Technology Mapping
The Directed Acyclic Graph-Covering Problem
and
The Tree-Covering Problem

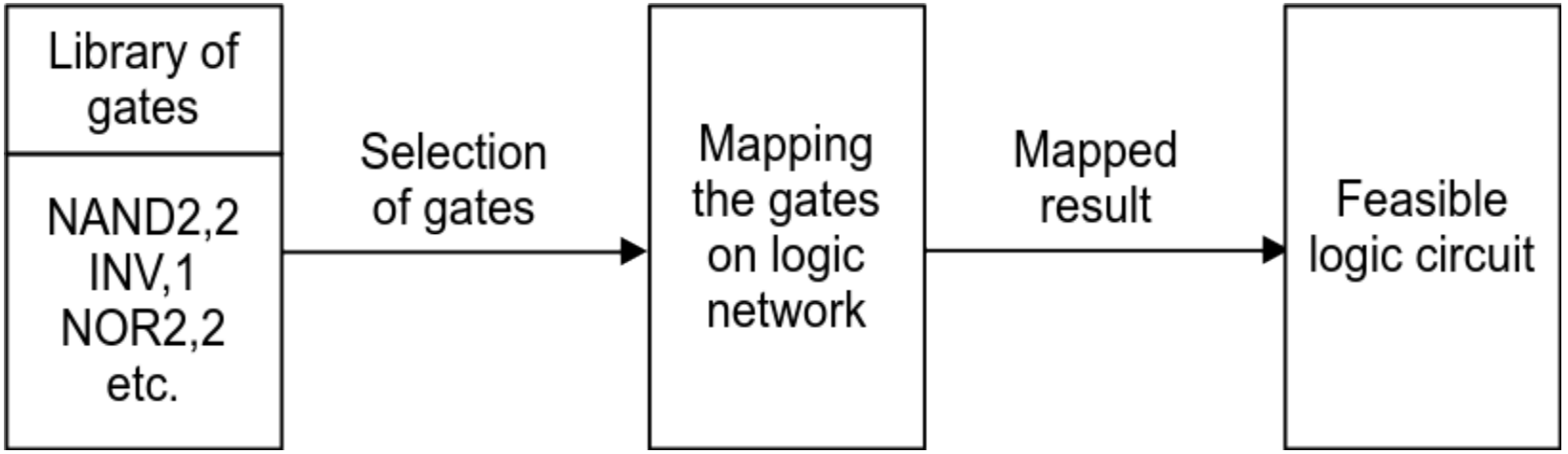
Presented by:
Rashed Hassan Siam
Class Roll: FH-2343
Registration No: 2022-518-433
Department of Computer Science and Engineering
University of Dhaka

Presented to:
Professor Dr. Hafiz Md. Hasan Babu
Department of Computer Science and Engineering
University of Dhaka

Contents

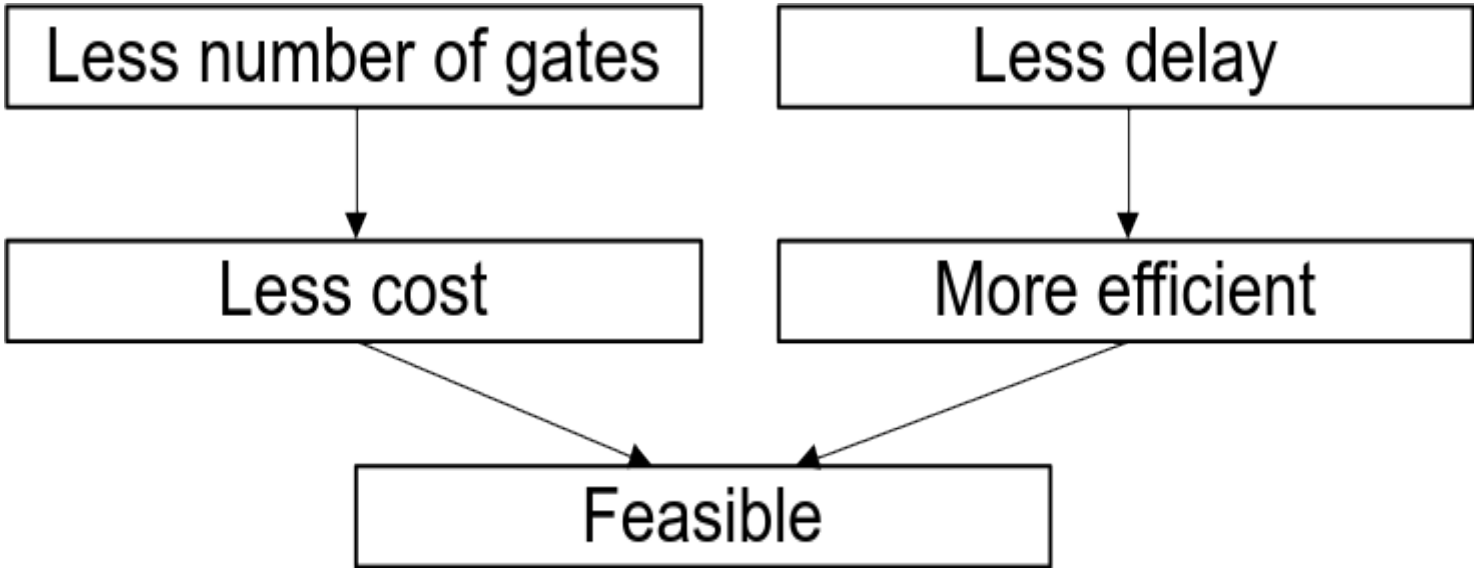
Technology Mapping

- Helps to implement a logic circuit.
- Performs the final gate selection from a particular library.
- The library is also known as the Technology Library.
  - Consists of several gates (NAND2,2; NOR2,2; INV,1 etc.).

