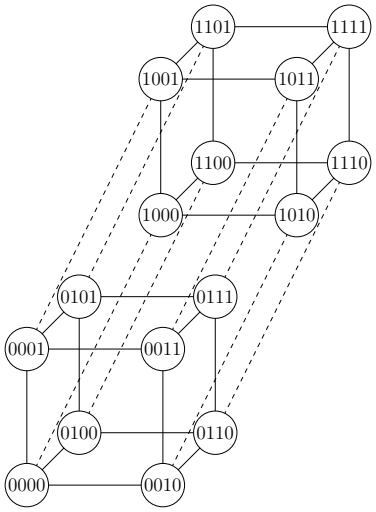


Figure 1: The 4D Boolean Hypercube Q_4.

This hypercube provides a natural geometric structure for Boolean functions: each minterm is a vertex, and adjacency corresponds to a single-bit change.

Definition 2.5 (Subcube)**.** A *subcube* of dimension k is a set of vertices obtained by fixing $n-k$ coordinates to constants and allowing the remaining k coordinates to vary freely. In algebraic terms, a subcube corresponds to a product term in which k literals are replaced by "don't-cares" symbols.

Definition 2.6 (Implicant)**.** A subcube $\mathcal{C}$ is an *implicant* of a boolean function f if

$$f(x) = 1 \text{ for all } x \in \mathcal{C}.$$

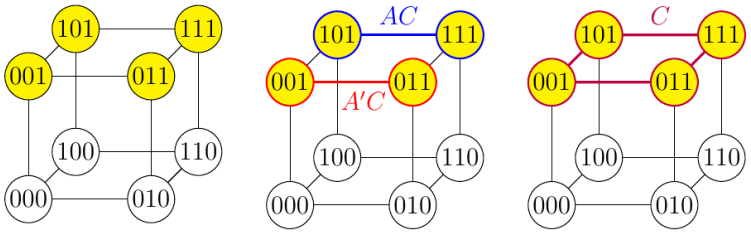
That is, it is fully contained in the ON-set of f that is the set of minterms for which $f = 1$.

Definition 2.7 (Prime Implicant)**.** An implicant $\mathcal{C}$ is a *prime implicant* if it is not properly contained in any larger implicant of f.

Prime implicants correspond exactly to *maximal subcubes* of Q_n contained in the ON-set.

m	A	B	C	$F(A,B,C)$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

(a) Truth table for $F(A,B,C)$.



(b) 3D Boolean cube with highlighted subcubes.

Figure 2: Illustration of maximal subcube generation.

2.3 Optimization Concepts

An *integer linear program (ILP)* is an optimization problem of the form

$$\max c^T x \quad \text{s.t.} \quad Ax \geq b, \quad x \in \mathbb{Z}^n.$$

Allowing real variables yields a *linear program (LP)*, which can be solved in polynomial time using methods such as the simplex algorithm or interior-point techniques. A maximization LP can be converted into a minimization LP by replacing the objective $C^T x$ with $-C^T x$.

3 Classical Approaches

The simplification of Boolean expressions is a fundamental task in the design of logical circuits. Each gate and literal introduced increases physical design costs such as circuit area, trace length, parasitic capacitance, resistance, and propogation delays. To illustrate the classical methods and their limitations, consider the following four variable system defined by its truth table.

m	A	B	C	D	$F(A,B,C,D)$
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	0

Table 2: Example truth table.

We begin with the worst-case implementation using a sum-of-minterms representations listing all input combinations for with the output is 1 as shown in Table 2.

$$\begin{aligned} F(A,B,C,D) &= \Sigma m(0,1,2,5,6,7,8,9,10,14) \\ &= A'B'C'D' + A'B'C'D + A'B'CD' \\ &+ A'BC'D + A'BCD' + A'BCD \\ &+ AB'C'D' + AB'C'D + AB'CD' \\ &+ ABCD' \end{aligned}$$

This expression requires ten 4-input AND gates feeding into a 10-input OR gate, an implementation that is both area intensive and slow due to large fan-in and resulting capacitences.

