Approximation Algorithm For Set Cover

Narasimhan Kovalai

Roll Number-20CS01075
School of Electrical Sciences (SES)
Indian Institute of Technology Bhubaneswar

20CS01075@iitbbs.ac.in

March 16, 2022

OUTLINE OF THE PRESENTATION

Problem Statement

Set Cover Problem

Given an instance (X, F), where X is finite in nature and F is a family of subsets in X, such that every element of X belongs to at least one subset in F, the problem is to find a minimum size subset Ψ, whose members cover all of X. Mathematically:

$$X = \bigcup_{S \in F} S$$

$$X = \bigcup_{S \in \Psi} S$$

Decision Set Cover Problem

- In the **Decision** Set cover Problem, given an additional positive integer **k** as an input, we decide whether a set cover of size utmost **k** exists or not.
- This version of the problem is **NP-Complete** in nature.It is one of Karp's 21 NP-Complete problems and can be reduced so from the vertex cover problem.

Optimization Set cover Problem

- Given an instance (X, F), where X is finite in nature and F is a family of subsets in X , such that every element of X belongs to at least one subset in F , the problem is to find a minimum size subset Ψ, whose members cover all of X.
- It is NP-hard in nature. However, we are always free to transform an optimization problem to a decision(Boolean Problem) by introducing an upper bound integer k.

NP,NP Complete and NP Hard

- A decision problem β is **NP-Complete** if it satisfies the following conditions:
  1. β is in NP.
  2. Every problem in NP is reducible to β in polynomial time.

  β can be shown to be in NP by demonstrating that a candidate/certificate solution to the problem can be verified in polynomial time.

- A problem statement which satisfies condition **1** but not necessarily condition **2** is said to be **NP-hard**.

Frame Title

From the diagram, we can observe the following:

- Not all NP-hard problems are NP-complete. The intersection between them indicates the NP-Hard problem is itself an NP problem.
- NP-Complete problems are key to decode the ever-pervading question of whether P = NP or P $\neq$ NP.
- If it is proved that a certain NP-Complete problem is solvable in polynomial time, this automatically implies that every NP problem is solvable in polynomial time, resulting in P=NP.

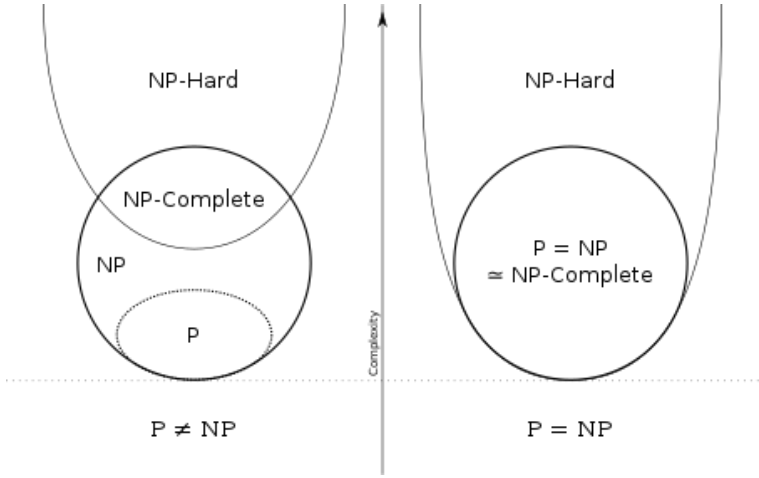


Figure: Euler diagram for P,NP,NP-Complete and NP-hard set of problems. The left diagram assumes that P$\neq$ NP while the right is valid for P=NP.

Designing the Algorithm(Greedy Approach)

The Greedy approach would give us a near-optimal solution. At each stage, we select the set S which covers the maximum number of the remaining elements and append that set S to Ψ.

Algorithm 1 GREEDY SET COVER APPROXIMATION ALGORITHM

1: $U{=}X$
2: $\Psi{=}\Phi$
3: **while** $U \neq \Phi$ **do**
4: select an $S \in F$ which maximizes $|S \cap U|$
5: $U = U - S$
6: $\Psi = \Psi \cap S$
7: **end while**
8: return Ψ

Time Complexity of the Algorithm

- The while loop at line 3 runs until U is a null set. At each iteration, S is extracted from it. Thus there are $\min(|X, F|)$ iterations in the algorithm.
- The loop body (lines 4-6) takes $\mathbf{O(|X||F|)}$ time to execute.
- Therefore, the overall time complexity of the Greedy Set Cover Algorithm is $\mathbf{O(|X||F|}min(|X||F|)\mathbf{)}$.
- However, we can also devise a separate algorithm for it to run in linear time.