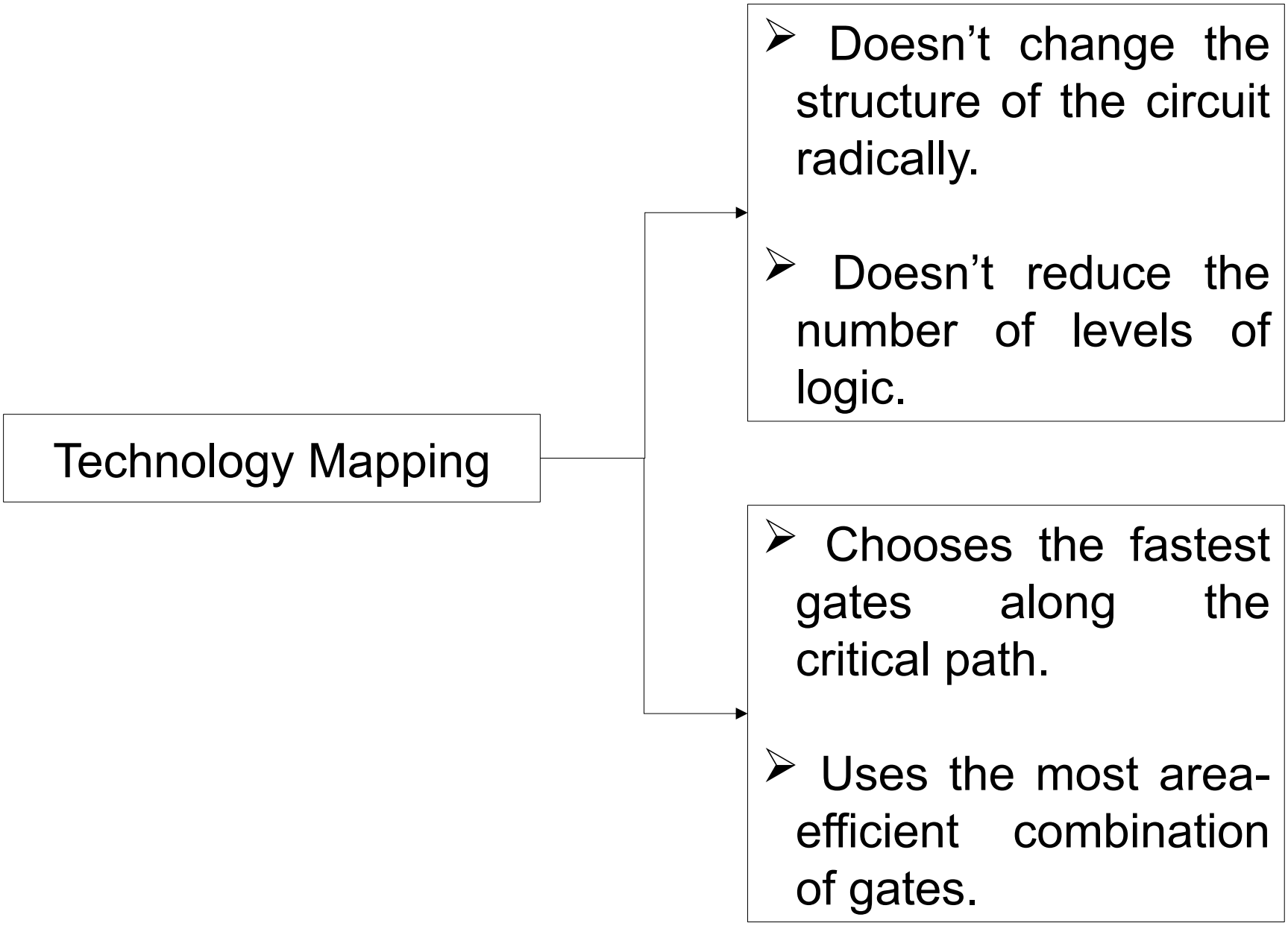


Feasible Circuits

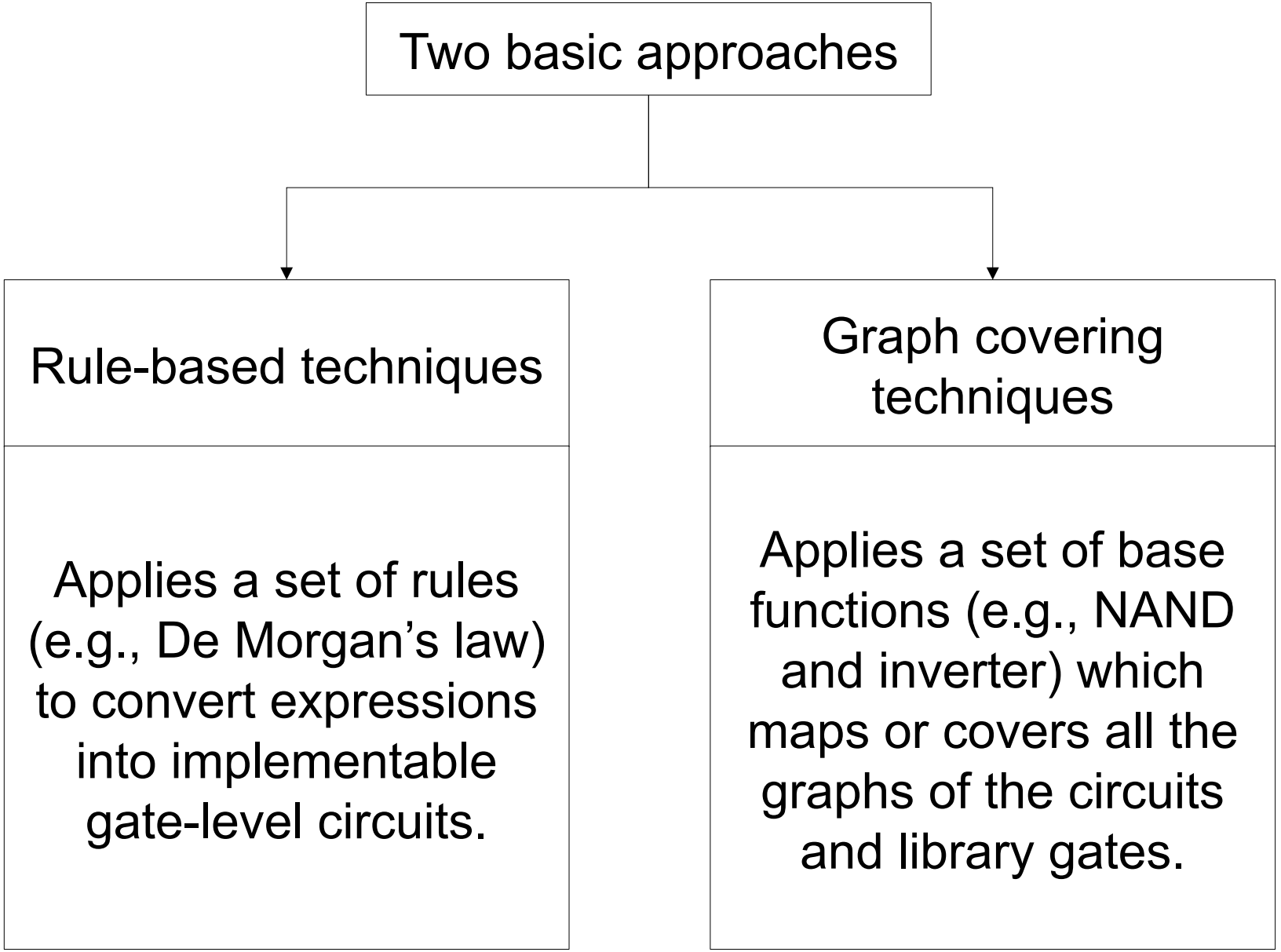
- Optimal with respect to area.
  - Least number of nodes/gates.
- Satisfies a maximum critical-path delay.
  - The longest path of the circuit from input to output.
  - Should be within a specified threshold.