3.1 Algebraic Simplification

Using the Boolean axioms introduced earlier, we obtain several standard simplification rules:

Simplification Rule	Duality	
	OR +	AND •
Idempotent	$A + A = A$	$A \cdot A = A$
Absorption	$A + AB = A$	$A \cdot (A + B) = A$
DeMorgan's Law	$(A + B)' = A' \cdot B'$	$(A \cdot B)' = A' + B'$
Concencus	$AB + A'C + BC = AB + A'C$	$(A + B)(A' + C)(B + C) = (A + B)(A' + C)$
Combining	$AB + AB' = A$	$(A + B) \cdot (A + B') = A$
Elimination	$A + A'B = A + B$	$A \cdot (A' + B) = AB$
Involution	$(A')' = A$	

Table 3: Useful Boolean simplification rules.

These rules enable systematic algebraic reduction of the sum-of-minterms expression:

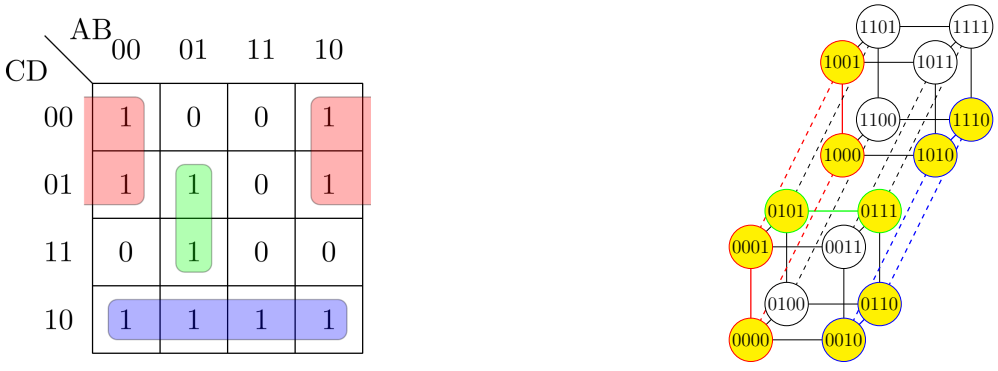
$$
\begin{aligned}
F(A,B,C,D) =& A'B'C'D' + A'B'C'D + AB'C'D' + AB'C'D \\
&+ A'B'CD' + A'BCD' + AB'CD' + AB'CD' + ABCD' \\
&+ A'BC'D + A'BCD \\
=& A'B'C'(D' + D) + AB'C'(D' + D) \\
&+ A'CD'(B' + B) + ACD'(B' + B) && \text{(Distributivity)} \\
&+ A'BD(C' + C) \\
=& A'B'C' + AB'C' \\
&+ A'CD' + ACD' && \text{(Complement)} \\
&+ A'BD \\
=& B'C'(A' + A) + CD'(A' + A) + A'BD && \text{(Distributivity)} \\
=& B'C' + CD' + A'BD. && \text{(Complement)}
\end{aligned}
$$

Thus a fully simplified expression is obtained using just two 2-input AND gates, one 3-input AND gate, and a 3-input OR gate. Although effective, this algebraic method requires careful identification of adjacencies and term groupings, often relying on intuition or experience. This motivates the development of approaches that avoid such reliance on manual insight.

3.2 Karnaugh Maps

The first simplification technique we examine is the Karnaugh map (k-map), introduced in 1953 by Maurice Karnaugh. A K-map arranges the functions truth table into a two-dimensional grid whose rows and columns are ordered by Gray code so that adjacent cells differ in exactly one variable. This mirrors the adjacency structure of the Boolean hypercube.

A K-map allows us to identify groups of 2^k adjacent 1-cells. Each grouping corresponds to an implicant in which k variables become "don't-care" terms, producing a simpler product expression



(a) (b)

Figure 3: Comparison of the K-map (a) and corresponding 4D hypercube representation (b).

While K-maps provide an intuitive and visually guided minimization procedure, they still rely heavily on pattern recognition and user insight. Moreover, their scalability is limited: K-maps for more than four variables become increasing difficult to interpret, and for six variables they are impracticably large. These limitations motivate the needs for more systematic and algorithmic methods.

