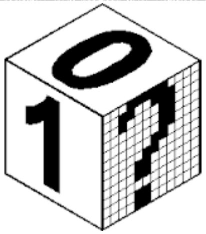
THE PROBLEM OF MINIMIZING BOOLEAN CIRCUITS I.E. THE MINIMUM SET COVER PROBLEM

Prof. Tiziana Calamoneri
Network Algorithms
A.y. 2020/21



THE PROBLEM

MINIMIZING BOOLEAN FUNCTIONS (1)

- All the datapath and control structures of a digital device can be represented as boolean functions, which can take a disjunctive normal form (DNF) on the variables and their complements:

$$y=C_1 \lor C_2 \lor \ldots \lor C_m$$

where $C_i=l_{i1} \land \ldots \land l_{ik_i}$ and l_{ij} is choosen among n boolean variables.

- These boolean functions must be converted into logic networks in the most economical way.
- …

MINIMIZING BOOLEAN FUNCTIONS (2)

- …
- What qualifies as the "most economical way" varies, depending on whether the network is built using discrete gates, a programmable logic device with a fixed complement of gates available, or a fully-customized integrated circuit. But in all cases, minimization yields a network with as a small number of gates as possible, and with each gate as simple as possible.

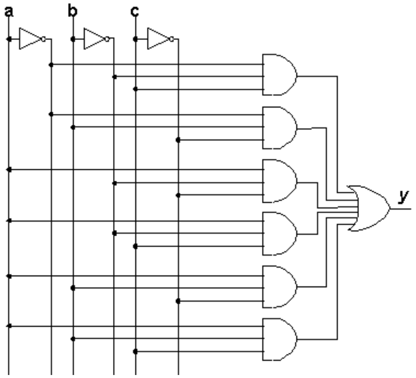
MINIMIZING BOOLEAN FUNCTIONS (3)

To appreciate the importance of minimization, consider as an example the following function:

$y=(a'\wedge b'\wedge c) \vee (a'\wedge b\wedge c') \vee (a\wedge b'\wedge c') \vee (a\wedge b'\wedge c) \vee (a\wedge b\wedge c') \vee (a\wedge b\wedge c)$

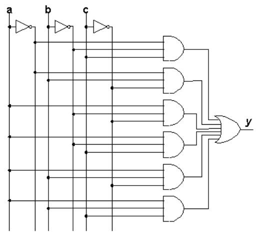
Note: l_{ij}' means not l_{ij}

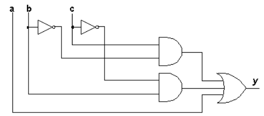
which can be easily translated in circuit as follows:



MINIMIZING BOOLEAN FUNCTIONS (4)

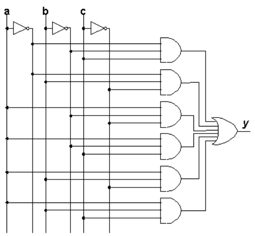
- But there is another circuit that produces at *y* exactly the same result if you put the same pattern of values into the corresponding inputs.
- Yet, this second network uses far fewer gates, and the gates it uses are simpler (have smaller fan-ins) than the gates of the first network.

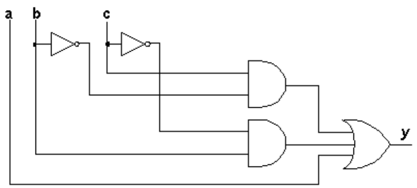




MINIMIZING BOOLEAN FUNCTIONS (5)

- Clearly, the minimized circuit is less expensive to build than the unminimized version.
- Although it is not true in this case, it is often the case that minimized networks will be faster (have fewer propagation delays) than unminimized networks.





MINIMIZING BOOLEAN FUNCTIONS (6)

Problem: We are given a particular Boolean function of *n* variables, which for each of the 2^n possible input vectors describes whether the desired output is *0* or *1*.

We seek the simplest circuit that exactly implements this function.