Dos and Don’ts of Technology Mapping



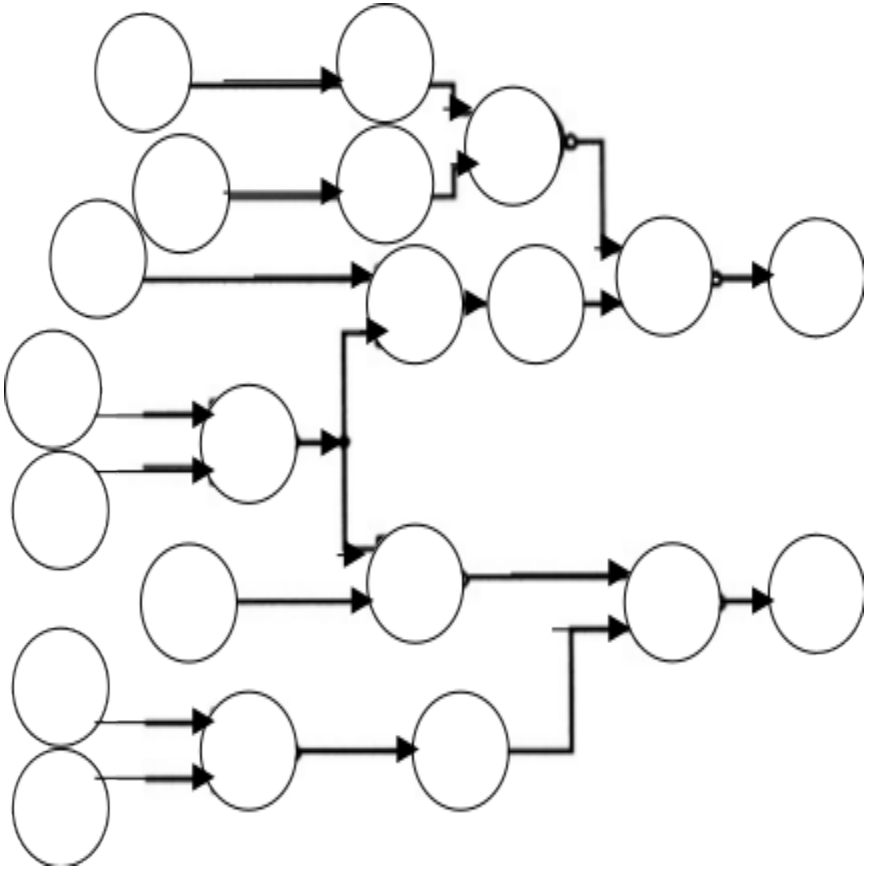
Technology Mapping Approaches



Directed Acyclic Graph

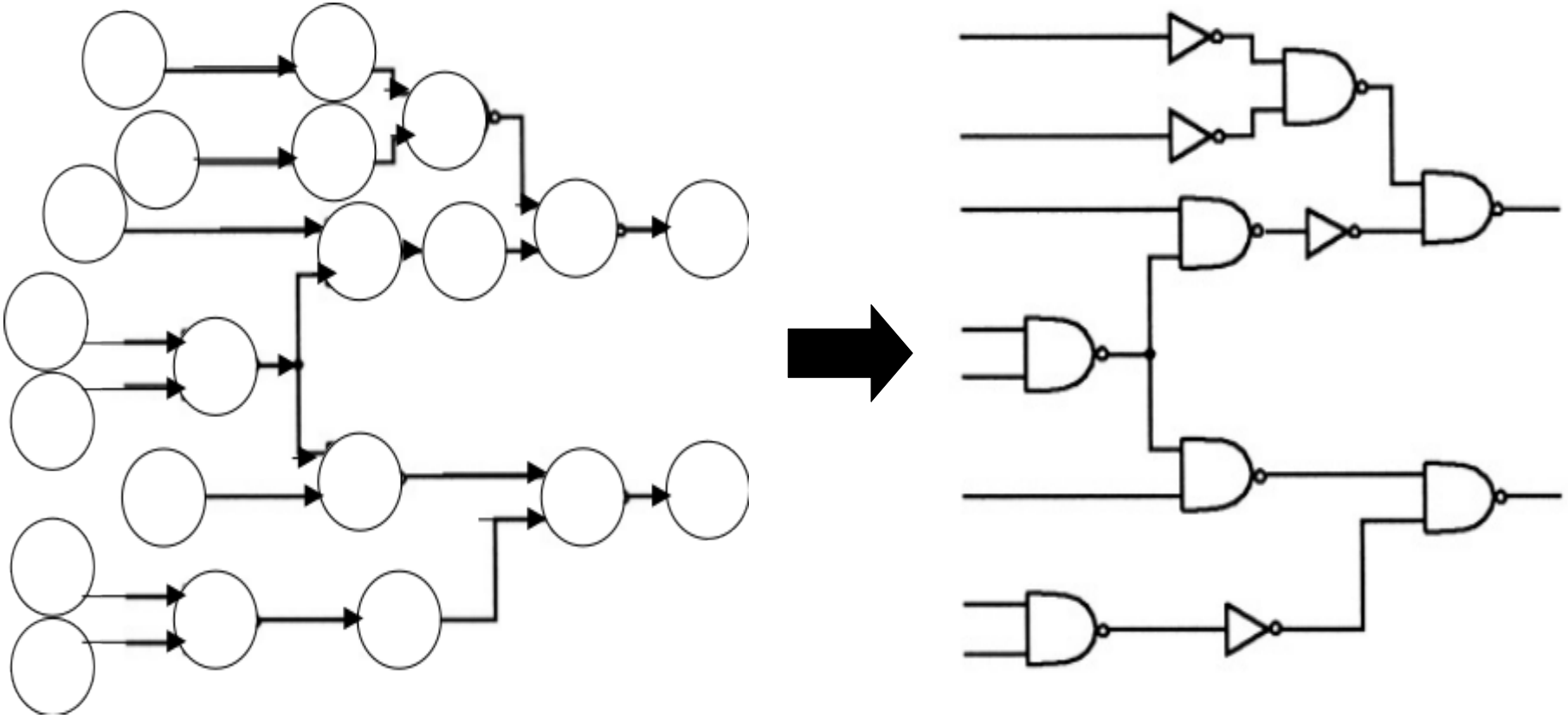
- A graph made of nodes connected by directed edges.
- There are no cycles.
  - We can't start at a node and return to it by following the edges.
- In short, it is also called DAG.

A directed acyclic graph, where the circles are nodes and arrows are the directed edges.



Logic Network in DAG Form

- If we consider the previous DAG example, and replace its nodes with various gates, inputs and outputs, we get:



So, combinational logic networks are basically DAGs!

Prerequisites of the DAG-Covering Problem

Base Functions	Subject Graph	Pattern Graph
➢ A set of chosen functions (e.g., a two-input NAND-gate and an inverter). ➢ Used to design the graphs of the logic function and the library gates.	➢ The simplified Boolean network is converted into the subject graph. ➢ Each node is restricted to one of the base functions.	➢ The logic function for each library gate is converted into a pattern graph. ➢ Each node is restricted to one of the base functions.