A 12 element Universe

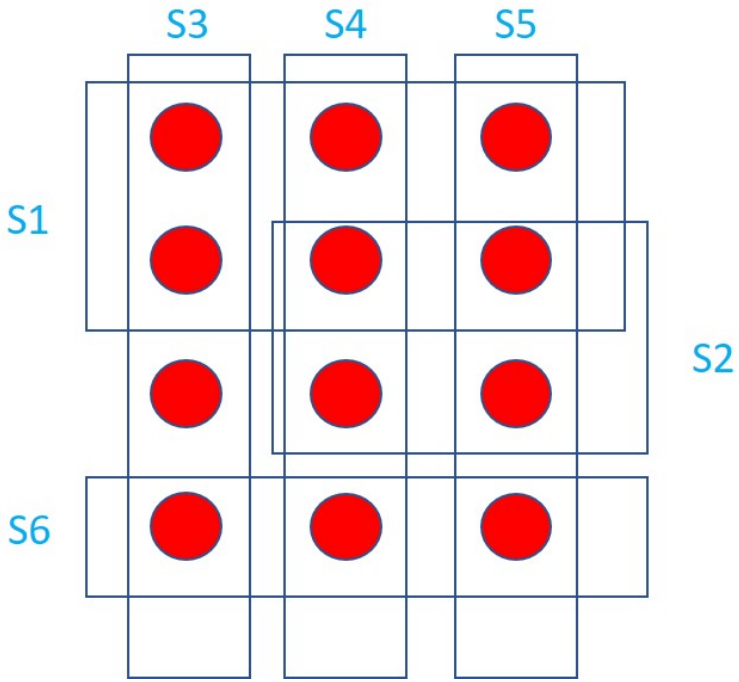


Figure: Initial Set X with $|X| = 12$ and $F = \{S1, S2, S3, S4, S5\}$

A 12 element Universe

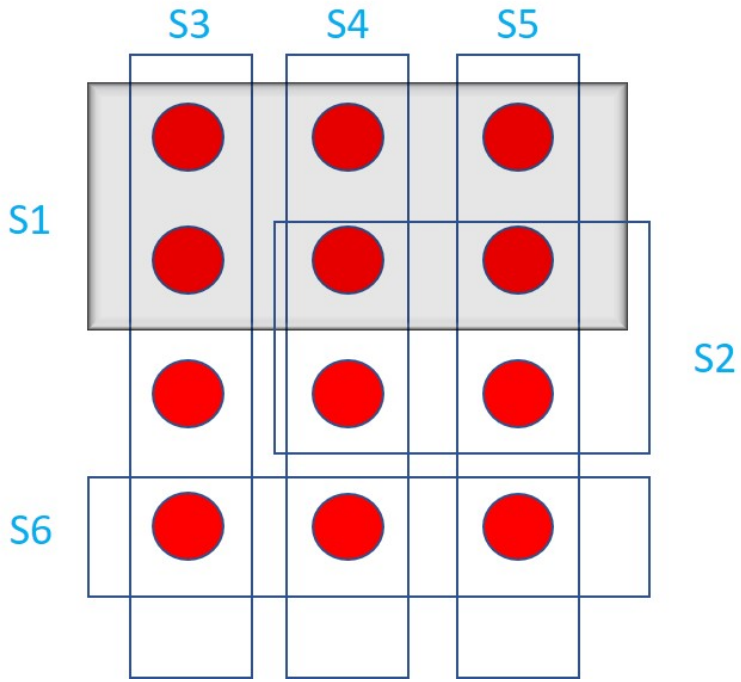


Figure: $|S1|$ is chosen greedily as $|S1 \cap U|$ is maximum=6

A 12 element Universe

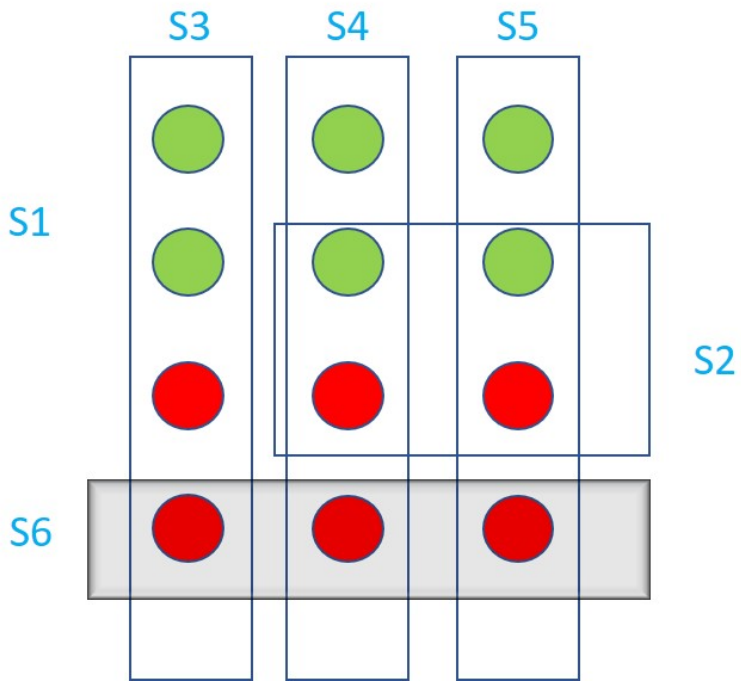


Figure: $|S6|$ is chosen greedily as $|S6 \cap U|$ is maximum=3

A 12 element Universe

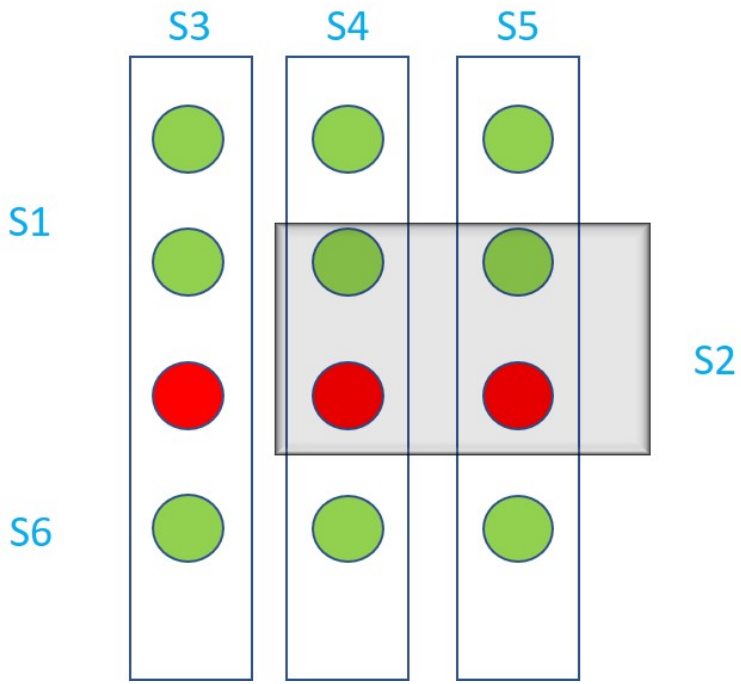