Example: $y=(a\wedge b') \vee (a\wedge c) \vee (b'\wedge c') \vee (a'\wedge c) \vee (a\wedge b) \vee (b\wedge c)$

Note:
- Rows indicate inputs;
- Columns indicate clauses;
- 1 means that the clause is true for that input;
- 0s are omitted.

abc	1st	2nd	3rd	4th	5th	6th
000			1			
001				1		
010						
011				1		1
100	1		1			
101	1	1				
110					1	
111		1			1	1

MINIMIZING BOOLEAN FUNCTIONS (7)

We could build one *and* term for each input vector and then *or* them all together, but we might save considerably by factoring out common subsets of variables.

Given a set of feasible *and* terms, each of which covers a subset of the vectors we need, we seek to *or* together the smallest number of terms that realize the function.

$y=(a \wedge b') \vee (a \wedge c) \vee (a' \wedge c') \vee (a' \wedge c) \vee (a \wedge b) \vee (b \wedge c)$

abc	1st	2nd	3rd	4th	5th	6th
000			1			
001				1		
010						
011				1		1
100	1		1			
101	1	1				
110					1	
111		1			1	1

Note: orange columns can be ignored without affecting the realization of the function

MINIMIZING BOOLEAN FUNCTIONS (8)

This is exactly the set cover problem:

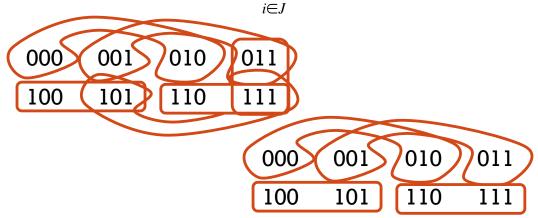
Set Cover Problem:

Given a set of subsets $S=\{S_1, \dots, S_n\}$ of the universal set U such that

$$\bigcup_{i=1..m} S_i = U$$

what is the smallest subset J of $\{1, ..n\}$ such that

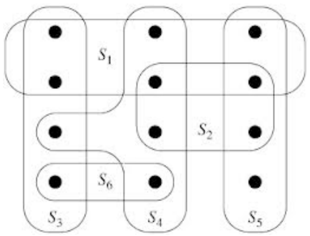
$$\bigcup_{i \in J} S_i = U \text{ ?}$$



Incidence matrix of the set of subsets

abc	1st	2nd	3rd	4th	5th	6th
000			1			
001				1		
010			1			
011				1		1
100	1					
101	1	1				
110					1	
111		1			1	1

THE SET COVER PROBLEM



ILP FORMULATION OF SET COVER

- Let x_i be a boolean variable associated with each subset S_i.
- x_i is 1 if S_i is in the solution, and 0 otherwise.
- The following ILP encodes the Set Cover Problem:

$$\min \sum_{i=1}^{n} x_i$$

s.t. $$\sum_{i: e \in S_i} x_i \geq 1 \quad \forall e \in U$$

$$x_i \in \{0,1\}$$

In other words, every element of U is present in at least one of the chosen subsets, and their number must be minimized.

COMPLEXITY OF SET COVER

- Vertex cover can be seen as a special case of set cover, namely:
  - Instance of VC: $G=(V,E)$
  - Instance of SC: the universe is the set of edges E and the subsets are: for each node v, $S_v = \{e_j : v$ is adjacent to edge $e_j\}$.
  - Solving SC implies solving VC hence:

Th. The Set Cover Problem is NP-hard.

- Note that this collection of sets has the property that each universe element appears in exactly two sets.
- This leads to what is called the *f*-frequency set cover problem where each element occurs in at most *f* sets.
- Vertex cover is essentially the 2-frequency set cover problem.

OTHER APPLICATIONS

The set cover problem has many applications. Here two are:

- There are n files $S_1, \ldots, S_n$, and there are m requests for information. Each unit of information is stored in at least one file. Find a subset of the files of minimum cardinality such that searching these will retrieve all the requested information.
- An airline has m flights $x_1, \ldots, x_m$. These flights can be combined into "flight legs" $S_1, \ldots, S_n$ such that the same crew can service all the flights in leg S_j. Find the minimum number of crews required to service all flights. Note that the number of flight legs may be much larger than the number of crews.

APPROXIMATION ALGORITHMS (1)

A simple approximation algorithm is the greedy algorithm, whose performance is $O(\log |U|)$.