Goals of the DAG-Covering Problem

- To find a “minimum cost covering” of the subject graph by choosing appropriate collection of pattern graphs.
  - Covering means, every node of the subject graph is contained in one (or more) of the pattern graphs.
  - Minimum cost covering means:
    - Optimizing area of the cover.
    - Optimizing minimum delay of the cover.

Choice of Base Functions

- The choice is arbitrary as long as the base function set is functionally complete.
  - A functionally complete function can express any other function.
  - Such as, two-input NANDs and inverters.

- Different combinations of base functions produce different patterns of graphs.

$$f = (abcd + efgh + ijkl + mnop)'$$

- Base function set with:
  - Two-input, three-input, and four-input NAND-gates: 1 pattern.
  - Two-input NAND-gates and inverters: 18 patterns.

Creating the Subject and Pattern Graphs

- The logic function may have many combinations of graphs.
  - Happens due to the various possible combinations of the base function set.

- Each representation is a potential subject graph for DAG-covering.
  - Every one of these subject graphs (starting points) should be considered for an optimum covering.

- Similarly, many combinations of pattern graphs may also be generated.
  - Further increases the complexity of the DAG-covering problem.

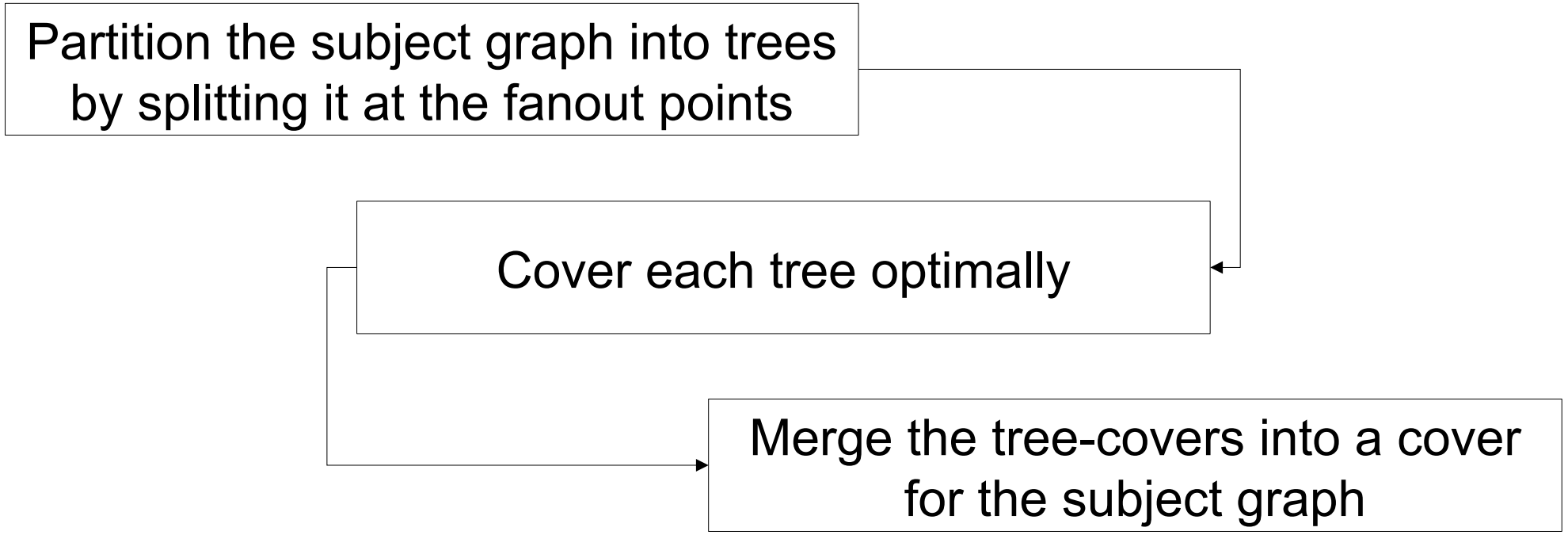
The DAG-Covering Problem Complexity

- DAG-covering-by-DAGs is NP-hard!
  - NP-hard means it's not solvable in polynomial time.
  - Even with only three pattern graphs (inverter, two-input NAND, two-input NOR), and
  - Each subject graph node having no more than two incoming and outgoing edges.

- Heuristics can be a more effective approach.
  - But this is still an open problem.
  - L. Lavagno at U.C. Berkeley has achieved some degree of success.

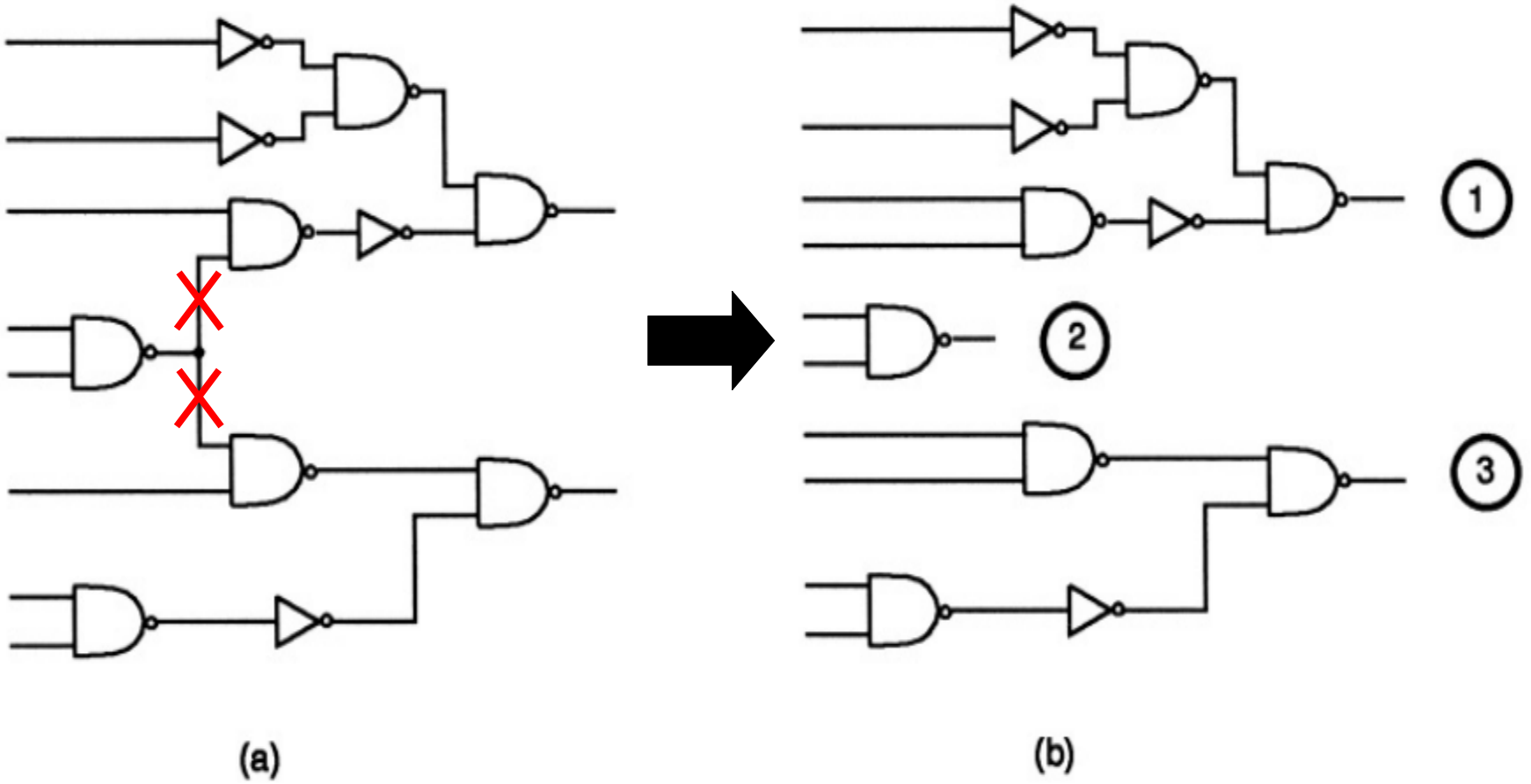
Alternative Approach: The Tree-Covering Problem