Figure: $|S2|$ is chosen greedily as $|S2 \cap U|$ is maximum=2

A 12 element Universe

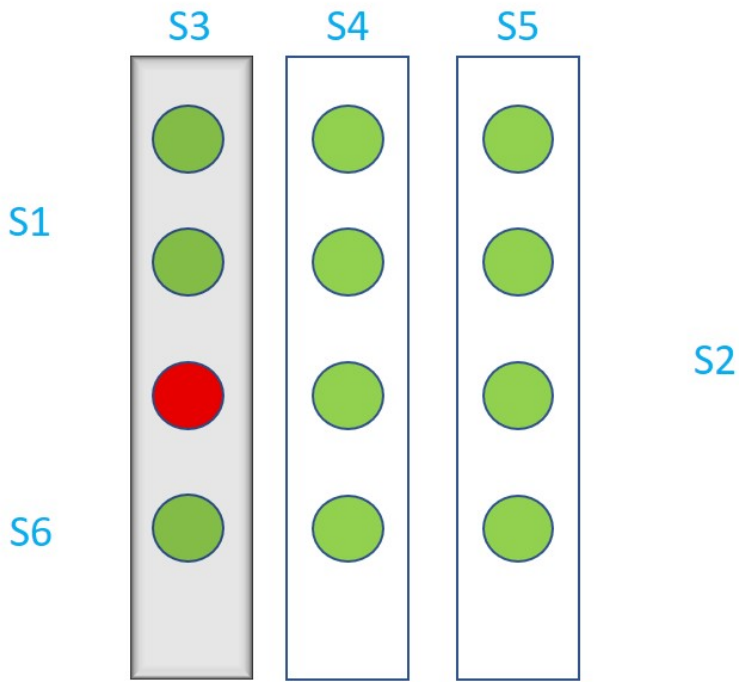


Figure: $|S3|$ is chosen greedily as $|S3 \cap U|$ is maximum=1

A 12 element Universe

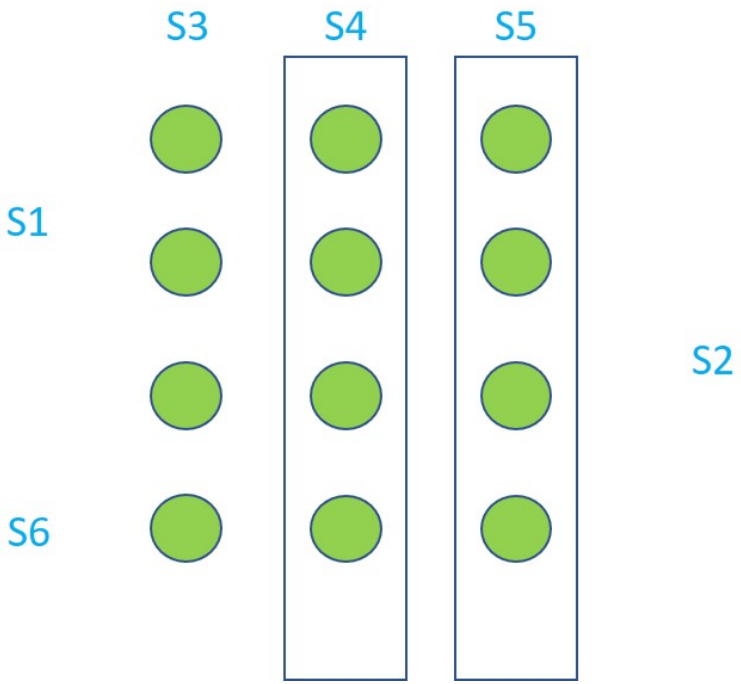


Figure: $U = \Phi$, algorithm is terminated with $\Psi = \{S1, S2, S3, S6\}$

In case of a tie (when 2 sets have equal cardinality), then the greedy algorithm may **foolishly select** the sets which will lead to significant deviation from the optimum set cover.

- In the diagram on the right, the optimal cover is given by $\Psi = \{S5, S6\}$ (shown in black). This is also the expected result using the greedy approach.
- But since in each step there is a tie, there may occur a case where the sets enclosed in green borders are selected. This results in $\Psi = \{S1, S2, S3, S4\}$ and thus $|\Psi| = 4$.