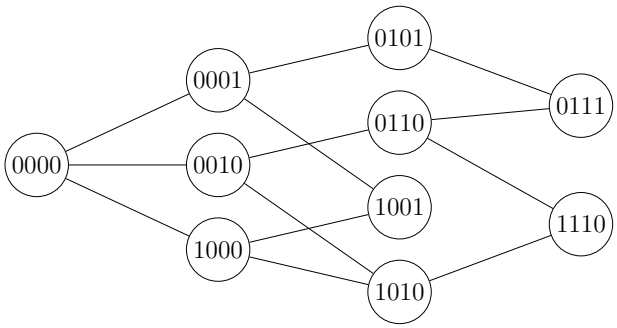
3.3 Quine-McClusky Algorithm

The limitations of algebraic and map-based simplification motivate the use of a fully systematic technique. Developed by Willard V. Quine in 1952 and later refined by Edward J. McCluskey, the Quine–McCluskey (QM) algorithm provides a complete tabular procedure for identifying all prime implicants of a Boolean function. The algorithm mirrors the logic of K-maps, buts its structured, combinational approach makes it far better suited for automation and larger input sizes. It proceeds in two phases, first an iterative combination followed by a construction of the prime implicant table.

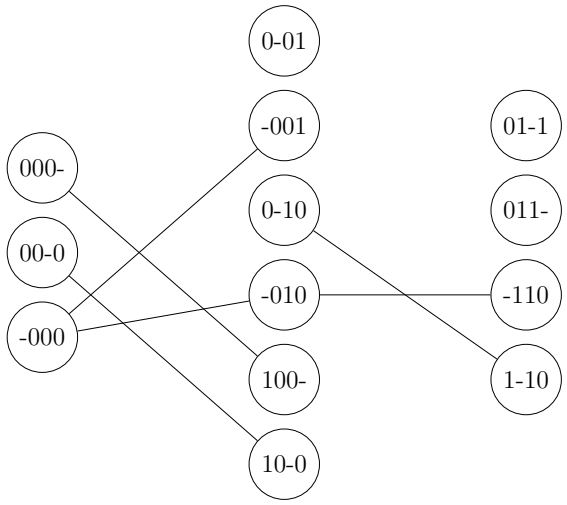
3.3.1 Iterative Combination

In the first phase, minterms are grouped by Hamming weight and combined whenever they differ in exactly one bit position. Each successful combination introduces a "don't-care" symbol in the differing position, producing a higher dimensional implicant. This process repeats until no further merging is possible. The tables and corresponding partite graphs below illustrate these combination steps for our example.

Group	Term	Binary	Used?
0	0	0000	✓
1	1	0001	✓
	2	0010	✓
	8	1000	✓
2	5	0101	✓
	6	0110	✓
	9	1001	✓
	10	1010	✓
3	7	0111	✓
	14	1110	✓



Group	Term	Binary	Used?
0	0, 1	000-	✓
	0,2	00-0	✓
	0, 8	-000	✓
1	1, 5	0-01	**X**
	1, 9	-001	✓
	2, 6	0-10	✓
	2, 10	-010	✓
	8, 9	100-	✓
	8, 10	10-0	✓
2	5, 7	01-1	**X**
	6, 7	011-	**X**
	6, 14	-110	✓
	10, 14	1-10	✓



Group	Term	Binary	Used?
0	0,1,8,9	-00-	**X**
	0,2,8,10	-0-0	**X**
1	1, 5	0-01	**X**
	2,6,10,14	--10	**X**
2	5, 7	01-1	**X**
	6, 7	011-	**X**

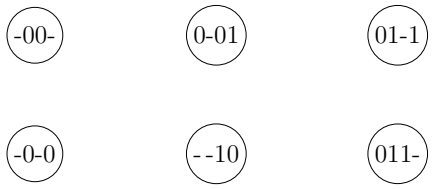


Figure 4: Three Stages of the QM Process with Associated Partite Graphs

Lemma 3.1. *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two subcubes of Q_n that are implicants of f (i.e. every vertex in each $\mathcal{C}_i$ lies in the ON-set of f). If $\mathcal{C}_1$ and $\mathcal{C}_2$ differ in exactly one fixed coordinate (equivalently, their pattern representations differ in exactly one position, with all other fixed positions identical), then the combined subcube $\mathcal{C}$ obtained by replacing that differing coordinate with a "don't-care" is also an implicant of f.*

Proof. Write the two subcubes in pattern form (using $0, 1$, and '$-$' for don't-care). By hypothesis there exists an index j such that the jth coordinate of every element of $\mathcal{C}_1$ is 0,

the jth coordinate of every element of $\mathcal{C}_2$ is 1, and for every coordinate $i \neq j$ the fixed values (or don't-cares) agree between $\mathcal{C}_1$ and $\mathcal{C}_2$. Let $\mathcal{C}$ denote the subcube obtained by placing a don't-care '$-$' in coordinate j while keeping the other coordinates the same.