- Reducing the DAG-covering problem to a set of tree-covering-by-trees problems.
  - Tree circuit is a single output circuit.
  - Each gate except the output, feeds exactly one other gate.
  - No fanout/branching at the outputs of any of the gates.



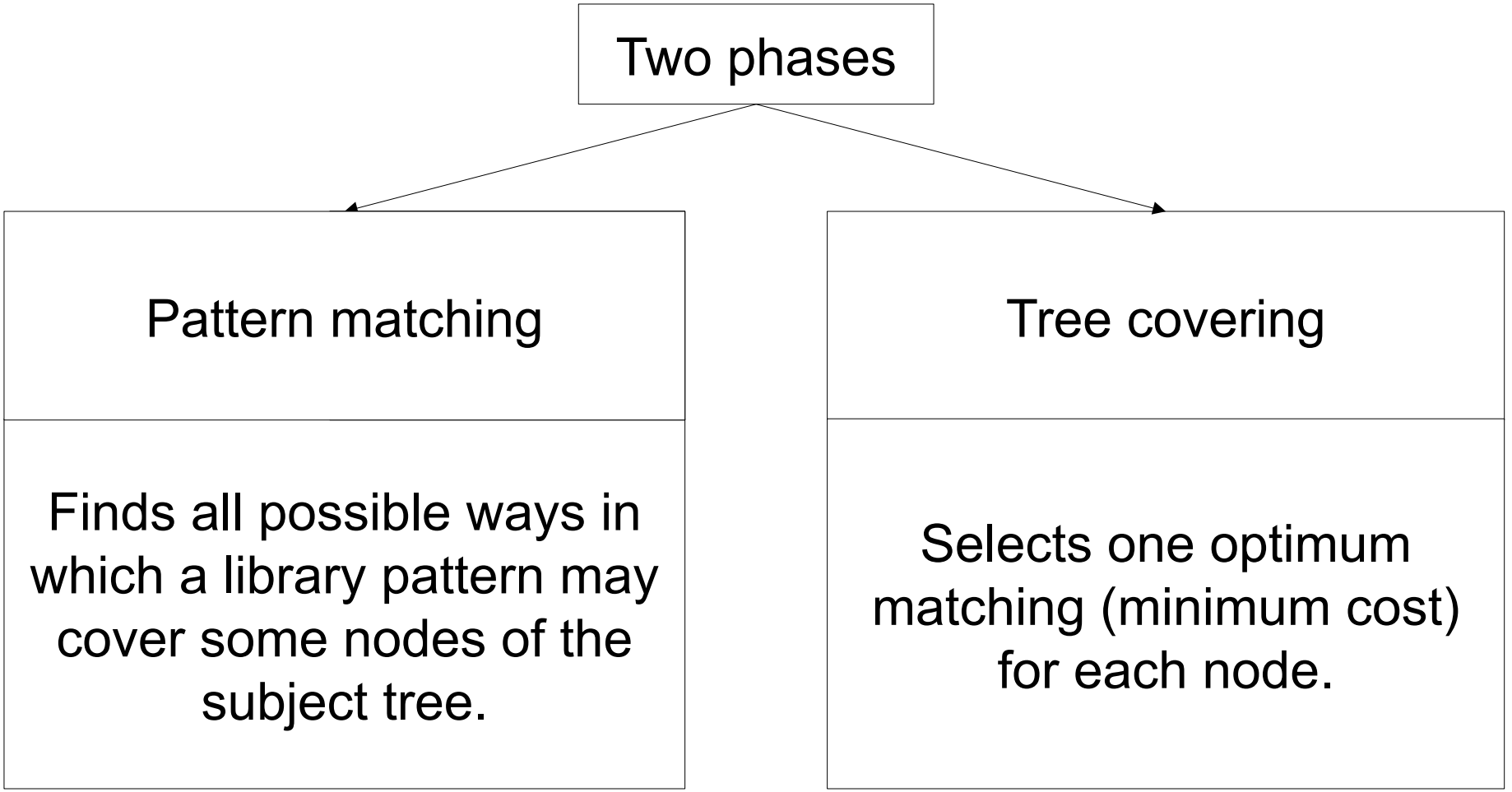
Splitting Subject Graph into Trees

- Let's consider the previous subject graph and split it:



The circuit on the left is not a tree, because there is one gate that feeds two other gates (marked as red crosses). So splitting it at these two locations, we obtained three trees (1, 2 and 3) on the right.

