State of the art - Hitting Set

Ana-Delia Manoliu, Alexandru-Cristian Grăjdeanu

March 14, 2025

1 Introduction

The **Hitting Set Problem** is a well-known problem in computer science and combinatorics that plays a significant role in the study of computational complexity. It is classified as an NP-complete problem, which means that it is in both the NP class (nondeterministic polynomial time) and as "hard" as any other problem in NP.

A problem is NP-complete if:

1. It is in the NP class: A proposed solution to the problem can be verified quickly (in polynomial time) by a deterministic algorithm;
2. It is NP-hard: Every problem in NP can be reduced to it in polynomial time. This means that if you can solve the NP-complete problem efficiently, you can solve all problems in NP efficiently.

1.1 W-hierarchy

The W hierarchy is a concept in computational complexity theory, which is a branch of computer science that studies how difficult problems are to solve.
The W hierarchy is a way to categorize problems based on their fixed-parameter tractability. It consists of levels like W[1], W[2], W[3], and so on, with each level representing a higher degree of complexity. Here's a simple breakdown:

1. W[0]: Problems that are fixed-parameter tractable (FPT). These are problems that can be solved efficiently if the parameter is fixed.
2. W[1]: Problems that are believed to be harder than those in W[0]. A classic example is the k-Clique problem (finding a clique of size k in a graph).
3. W[2]: Problems that are even harder, such as the Dominating Set problem (finding a set of k vertices that "dominate" all other vertices in a graph) or Hitting Set problem.

2 Problem formulation

Algorithm 1 Hitting Set Problem

Input: A set system / hypergraph $\mathcal{S}$

Output: A hitting set for $\mathcal{S}$. A hitting set is a set $H \subseteq V(G)$ such that every set $S \in E(\mathcal{S})$ contains at least one vertex from H, that is, for all $S \in E(\mathcal{S})$ we have $S \cap H = \emptyset$

3 Algorithms

3.1 Deterministic algorithm

Theorem 1 *Let $\Sigma = (S_1, ..., S_n)$ where S_i is a subset of $V = \{1, ..., n\}$ of size $|S_i| \geq R$. There is a deterministic algorithm that runs in O(nR) time and finds a subset $S \subseteq V$ with $|S| \leq (n/R) \ln n$ and $S \cup S_i \neq \emptyset$ for all i.*

The algorithm uses a greedy approach, starting with an empty set S and iteratively adding elements to S based on their frequency of appearance in the subset of Σ, and achieves the desired runtime by using a data structure that supports insertions, maximum value queries, and decrements in $O(\log n)$ time.

3.2 Randomized algorithm

Theorem 2 *Let $\Sigma = (S_1, ..., S_n)$ with each S_i a subset of $V = \{1, ..., n\}$ of size $|S_i| \geq R$. For any constant $C > 0$, there is a randomized algorithm that runs in time $O(n)$ and finds a subset $S \subseteq V$ with $|S| \leq (n(1+c)/R) \log n$, such that $S \cup S_i \neq \emptyset$ for all i holds with probability $\geq 1 - n^{-c}$. The algorithm does not need to know Σ.*

Both algorithms provide a solution to the hitting set problem, but the randomized algorithm has a faster runtime and a higher probability of success, while the deterministic algorithm provides a guaranteed bound on the size of the hitting set.

The probability of a set S missing a particular subset S_i is calculated using formula

$$(1 - s/n)^{|S_i|} \leq (1 - (C/R) \ln n)^R \leq n^{-C} = n^{-1-c} \tag{1}$$

and taking the union bound over Σ proves that S fails to be a hitting set with probability at most n^{-c}.

To return a hitting set of exactly the correct size, a random subset of V of size s is chosen and the probability for S to miss a particular subset S_i is calculated using the formula

$$\prod_{j=1}^{s} \frac{n - R - (j-1)}{n - (j-1)} \leq (1 - R/n)^s \leq n^{-C} \leq n^{-1-c} \tag{2}$$

The use of a randomized algorithm to obtain a hitting set can lead to a faster time algorithm of $O(m\sqrt{n})$ compared to the deterministic approach, which has a time complexity of $O(m\sqrt{n} + n^2)$, although the randomized algorithm may fail to obtain a good estimate with a small probability.

4 Benchmark instances

To test potential implementations of the Hitting Set problem, a dataset from the *PACE Challenges* will be used. The data consists of DIMACS-format files, specifically **.hgr** files for representing hypergraphs and **.sol** files for verifying whether the solution produced by the implementation matches the official solution.

Below is an example of a .hgr file and a .sol file.

`c I am a comment`
`p hs 6 5`
`1 2`
`2 3 4`
`2 4 5`
`1 3 6`
`5 6`
	`c The first non-comment line`
	`represents the solution size`
	`2`
	`2`
	`6`

.hgr file	**.sol file**

Table 1: Example of a .hgr file and a .sol file

References

[1] PACE Challenge. Hitting Set. Available at: https://pacechallenge.org/2025/hs/.

[2] CS 267 Lecture 5. Retrieved from: https://theory.stanford.edu/~virgi/cs267/lecture5.pdf.