Every vertex $x \in \mathcal{C}$ has the property that its jth bit is either 0 or 1. If $x_j = 0$ then $x \in \mathcal{C}_1$, and if $x_j = 1$ then $x \in \mathcal{C}_2$. Since $\mathcal{C}_1$ and $\mathcal{C}_2$ are implicants, $f(x) = 1$ for every $x \in \mathcal{C}_1 \cup \mathcal{C}_2$. Therefore $f(x) = 1$ for every $x \in \mathcal{C}$, and so $\mathcal{C}$ is an implicant. □

Theorem 3.1. *The iterative combination phase of the Quine–McCluskey algorithm (1) produces every prime implicant of f, and (2) outputs only prime implicants (i.e. it does not output any implicant that is properly contained in a strictly larger implicant).*

Proof. We prove the two claims separately.

(i) Every prime implicant is produced. Let $\mathcal{P}$ be any prime implicant of f. Represent $\mathcal{P}$ in pattern form: it has some coordinates fixed to 0 or 1 and r coordinates marked '$-$' (don't-cares). Thus $\mathcal{P}$ is a r-dimensional subcube and contains exactly 2^r minterms (vertices) of $\{0,1\}^n$.

Choose any two minterms $m, m' \in \mathcal{P}$ that differ in exactly one of the r free coordinates; such a pair exists whenever $r \geq 1$. Starting from the initial QM grouping, QM's combination rule will merge m and m' into the adjacent subcube where that coordinate becomes a '$-$'. Repeating this reasoning, pairs of subcubes produced at round t that lie inside $\mathcal{P}$ and differ in one additional free coordinate will be merged at round $t+1$. By induction on the number of '$-$' positions, after r rounds the algorithm produces the full pattern of $\mathcal{P}$ (the subcube having those r don't-cares). If $r = 0$ (a single minterm), that minterm is present initially. Hence every prime implicant $\mathcal{P}$ is generated by repeated combinations beginning from the minterms contained in $\mathcal{P}$.

(ii) Only prime implicants are output. QM marks as *combined* any implicant that is merged further in a subsequent round; only those implicants that are never combined again are retained as final implicants. Suppose, for contradiction, that an implicant $\mathcal{C}$ that the algorithm outputs is not prime. Then there exists a strictly larger implicant $\mathcal{C}^*$ with $\mathcal{C} \subsetneq \mathcal{C}^*$ (i.e. $\mathcal{C}^*$ has at least one additional '$-$' position compared to $\mathcal{C}$). By construction of subcubes, there must be another implicant $\mathcal{C}'$ produced in the combination process which differs from $\mathcal{C}$ in exactly one coordinate and whose combination with $\mathcal{C}$ yields (or is contained in) $\mathcal{C}^*$. But if such a $\mathcal{C}'$ exists then $\mathcal{C}$ would have been combined with $\mathcal{C}'$ during the algorithm and thus would not appear among the final, uncombined implicants. This contradiction shows that every implicant output by QM is maximal, i.e. prime.

Combining (i) and (ii) completes the proof: the algorithm generates every prime implicant and the final list contains no non-prime implicants. □

3.3.2 Prime Implicant Table

Once the full set of prime implicants has been generated, the algorithm constructs a chart indicating which implicants cover which minterms. Essential prime implicants, those that uniquely cover at least one minterm are immediately selected. The remaining minterms must

be selected by choosing a subset of the non-essential prime implicants. For small functions this selection may be performed by brute force, or a heuristic strategy selecting implicants which cover the most minterms can be employed for larger functions.

