Introduction

when k<12, we adopt Beam Search Algorithm. Instead of selecting only the single best option at each step, our approach maintains multiple promising candidates in parallel, effectively overcoming the classical greedy algorithm's weakness: getting trapped in local optima.

Step-by-Step Explanation

Now, let's see how this works, referring to the diagram you can see here:

Step 1: Expanding Paths (Gain Evaluation)

- At each step (**Layer 1** in the diagram), we consider multiple paths separately.
- For each existing path (e.g., paths with one K-block like {K1} or {K2}), we evaluate several potential next blocks (such as {K1,K2}, {K1,K5}, etc.).
- Each candidate block is assigned a **gain value**, which represents how many new subsets (**S**) this block can cover, beyond what's already covered.
- These candidate blocks are sorted from highest to lowest gain, and we retain the top α percent—typically **20-30%**—as candidates for the next step.

This α-cutoff step is crucial:

→ **It maintains multiple promising directions**, not just the single best one, allowing more thorough exploration.

Step 2: Beam Pruning (Global Evaluation)

- Next, all candidate blocks generated from each individual path (**Layer 2**) are collected together for a **global evaluation**.
- At this stage, each path is given an overall quality score, considering factors such as:
  - **Number of blocks used** (fewer blocks are preferred),
  - **Number of subsets (S) still uncovered** (fewer remaining uncovered subsets are preferred),
  - Coverage balance (even coverage across subsets)
- After scoring, all candidate paths are sorted again according to their scores.

- Only the top-scoring **n** paths (also called **beam width**) are preserved for further search. This beam width controls the breadth of our search and significantly improves performance by eliminating less promising directions.

In the diagram, this step is illustrated clearly by the arrows labeled **"Beam Evaluation,"** which select the highest-scoring candidate paths to enter **Layer 2** and beyond.

Step 3: Adaptive Beam Width

- To balance breadth and efficiency, we use an adaptive strategy for beam width:
    - In early iterations (**Layer 1** → **Layer 2**), we maintain a wider beam (e.g., 16 paths), encouraging broad exploration.
    - Later, as the search advances (**Layer 3** → **Layer 4**), we reduce the beam width (e.g., to 6 paths) to focus on the most promising solutions, thus optimizing resource usage.

Besides，To efficiently handle subset checking, we've adopted a simple yet powerful **bitwise representation** technique .The detailed steps won't be elaborated here

demonstration

here is the cases from the pdf
our results are as follows:
It should be noted that the fifth example (with a strict condition "at least 4 ") , our results is 12 groups, quite approaching to the nearly optimal solution 10