The Tree Mapping Procedure



Some Assumptions

- Let's consider the Tree-3 for pattern matching and the library containing three gates (INV,1; NAND2,2; NAND3,3) as shown below:

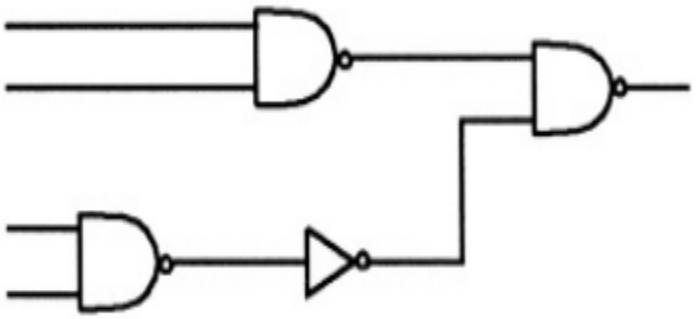


Figure: Tree-3 of the subject graph

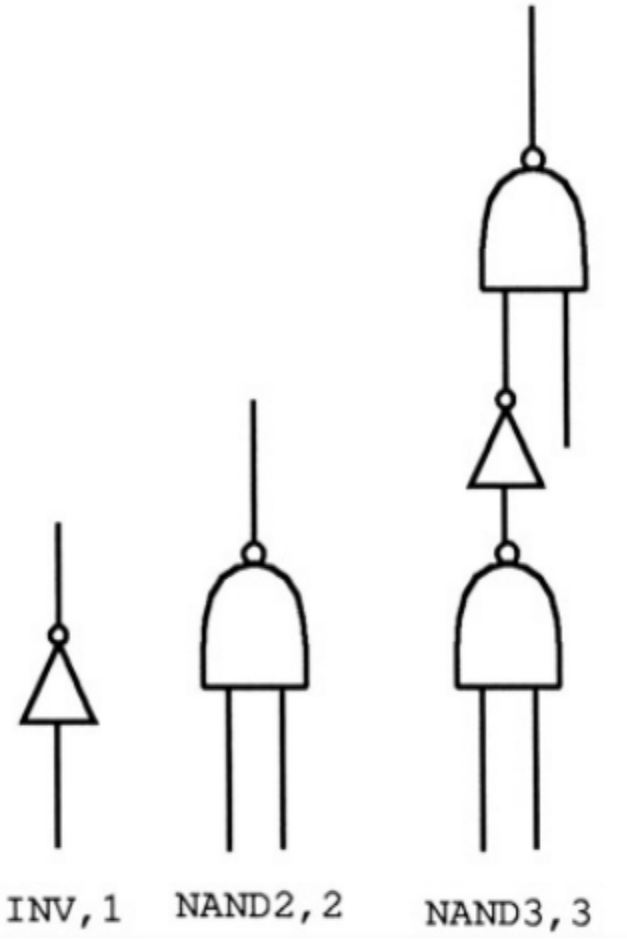


Figure: Library gates

Pattern Matching (Cover 1)

- If we select the three-input NAND gate to match the output node, then we cover gates f and g, besides h; hence, we do not need matches for f and g.

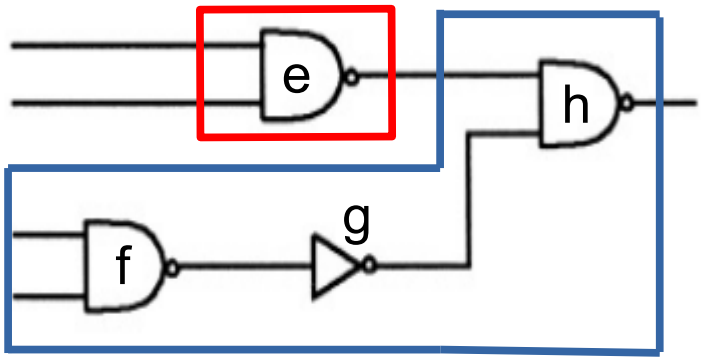


Figure: Tree-3 of the subject graph

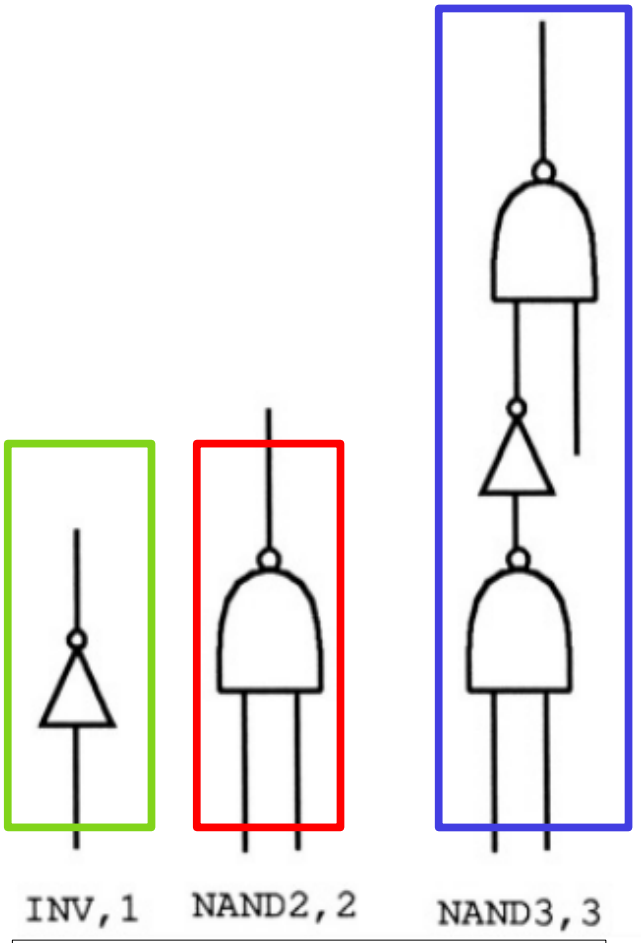


Figure: Library gates

Pattern Matching (Cover 2)

- If we choose the two-input NAND gate, then we need to select gates to cover f and g.