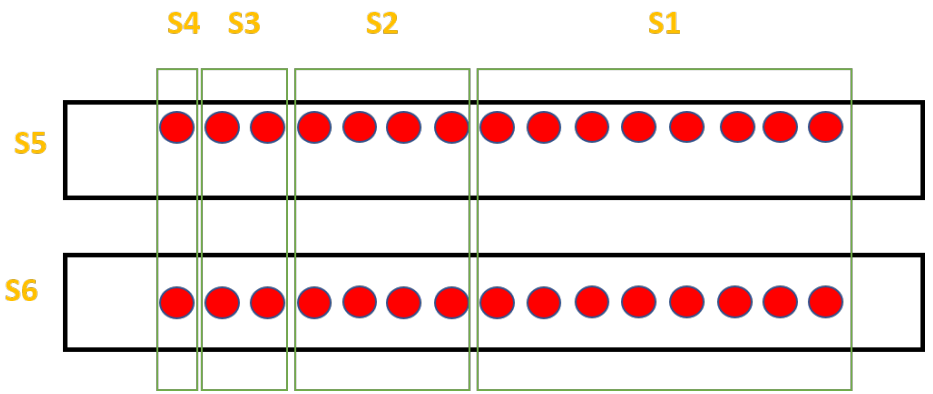
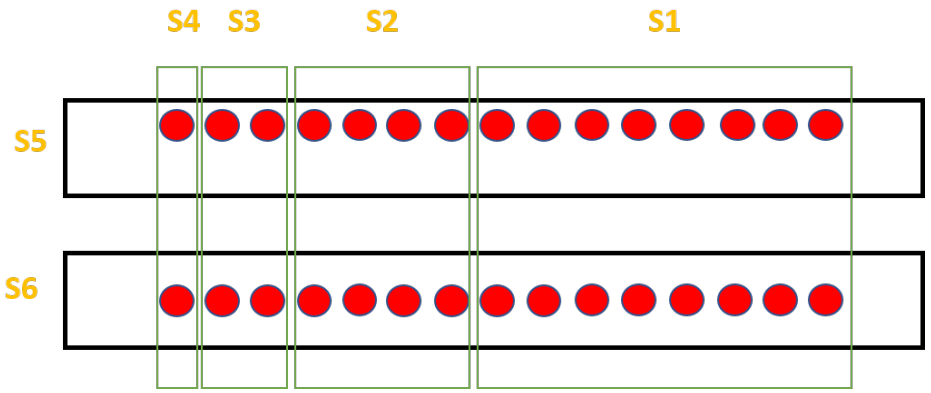


Figure: Diagram showing the case when ties are broken in the worst possible way.

- If C represents the optimum count of Ψ while C^* represents the greedy count of Ψ, then in this case $C^*/C = 2$, which is upper bounded by $\ln |X|$. An even tighter bound is $\log_2 |X|/2$.
- In general, if $|X| = 2^{(k+1)} - 2$, then k sets($\lfloor \log_2 |X| \rfloor$) are chosen by the greedy algorithm, which implies an approximation ratio of $\log_2 |X|/2$
- This can be solved efficiently by assigning a cost $c_i, \forall S_i \in F$. This leads to a weighted set cover problem which is discussed next.



We say that an algorithm has an $\rho(n)$ **approximation ratio**,if for any input size n, the cost C of the solution produced by the algorithm is within a factor of $\rho(n)$ of the cost C^* of an optimal solution. This is always greater than 1 for both maximization and minimization problems. Here it implies how slow the greedy cover is with respect to the optimal cover.

Theorem

The GREEDY SET COVER APPROXIMATION ALGORITHM is a ρ (n) approximation algorithm, where $\rho(n) = H(max\{|S| : S \in F\})$, where $G(d)$ is the harmonic sum from 1 to d, for $d \in Z^+$.

Theorem

Given any set system $\Sigma = (X, S)$, let G be the output of the greedy cover and let O be an optimum cover. Then $|G| \leq |O| \cdot \ln m$, where $m = |X|$.

Proof.

We now provide a crude proof or intuition behind the second theorem(derived from the first theorem).
We know that $H(n) = 1 + 1/2 + 1/3...1/n$. If we approximate it to a continuous nature integral: $\int_1^n f(r)dr$, where $f(r) = 1/r$, we see that it is equivalent to $\ln(n) - \ln(1) = \ln(n)$, thus obtaining the desired result.
A more rigorous proof is included in the report. □

Weighted Set Cover Problem

Problem Statement :Given a universe U of n elements, a collection of subsets of U say $F = \{S_1, S_2 \ldots, S_m\}$, where each $S_i \in F$ has a cost w_i, the task is to find out the Minimum Set cover.

Approach: We can model it as a Integer Linear Program (ILP):

Objective function to minimize :

$$\sum_{i=1}^{m} x_i \cdot w_i$$

Constraints:

$$\sum_{i|j \in S_i} x_i \geq 1, \forall j \in \{1, 2..|U|\}$$

$$x_i = \{0, 1\}, \forall i \in \{1, 2...m\}$$

The Famous Coffee Shop Problem

The problem detail and implementation are as follows :

LOCATION INDEX	SUBSET
1	1,2,3
2	1,2,3,4
3	1,2,3,4
4	2,3,4
5	5,6,7
6	5,6,8
7	5,7,8,9
8	6,7,8,9
9	7.8.9
10	8.10,11
11	10,11,12,13
12	10,11,12,13
13	11,12,13

Figure: Table showing the formulation of the problem as a weighted set cover problem.