Algorithm Greedy

Input: family $S=\{S_1, \ldots, S_n\}$ of the universal set U

Output: J subset of $\{1, \ldots, n\}$ s.t. $\bigcup_{i \in J} S_i = U$

$X = U$ /*currently uncovered elements
J=empty set
While X is not empty do
  choose a subset S_j in S such that $|S_j \cap X|$ is max
  $X = X \setminus S_j$
  $J = J \cup \{j\}$

APPROXIMATION ALGORITHMS (2)

Th. The performance ratio of Algorithm Greedy is $O(\ln |U|)$.

Sketch of proof. The proof is based on the key point that the greedy chosen set S_j is such that:

$$|S_j \cap X| \ge \frac{|X|}{|J_{opt}|}$$

This obs. is a consequence of the greedy choice, and the def. of optimal solution: J_{opt} covers all elements of U, and hence also the elements of X. By averaging among the sets in J_{opt}, the one which covers the max number of points of X must cover at least $|X|/|J_{opt}|$.

Since the greedy alg. chooses among all the sets the one with the max new coverage, this coverage must be at least as much as claimed.

APPROXIMATION ALGORITHMS (3)

(proof of the performance ratio of Alg Greedy – cntd)

Let the indices of the sets picked by the greedy alg. in the order they were picked be $j_1, \ldots, j_g$.

For $t=1, \ldots, g$ let X_t be the set X just before the set J_{jt} was picked.

So, for example, $X_1=U$.

Define X_{g+1}=empty set.

The following simple recurrency holds:

$|X_{t+1}|=|X_t|-|S_{jt} \cap X_t|$.

APPROXIMATION ALGORITHMS (4)

(proof of the performance ratio of Alg Greedy – cntd)

Join together

$|X_{t+1}|=|X_t|-|S_{jt} \cap X_t|$ and $|S_j \cap X_t| \geq \frac{|X_t|}{|J_{opt}|}$ to get:

$|X_{t+1}| \leq |X_t|-|X_t|/|J_{opt}|=|X_t|(1-1/|J_{opt}|)$.

Enrolling the recurrence:

$|X_{t+1}| \leq |X_t|(1-1/|J_{opt}|) \leq |X_{t-1}|(1-1/|J_{opt}|)^2 \leq \ldots$

$\leq |X_{t-(t-1)}|(1-1/|J_{opt}|)^t=|X_1|(1-1/|J_{opt}|)^t=$

$=|U|(1-1/|J_{opt}|)^t$

APPROXIMATION ALGORITHMS (5)

(proof of the performance ratio of Alg Greedy – cntd)

$|X_{t+1}| \leq |U|(1-1/|J_{opt}|)^t$

$$\frac{|X_{t+1}|}{|U|} \leq \left(1-\frac{1}{|J_{opt}|}\right)^t \Rightarrow \frac{|U|}{|X_{t+1}|} \geq \left(\frac{|J_{opt}|}{|J_{opt}|-1}\right)^t$$

$$\Rightarrow \ln\frac{|U|}{|X_{t+1}|} \geq t\ln\left(1+\frac{1}{|J_{opt}|-1}\right) \approx \frac{t}{|J_{opt}|}$$

$$\Rightarrow t \leq |J_{opt}|\ln\frac{|U|}{|X_{t+1}|} \quad \forall t=1,\ldots,g$$

Since X_g is not empty: $|J_{greedy}|-1=g-1 \leq |J_{opt}| \ln |U|$.

APPROXIMATION ALGORITHMS (6)

This result is the best we can do, indeed:

If we call $n=|U|$, Set Cover cannot be approximated within:

- a factor of *½ log n* [Lund & Yannakakis '94]
- a factor of *(1-o(1))ln n* [Feige '98]

(unless NP has quasi-polynomial time algs)

- a factor of *c log n* [Raz & Safra '97]
- a similar result with a higher value of *c* [Alon, Moshkovitz & Safra '06]

(unless P does not coincides with NP – weaker hypothesis).

APPROXIMATION ALGORITHMS (7)

Another approximation algorithm is based on a relaxation of the ILP formulation.

Let F the max frequency of an element, i.e. the max number of subsets an element appears in.