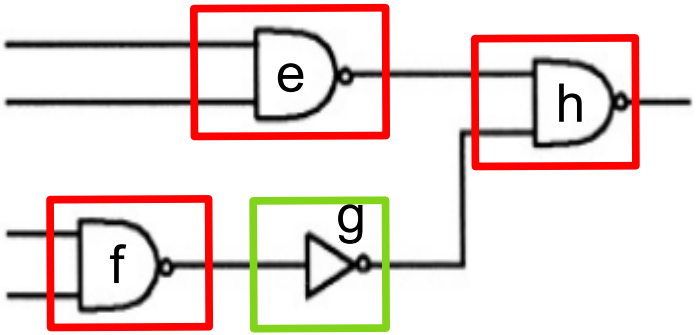


Figure: Tree-3 of the subject graph

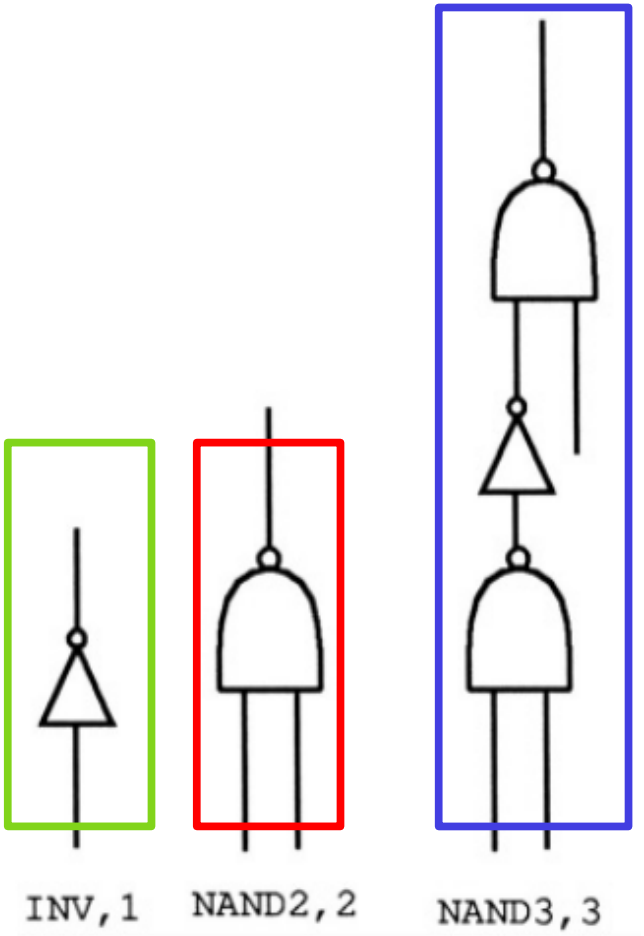
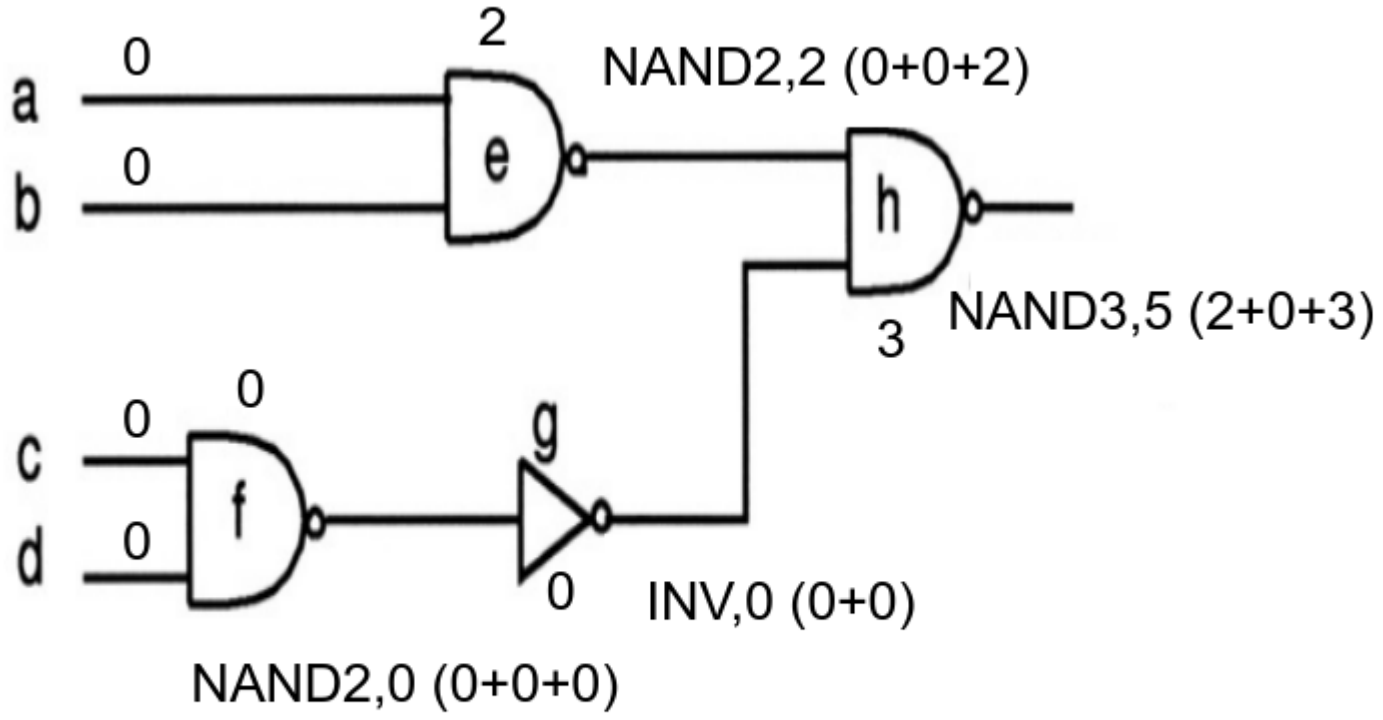


Figure: Library gates

Tree Covering (Cover 1)



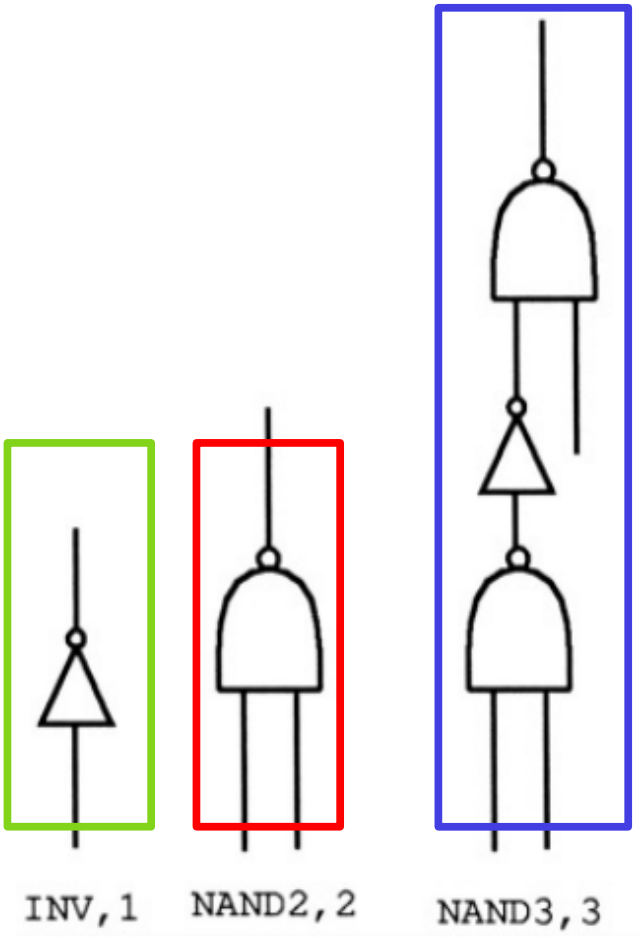
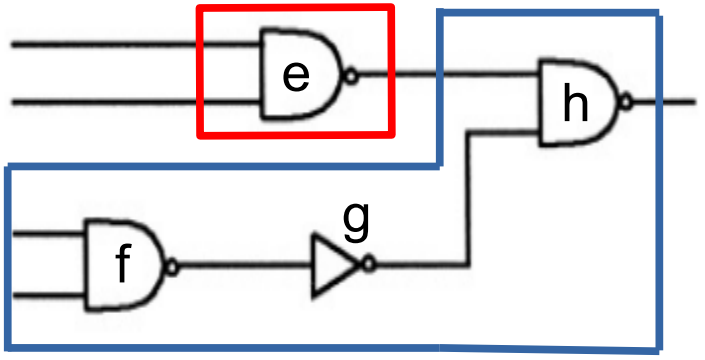