Term	Binary	Prime Implicant	0	1	2	5	6	7	8	9	10	14
0,1,8,9	-00-	B'C'	x	x					x	x		
2,6,10,14	--10	CD'			x		x				x	x
0,2,8,10	-0-0	B'D'	x		x				x		x	
1, 5	0-01	A'C'D		x		x						
5, 7	01-1	A'BD				x		x				
6, 7	011-	A'BC					x	x				

Table 4: Implicant Selection Table from QM

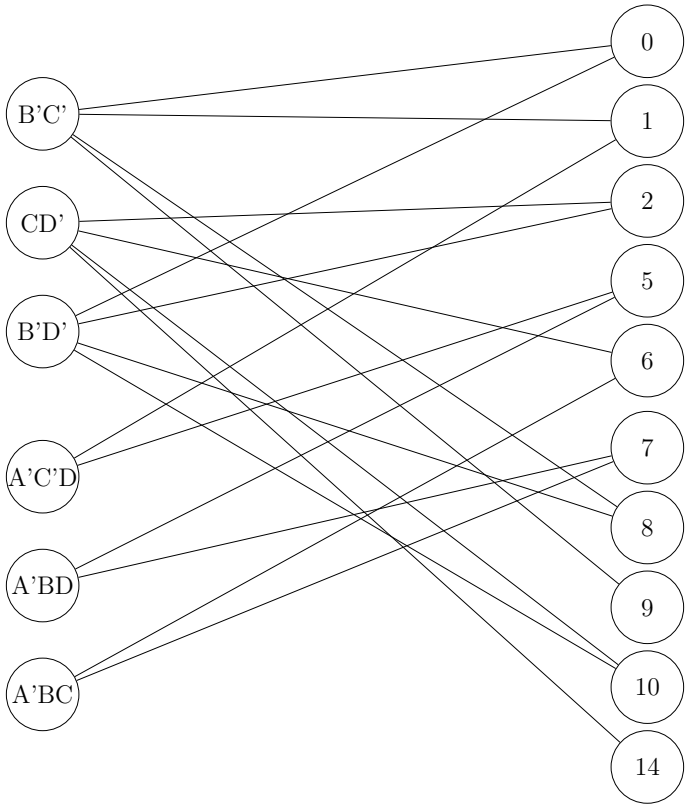


Figure 5: Bipartite Graph Structure Associate with Implicant Table

4 Graph-Theoretic Perspective

4.1 Subcubes as Induced Subgraphs

Definition 4.1 (Induced Subgraph)**.** Given a graph $G = (V, E)$ and a subset $V' \subseteq V$, the *induced subgraph* on V' is the graph with vertex set V' and all edges of G with both endpoints in V'.

Every implicant corresponds to a subcube of Q_n, which is an induced subgraph where $f = 1$. Prime implicants correspond to maximal subgraphs of this form.

Definition 4.2 (Maximal Subcube)**.** A subcube $\mathcal{C}$ is *maximal* if it is an implicant and is not contained within a strictly larger subcube that is also an implicant.

4.2 Minimal Cover Problem

Let $U \subseteq \{0,1\}^n$ denote the ON-set of the Boolean function f. Let $\mathcal{I} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ be the set of prime implicants of f.

Definition 4.3 (Covering)**.** A set of implicants $C \subseteq \mathcal{I}$ is said to *cover* U if $\bigcup_{\mathcal{C} \in C} \mathcal{C} = U$.

Theorem 4.1. *Selecting a minimum subset of prime implicants covering U is equivalent to the minimum set cover problem.*

Proof. For each implicant $\mathcal{C}_i$, create a set $S_i \subseteq U$ consisting of all minterms it covers. A valid logic cover corresponds to selecting sets whose union is U. Minimizing the number of selected implicants is exactly minimum set cover. □

Since set cover is NP-hard, the implicant selection problem inherits this hardness.

5 Linear Programming Formulation

5.1 ILP Model

Let $x_i \in \{0,1\}$ be a binary decision variable indicating whether prime implicant $\mathcal{C}_i$ is selected. Construct an adjacency matrix $A \in \{0,1\}^{|U| \times |\mathcal{I}|}$ where

$$A_{ij} = \begin{cases} 1 & \text{if minterm } u_i \text{ is covered by } \mathcal{C}_j, \\ 0 & \text{otherwise.} \end{cases}$$

The integer linear program (ILP) is

$$\min \mathbf{1}^T x \quad \text{s.t.} \quad Ax \geq \mathbf{1}, \quad x \in \{0,1\}^{|\mathcal{I}|},$$

where $\mathbf{1}$ is a vector of ones.