Algorithm LP
Input: family $S=\{S_1, \ldots, S_n\}$ of the universal set U
Output: J subset of $\{1, \ldots, n\}$ s.t.

Solve the LP formultion and let $x'_1, \ldots, x'_n$ be a solution
for i=1 to n do
  if $x'_i \geq 1/F$
    then $x_i=1$
    else $x_i=0$

APPROXIMATION ALGORITHMS (8)

Th. Alg LP works correctly and its approximation ratio is F.

Proof. Let us remind the LP formulation:

$$\min \sum_{i=1}^{n} x_i \quad \text{s.t.} \quad \sum_{i:e\in S_i} x_i \geq 1 \forall e \in U$$

In every constraint, there are at most F variables to be summed, so at least one of them must have value ≥1/F. So, the whole universe U is covered.

Since $x_i=1$ if $x'_i \geq 1/F$ and 0 otherwise, it holds that $x'_i \geq x_i/F$ from which: $|J_{opt}| \geq \sum_{i=1}^{n} x'_i \geq \frac{1}{F}\sum_{i=1}^{n} x_i \geq 1/F|J|$. ■

APPROXIMATION ALGORITHMS (9)

The same approximation ratio can be achieved by extending one of the algorithms designed for the vertex cover problem:

Algorithm SetCover
Input: family $S=\{S_1, \ldots, S_n\}$ of the universal set U
Output: J subset of $\{1, \ldots, n\}$ s.t. $\bigcup_{i\in J} S_i = U$

$X = U$ /*currently uncovered elements
J=empty set
While X is not empty do
  pick an element e of X not covered by J
  add to J the indices of all sets S_i containing e
  eliminate from X all element covered by the found sets

APPROXIMATION ALGORITHMS (10)

Th. Alg. SetCover works correctly and its approximation ratio is F.

Proof. It is a generalization of the proof for the Vertex Cover.

RELATED PROBLEMS (1)

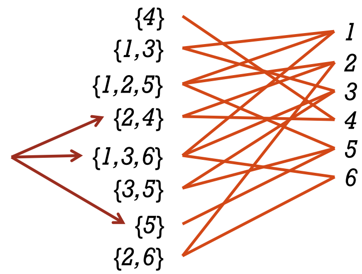
- Besides Vertex Cover, that is a special case of Set Cover, many other problems are related with Set Cover.
- An instance of Set Cover can be viewed as an arbitrary bipartite graph, with sets represented by nodes on the left, the universe represented by nodes on the right, and edges representing the inclusion of elements in sets.
- The task is to find a minimum cardinality subset of left-nodes which covers all of the right-nodes.

RELATED PROBLEMS (2)

Example:

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

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

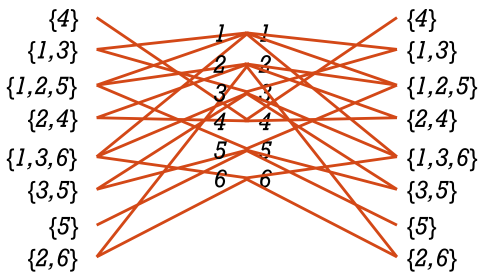


RELATED PROBLEMS (3)

Hitting set problem:

Given a bipartite graph, the objective is to cover the left-nodes using a minimum subset of the right nodes.

Converting from Set Cover to the Hitting Set is therefore achieved by interchanging the two sets of vertices.



RELATED PROBLEMS (4)

Edge Cover problem:

Given a graph, an edge cover is a set of edges such that every node is incident to at least one edge of the set.

The minimum edge cover problem is the problem of finding an edge cover of minimum size.

- Edge Cover is a special case of Set Cover, where:
  - $U=V$ and $S=E$

RELATED PROBLEMS (5)

Set Packing problem:

Given a universe *U* and a family *S* of subsets of *U*, a packing is a subfamily *J* of sets such that all sets in *J* are pairwise disjoint.

In the set packing problem, the input is a pair *(U,S)*, and the task is to find a set packing that uses the most sets.

- Set Packing is the dual problem of Set Cover.

RELATED PROBLEMS (6)

Exact Cover problem:

Exact Cover problem is to choose a Set Cover with no element included in more than one covering set.