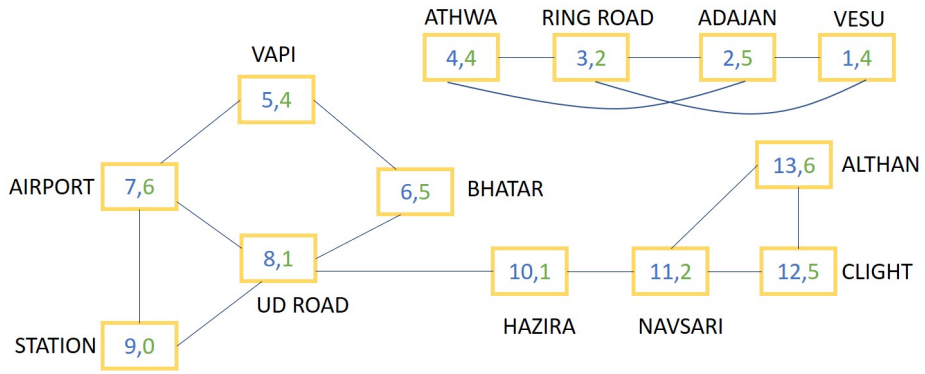


Figure: Diagram showing a city map with a cost associated to each area to open a shop. The first number is index while second number indicates cost of setting up.

The Famous Coffee Shop Problem

Objective Function(to be minimised):

$$H = 4x_1 + 5x_2 + 2x_3 + 4x_4 + 4x_5 + 5x_6 + 6x_7 + x_8 + 0x_9 + x_{10} + 2x_{11} + 5x_{12} + 6x_{13}$$

Constraints:

- $x_1 + x_2 + x_3 \geq 1$
- $x_1 + x_2 + x_3 + x_4 \geq 1$
- $x_1 + x_2 + x_3 + x_4 \geq 1$
- $x_2 + x_3 + x_4 \geq 1$
- $x_5 + x_6 + x_7 \geq 1$
- $x_5 + x_6 + x_8 \geq 1$
- $x_5 + x_7 + x_8 + x_9 \geq 1$

The Famous Coffee Shop Problem

- $x_6 + x_7 + x_8 + x_9 \geq 1$
- $x_7 + x_8 + x_9 \geq 1$
- $x_8 + x_{10} + x_{11} \geq 1$
- $x_{10} + x_{11} + x_{12} + x_{13} \geq 1$
- $x_{10} + x_{11} + x_{12} + x_{13} \geq 1$
- $x_{11} + x_{12} + x_{13} \geq 1$
- $x_i \in \{0, 1\}, \forall i \in \{1, 13\}$

This is a NP-hard problem too. We can solve this in standard Excel solver using the famous **Simplex Algorithm**. The following slides show the Greedy approach

Greedy Weighted Set Cover

Algorithm 2 GREEDY WEIGHTED SET COVER APPROXIMATION

1: I: Set of covered elements.
2: $I = \Phi$
3: **for** $S_i \in U$ **do**
4: Ratio=C(S_i)/$|S_i - I|$
5: **end for**
6: **while** $I \neq U$ **do**
7: Find S_i corresponding to minimum(Ratio).Prefer minimum cost if ratio is tied.
8: $I = I \cup S_i$
9: Update Ratio
10: **end while**
11: return all selected S_i

ITERATION-1 : Select S_9.

LOCATION INDEX	SUBSET	I	S_i-I	COST	Ratio
1	1,2,3	NULL	1,2,3	4	1.33
2	1,2,3,4	NULL	1,2,3,4	5	1.25
3	1,2,3,4	NULL	1,2,3,4	2	0.5
4	2,3,4	NULL	2,3,4	4	1.33
5	5,6,7	NULL	5,6,7	4	1.33
6	5,6,8	NULL	5,6,8	5	1.67
7	5,7,8,9	NULL	5,7,8,9	6	1.5
8	6,7,8,9	NULL	6,7,8,9	1	0.25
9	7,8,9	NULL	7,8,9	0	0
10	8,10,11	NULL	8,10,11	1	0.33
11	10,11,12,13	NULL	10,11,12,13	2	0.5
12	10,11,12,13	NULL	10,11,12,13	5	1.25
13	11,12,13	NULL	11,12,13	6	2

ITERATION-2 : Select S_{10}.