5.2 LP Relaxation

Relaxing the integrality constraint to $x \in [0,1]^{|\mathcal{I}|}$ gives the linear program

$$\min \mathbf{1}^T x \quad \text{s.t.} \quad Ax \geq \mathbf{1}, \quad x \in [0,1]^{|\mathcal{I}|}.$$

Although optimal solutions for a LP relaxation may in general be fractional, the LP yields a lower bound on the minimum number of prime implicants required to cover the ON-set.

5.3 Simplex Method

The simplex method is used to solve the LP, which it does so efficiently. In the following section, we show that due to the structure of A, this LP always yields integer optimal solutions, making the relaxation exact for this problem.

6 Integrality of the LP Solution

LP relaxations of integer problems often yield fractional solutions which give a bound on the true solution and require rounding heuristics to find integral solutions. However, LPs whose feasible region is the polyhedron $P = \{x : Ax \leq b\}$ with only integral vertices, the optimal solution will be integral. Proving integrality of P can be quite challenging; however, certain conditions on A, like being balanced, totally unimodular, or ideal can show integrality. We focus on showing that the LP is totally dual integral.

Definition 6.1 (Totally Dual Integral)**.** A linear system $Ax \leq b$ with rational A and b is said to be *totally dual integral* (TDI) if for any integral c, the linear program

$$\max c^T x \quad \text{s.t.} \quad Ax \leq b, x \geq 0$$

has an optimal solution, the corresponding dual linear program

$$\min b^T y \quad \text{s.t.} \quad A^T y \geq c, y \geq 0$$

admits an integer optimal solution.

We note that since A and b need only be rational that this definition can be augmented to work for a minimization primal.

Theorem 6.1. *If a polyhedron P is the solution set of a TDI system $Ax \leq b$, where b is integral, then every vertex of P is integer valued.*

Proof. (see [3]) □

Theorem 6.2. *Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be the family of maximal axis-aligned subcubes of the ON-set $U \subseteq \{0,1\}^n$. Let $A \in \{0,1\}^{|U| \times m}$ be the incidence matrix defined by $A_{u,i} = 1$ iff $u \in C_i$. Then for every integer vector $b \in \mathbb{Z}^{|U|}$ the system*

$$\{x \geq 0 : Ax \geq b\}$$

is integral; equivalently, the system is TDI.

Proof. Consider the primal LP

$$\text{(P)} \qquad \min 1^\top x \quad \text{s.t.} \quad Ax \geq b,\ x \geq 0,$$

with one variable x_i for each subcube C_i. Its dual is

$$\text{(D)} \qquad \max b^\top y \quad \text{s.t.} \quad A^\top y \leq 1,\ y \geq 0.$$

We prove that (D) always has an integral optimal solution when b is integral. This implies that the system $Ax \geq b$ is TDI.

Interpretation of the dual. The constraint matrix $A^\top$ is the incidence matrix of a bipartite graph with one partition of minterms (call it U) and on partition of subcubes (say $\mathcal{C}$) A minterm u is adjacent to a subcube C_i exactly when $u \in C_i$.

The dual variables y_u assign a nonnegative weight to each minterm. The constraint

$$\sum_{u \in C_i} y_u \leq 1 \qquad (i = 1, \ldots, m)$$

says that each subcube has capacity 1, and the total weight of the minterms belonging to that subcube cannot exceed this capacity.

Therefore (D) is a *unit-capacity bipartite packing problem*: we assign weights to minterms so that no subcube receives total weight more than 1, and we maximize the profit $b^\top y$.

Total unimodularity. The constraint matrix of a bipartite packing problem is a 0-1 incidence matrix of a bipartite graph, and such matrices are totally unimodular (see [1, Thm. 19.3]). Hence every feasible extreme point of (D) is integral. In particular, (D) has an integral optimal solution for every integral b.

Conclusion. Since the dual has an integral optimal solution for every integral right-hand side, the primal system $\{x \geq 0 : Ax \geq b\}$ is TDI. By Theorem 6.1, the feasible region has integral vertices whenever b is integral. □

Thus, the LP relaxation of our prime implicant selection ILP always yields an integral solution. Consequently, the LP provides not just a lower bound, but an exact solution to the minimum implicant cover problem.

7 Example

We illustrate the framework for the Boolean function

$$f(A, B, C, D, E) = \sum m(1, 2, \cdots, 29, 30).$$

This function was selected due to it having no essential prime implicants and all the implicants have significant overlap increasing the computational complexity of the result minimum cover problem.