Figure: Optimum cover selection



Tree Covering (Cover 2)

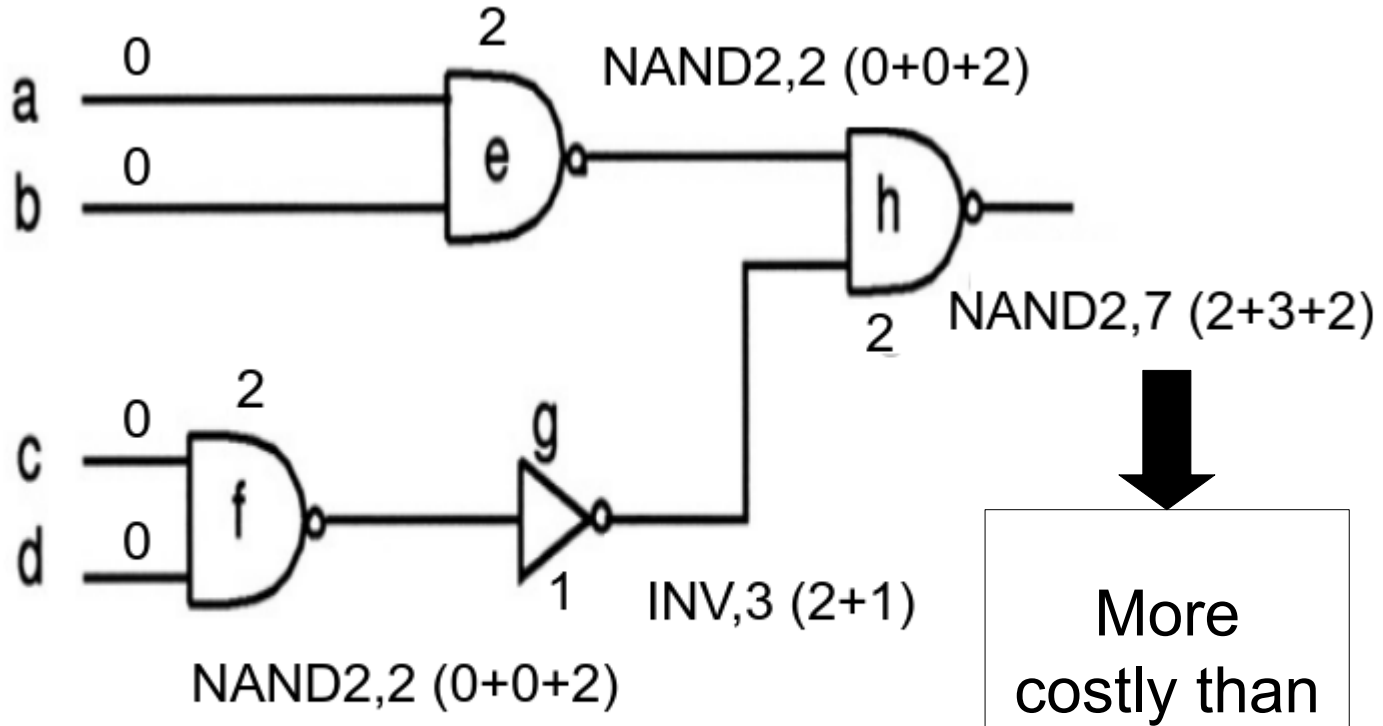
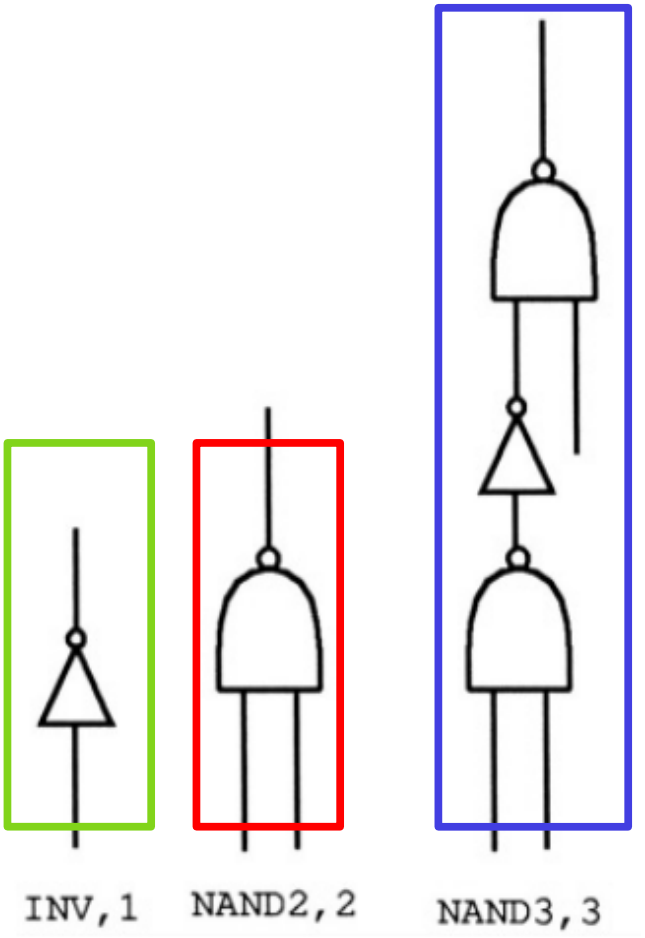
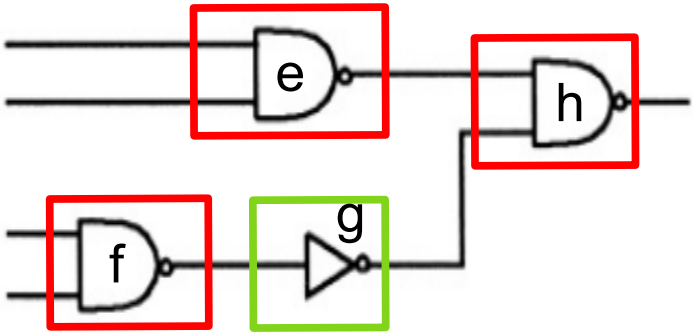


Figure: Optimum cover selection

More costly than Cover 1, so it is discarded





Advantages of Tree-Covering Problem

- The NP-hardness of DAG-covering is removed.
- Linear complexity can be achieved due to splitting of the subject graph.
- Pattern matching is easier due to the reduction of search space.
- This method has proven to be quite effective.

Limitations of Tree-Covering Problem

- Loss of global view due to the step of partitioning into trees.
- Covers across partition boundaries are not allowed.
- Inefficient reduction of search space may affect the quality of the final solution.
- Does not always provide the exact optimum solution.

References

- Logic Synthesis and Verification Algorithms, Gary D. Hachtel and Fabio Somenzi.
- DAGON: Technology Binding and Local Optimization by DAG Matching, K. Keutzer.

Thank You!