Uniform Random k-SAT Problem Generator

A. Objective

The objective is to generate random k-SAT instances with a specified number of variables, clauses, and clause size. Each generated instance contains m clauses, each of length k, with literals chosen randomly from n variables. These instances are intended for testing SAT solvers and analyzing algorithm performance. The generator ensures that each clause contains distinct variables or their negations.

B. Problem Statement

We are tasked with generating uniform random k-SAT problems for given parameters:

- k: number of literals per clause
- n: number of variables
- m: number of clauses

Each clause must contain k distinct variables or their negations. The total number of possible clauses is:

$$\binom{n}{k} \cdot 2^k$$

State Space: All possible combinations of m clauses of length k from n variables, with each literal potentially negated.

Goal State: No specific solution exists since the generator produces problem instances. The goal is to produce valid k-SAT formulas satisfying the input parameters.

Constraints:

1. $k \leq n$
2. Each clause contains unique variables
3. Each literal is randomly negated or positive

C. Methodology

State Representation

Let the set of variables be

$$V = \{x_1, x_2, ..., x_n\}$$

A clause C of length k is represented as a list of integers:

$$C = [l_1, l_2, ..., l_k]$$

where $l_i = v$ for a positive literal or $l_i = -v$ for a negated literal.

A formula F is a list of m clauses:

$$F = [C_1, C_2, ..., C_m]$$

Example: (3-SAT, 4 variables, 5 clauses)

$$F = [[1, -3, 4], [-2, 3, -4], [1, -2, 3], [-1, 3, -4], [1, 2, -3]]$$

Clause Generation Procedure

1. Randomly sample k distinct variables from 1 to n
2. For each variable, randomly assign positive or negative sign
3. Repeat for all m clauses

Mathematical Formulation

Let $F = \{C_1, C_2, ..., C_m\}$ and $C_i = \{l_1, ..., l_k\}$, then

$$C_i \subseteq \{-n, ..., -1, 1, ..., n\}, \quad |C_i| = k$$

and

$$\forall i, j \in C_i, \quad |i| \neq |j|$$

D. Pseudocode

```
FUNCTION Generate_k_SAT(k, n, m):
    clauses = empty list
    FOR each clause_index from 1 to m:
        vars_chosen = randomly select k distinct integers from 1..n
        clause = empty list
        FOR each variable v in vars_chosen:
            IF random choice is True:
                clause.append(v)       # positive literal
            ELSE:
                clause.append(-v)      # negated literal
        clauses.append(clause)
    RETURN clauses
```

```
FUNCTION FormatClauses(clauses):
    RETURN "-".join("[" + ",".join(str(l) for l in clause) + "]" for clause
```

Main Program Flow

1. Parse input parameters (k, n, m, optional seed)
2. Validate input ($k \leq n$)
3. Generate clauses using `Generate_k_SAT()`
4. Print the formula using `FormatClauses()`

E. Expected Results

- The program generates valid random k-SAT formulas as a list of clauses
- Each clause contains exactly k literals with no repeated variables
- Literals are randomly negated or positive
- Example output for $k = 3, n = 4, m = 5$:

$$[1, -3, 4][-2, 3, -4][1, -2, 3][-1, 3, -4][1, 2, -3]$$

- With a fixed seed, the same formula is reproducible
- These formulas can be used to benchmark SAT solvers or test algorithms

F. Conclusion

The random k-SAT generator provides a simple, reliable way to produce uniform random k-SAT instances. It respects the constraints of distinct variables per clause and random negations. By controlling k, n, and m, users can generate instances of varying complexity, supporting both educational and research purposes. This method allows the creation of benchmark datasets for SAT problem analysis, algorithm testing, and theoretical studies of combinatorial problem behavior.