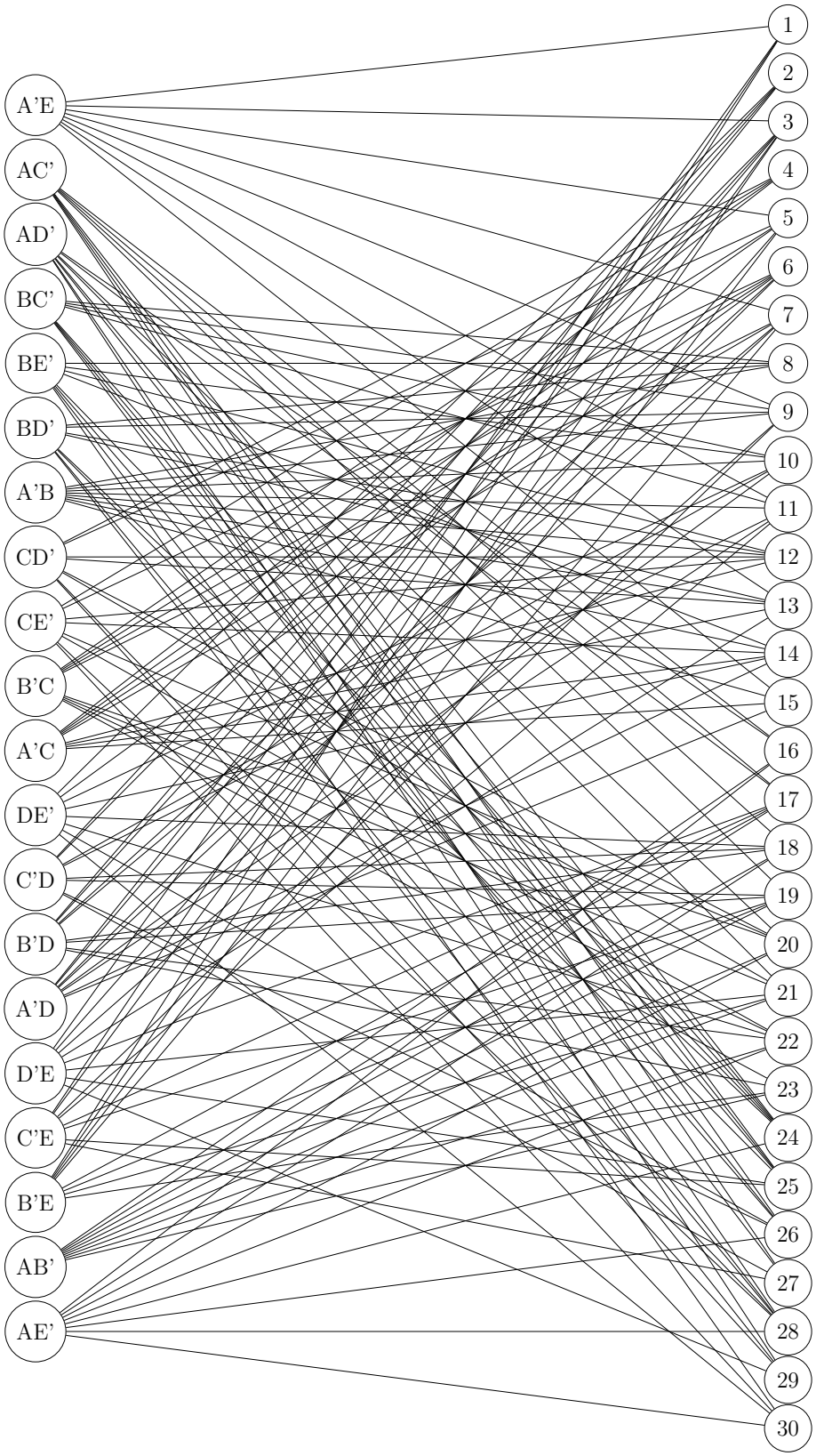
7.1 Prime Implicant Generation

Applying the QM combination procedure yields the following prime implicants.