LOCATION INDEX	SUBSET	I	S_i-I	COST	Ratio
1	1,2,3	7,8,9	1,2,3	4	1.33
2	1,2,3,4	7,8,9	1,2,3,4	5	1.25
3	1,2,3,4	7,8,9	1,2,3,4	2	0.5
4	2,3,4	7,8,9	2,3,4	4	1.33
5	5,6,7	7,8,9	5,6	4	2
6	5,6,8	7,8,9	5,6	5	2.5
7	5,7,8,9	7,8,9	5	6	6
8	6,7,8,9	7,8,9	6	1	1
9	7,8,9	7,8,9	NULL	0	-
10	8,10,11	7,8,9	10,11	1	0.5
11	10,11,12,13	7,8,9	10,11,12,13	2	0.5
12	10,11,12,13	7,8,9	10,11,12,13	5	1.25
13	11,12,13	7,8,9	11,12,13	6	2

ITERATION-3 : Select S_3.

LOCATION INDEX	SUBSET	I	S_i-I	COST	Ratio
1	1,2,3	7,8,9,10,11	1,2,3	4	1.33
2	1,2,3,4	7,8,9,10,11	1,2,3,4	5	1.25
3	1,2,3,4	7,8,9,10,11	1,2,3,4	2	0.5
4	2,3,4	7,8,9,10,11	2,3,4	4	1.33
5	5,6,7	7,8,9,10,11	5,6	4	2
6	5,6,8	7,8,9,10,11	5,6	5	2.5
7	5,7,8,9	7,8,9,10,11	5	6	6
8	6,7,8,9	7,8,9,10,11	6	1	1
9	7,8,9	7,8,9,10,11	NULL	0	-
10	8,10,11	7,8,9,10,11	NULL	1	-
11	10,11,12,13	7,8,9,10,11	12,13	2	1
12	10,11,12,13	7,8,9,10,11	12,13	5	2.5
13	11,12,13	7,8,9,10,11	12,13	6	3

ITERATION-4 : Select S_8.

LOCATION INDEX	SUBSET	I	S_i-I	COST	Ratio
1	1,2,3	1,2,3,4,7,8,9,10,11	NULL	4	-
2	1,2,3,4	1,2,3,4,7,8,9,10,11	NULL	5	-
3	1,2,3,4	1,2,3,4,7,8,9,10,11	NULL	2	-
4	2,3,4	1,2,3,4,7,8,9,10,11	NULL	4	-
5	5,6,7	1,2,3,4,7,8,9,10,11	5,6	4	2
6	5,6,8	1,2,3,4,7,8,9,10,11	5,6	5	2.5
7	5,7,8,9	1,2,3,4,7,8,9,10,11	5	6	6
8	6,7,8,9	1,2,3,4,7,8,9,10,11	6	1	1
9	7,8,9	1,2,3,4,7,8,9,10,11	NULL	0	-
10	8,10,11	1,2,3,4,7,8,9,10,11	NULL	1	-
11	10,11,12,13	1,2,3,4,7,8,9,10,11	12,13	2	1
12	10,11,12,13	1,2,3,4,7,8,9,10,11	12,13	5	2.5
13	11,12,13	1,2,3,4,7,8,9,10,11	12,13	6	3

ITERATION-5 : Select S_{11}.

LOCATION INDEX	SUBSET	I	S_i-I	COST	Ratio
1	1,2,3	1,2,3,4,7,8,9,10,11	NULL	4	-
2	1,2,3,4	1,2,3,4,7,8,9,10,11	NULL	5	-
3	1,2,3,4	1,2,3,4,7,8,9,10,11	NULL	2	-
4	2,3,4	1,2,3,4,7,8,9,10,11	NULL	4	-
5	5,6,7	1,2,3,4,7,8,9,10,11	5	4	4
6	5,6,8	1,2,3,4,7,8,9,10,11	5	5	5
7	5,7,8,9	1,2,3,4,7,8,9,10,11	5	6	6
8	6,7,8,9	1,2,3,4,7,8,9,10,11	NULL	1	-
9	7,8,9	1,2,3,4,7,8,9,10,11	NULL	0	-
10	8,10,11	1,2,3,4,7,8,9,10,11	NULL	1	-
11	10,11,12,13	1,2,3,4,7,8,9,10,11	12,13	2	1
12	10,11,12,13	1,2,3,4,7,8,9,10,11	12,13	5	2.5
13	11,12,13	1,2,3,4,7,8,9,10,11	12,13	6	3

ITERATION-6 : Select S_5.

LOCATION INDEX	SUBSET	I	S_I-I	COST	Ratio
1	1,2,3	1,2,3,4,7,8,9,10,11,12,13	NULL	4	-
2	1,2,3,4	1,2,3,4,7,8,9,10,11,12,13	NULL	5	-
3	1,2,3,4	1,2,3,4,7,8,9,10,11,12,13	NULL	2	-
4	2,3,4	1,2,3,4,7,8,9,10,11,12,13	NULL	4	-
5	5,6,7	1,2,3,4,7,8,9,10,11,12,13	5	4	4
6	5,6,8	1,2,3,4,7,8,9,10,11,12,13	5	5	5
7	5,7,8,9	1,2,3,4,7,8,9,10,11,12,13	5	6	6
8	6,7,8,9	1,2,3,4,7,8,9,10,11,12,13	NULL	1	-
9	7,8,9	1,2,3,4,7,8,9,10,11,12,13	NULL	0	-
10	8,10,11	1,2,3,4,7,8,9,10,11,12,13	NULL	1	-
11	10,11,12,13	1,2,3,4,7,8,9,10,11,12,13	NULL	2	-
12	10,11,12,13	1,2,3,4,7,8,9,10,11,12,13	NULL	5	-
13	11,12,13	1,2,3,4,7,8,9,10,11,12,13	NULL	6	-

ITERATION-7 : U is covered.

LOCATION INDEX	SUBSET	I	S_I-I	COST	Ratio
1	1,2,3	1,2,3,4,5,6,7,8,9,10,11,12,13	NULL	4	-
2	1,2,3,4	1,2,3,4,5,6,7,8,9,10,11,12,13	NULL	5	-
3	1,2,3,4	1,2,3,4,5,6,7,8,9,10,11,12,13	NULL	2	-
4	2,3,4	1,2,3,4,5,6,7,8,9,10,11,12,13	NULL	4	-
5	5,6,7	1,2,3,4,5,6,7,8,9,10,11,12,13	NULL	4	-
6	5,6,8	1,2,3,4,5,6,7,8,9,10,11,12,13	NULL	5	-
7	5,7,8,9	1,2,3,4,5,6,7,8,9,10,11,12,13	NULL	6	-
8	6,7,8,9	1,2,3,4,5,6,7,8,9,10,11,12,13	NULL	1	-
9	7,8,9	1,2,3,4,5,6,7,8,9,10,11,12,13	NULL	0	-
10	8,10,11	1,2,3,4,5,6,7,8,9,10,11,12,13	NULL	1	-
11	10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13	NULL	2	-
12	10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13	NULL	5	-
13	11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13	NULL	6	-

Hence, the **GREEDY WEIGHTED SET COVER APPROXIMATION** results in the set cover $\{S_9, S_{10}, S_3, S_8, S_{11}, S_5\}$ and its total cost is $0 + 1 + 2 + 1 + 2 + 4 = 10$. The optimum cost is 8 using the Linear Programming Method(Excel Solver).

APPLICATIONS OF THE SET COVER PROBLEM

- **Undirected Graph** : An undirected graph $G = (V, E)$ is a set system where V constitutes the universe and the edges E are the subsets of cardinality 2.
- **Wireless Coverage** : The universe consists all the locations on a college campus, and for each possible location of a wireless transmitter there is an associated region of the campus that is covered by placing a wireless transmitter at this location.
- **Solving Sudoku** : Solving Sudokus can be reduced to an exact cover problem (which is related to the set cover problem, but not exactly the same). This is however slower with respect to the classic backtracking with advanced searching heuristics.

CONCLUSION

- We have successfully studied the Set Cover Problem and its versions, and also discussed how Optimization is NP-hard while Decision problem is NP-Complete.
- We were able to come up with an algorithm of Weighted Set Cover to break ties. We provided a mathematical rigour to compare the greedy cover with respect to the optical cover. Dry run of the algorithms designed were successfully shown.
- We also showed some applications of this very important problem which is one of Karp's 21 NP-Complete problem. The report provides further insight and mathematical rigour to this topic and how certain steps can be further optimised.