Prime Implicant	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
A'E	x		x		x		x		x		x		x		x															
AC'																x	x	x	x					x	x	x	x			
AD'																x	x			x	x			x	x			x	x	
BC'								x	x	x	x													x	x	x	x			
BE'								x		x		x		x										x		x		x		x
BD'								x	x			x	x											x	x			x	x	
A'B								x	x	x	x	x	x	x	x															
CD'				x	x							x	x							x	x							x	x	
CE'				x		x						x		x						x		x						x		x
B'C				x	x	x	x													x	x	x	x							
A'C				x	x	x	x					x	x	x	x															
DE'		x				x				x				x				x				x				x				x
C'D		x	x							x	x							x	x							x	x			
B'D		x	x			x	x											x	x			x	x							
A'D		x	x			x	x			x	x			x	x															
D'E	x				x				x				x				x				x				x				x	
C'E	x		x						x		x						x		x						x		x			
B'E	x		x		x		x										x		x		x		x							
AB'																x	x	x	x	x	x	x	x							
AE'																x		x		x		x		x		x		x		x

7.2 Graph-Theoretic View

From the result of QM, we plot the resulting bipartite graph, with prime implicants on the left and the minterms they cover on the right.


A'E
AC'
AD'
BC'
BE'
BD'
A'B
CD'
CE'
B'C
A'C
DE'
C'D
B'D
A'D
D'E
C'E
B'E
AB'
AE'
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

7.3 ILP and LP Solutions

From the bipartite graph, or more directly from the result of QM we form the adjacency matrix to be used in the LP.

$$A = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

We construct our LP as,

$$\min \mathbf{1}^T x$$

Such that,

$$\begin{aligned} Ax &\geq 1 \\ x &\geq 0. \end{aligned}$$

After running simplex, our optimal solution is returned to us in 0.03 seconds as

$$x = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

and the optimal value is 5. This solution informs us that a minimal function satisfying the selected minterms is given by

$$f(A, B, C, D, E) = BC' + CE' + A'D + D'E + AB'.$$

Applying the two other methods described earlier, the greedy algorithm failed to find an optimal solution and instead found solutions consisting of 6 minterms, the brute force method was computationally infeasible, with the program crashing due to insufficient memory.

8 Expansion

The results presented for the ON-set of a Boolean function extend naturally to the OFF-set. Specifically, the maxterm representation of a function corresponds to the OFF-set, which is simply the complement of the ON-set. All structural results, including the LP formulation and integrality properties, hold under this complement, with the roles of zeros and ones interchanged.

Additionally, the inclusion of "don't-care" terms in the function definition can be accommodated. In this case, the decision variable vector x and the incidence matrix A remain unchanged, but the cost vector c can be adjusted to carry weights corresponding to the lengths (number of literals) of the chosen implicants. This allows the LP formulation to minimize weighted sums, reflecting the relative complexity of implicants rather than simply counting them.

9 Conclusion

In this work, we explored the simplification of Boolean functions from classical algebraic methods to modern optimization-based approaches. We demonstrated how sum-of-minterms and algebraic simplification, while straightforward, often require extensive manual insight, motivating the use of Karnaugh maps and the Quine–McCluskey algorithm for systematic reduction.

By framing the problem in a graph-theoretic perspective, we identified prime implicants as maximal subcubes of the hypercube and formalized the minimal implicant selection problem as an instance of the set cover problem. Translating this to a linear programming framework allowed us to leverage total dual integrality to guarantee integral optimal solutions, providing both theoretical insight and practical computational efficiency.

Finally, we highlighted that these results naturally extend to maxterm representations and functions with "don't-care" terms, with appropriate weighting to account for implicant complexity. This unified framework provides a robust method for minimizing Boolean functions and offers a foundation for future extensions to more complex digital logic optimization problems.

References

[1] A. Schrijver, Theory of Linear and Integer Programming. Chichester U.K.: Wiley, 1999.

[2] C. H. Roth and L. L. Kinney, *Fundamentals of Logic Design*. Stamford, CT: Cengage Learning, 2014.

[3] J. Edmonds and R. Giles, "A min-max relation for submodular functions on graphs," Annals of Discrete Mathematics, pp. 185–204, 1977. doi:10.1016/s0167-5060(08)70734-9

[4] K. Hollingsworth, Class Lecture, Topic: "Graphs and Algorithms." MATH605, College of Science Engineering and Technology, Minnesota State University, Mankato, 2025

[5] R. Shirk, Class Lecture, Topic: "Introduction to Digital Logic." EE106, College of Science Engineering and Technology, Minnesota State University, Mankato, 2022