CS 240 HW3

YUXIANG WEI

Problem 1: ILP for Optimal Edge Cover on Hyperbench Hypergraphs

1. ILP Formulation

For a hypergraph $H = (V, E)$, an *edge cover* is a subset $E' \subseteq E$ such that every vertex in V appears in at least one hyperedge in E'. We seek a minimum-size edge cover.

For each hyperedge $e \in E$, introduce a binary variable x_e indicating whether the edge is selected. The ILP is:

$$\begin{aligned} \min \quad & \sum_{e \in E} x_e \\ \text{s.t.} \quad & \sum_{e \in E : v \in e} x_e \geq 1, \quad \forall v \in V, \\ & x_e \in \{0,1\}, \qquad \forall e \in E. \end{aligned}$$

The covering constraint ensures that every vertex is contained in at least one chosen hyperedge, and the objective minimizes the number of selected hyperedges.

2. Implementation Summary

I implemented this ILP in Python using the PuLP library. Each hypergraph instance is given as a `.dtl` file from the hyperbench CQ dataset. Each line in a `.dtl` file describes one hyperedge by listing all of its vertices.

The program performs the following steps:

- parse each `.dtl` file into a list of hyperedges,
- identify all vertices appearing across these hyperedges,
- construct the ILP using one binary variable per hyperedge,
- add one covering constraint for each vertex,
- solve the ILP with PuLP's MILP solver,
- report the optimal edge-cover size and the selected hyperedges.

I applied the solver to the first ten hypergraphs in the dataset.

3. Results on the First Ten Hypergraphs

The table below reports, for `1.dtl`–`10.dtl`, the number of hyperedges and the optimal edge-cover size produced by the ILP solver:

File	#Edges	Optimal Edge-Cover Size
`1.dtl`	13	6
`2.dtl`	3	2
`3.dtl`	5	5
`4.dtl`	13	6
`5.dtl`	13	6
`6.dtl`	5	4
`7.dtl`	6	5
`8.dtl`	6	3
`9.dtl`	5	5
`10.dtl`	7	2

These results illustrate that some hypergraphs require selecting many hyperedges (e.g., `3.dtl` and `9.dtl`, where almost no hyperedges overlap), while others admit much smaller edge covers (e.g., `10.dtl` with optimal size 2).

Problem 2: Optimal fractional edge covers via LP

LP formulation

For a hypergraph $H = (V, E)$, a *fractional edge cover* assigns a nonnegative weight $x_e \geq 0$ to each hyperedge $e \in E$ such that every vertex is covered to total weight at least 1:

$$\forall v \in V : \quad \sum_{e \ni v} x_e \;\geq\; 1.$$

The cost of a fractional edge cover is $\sum_{e \in E} x_e$. The optimal fractional edge cover is obtained by solving the following linear program:

$$\begin{aligned} \min \quad & \sum_{e \in E} x_e \\ \text{s.t.} \quad & \sum_{e \ni v} x_e \geq 1, \quad && \forall v \in V, \\ & x_e \geq 0, && \forall e \in E. \end{aligned}$$

This LP is exactly the linear relaxation of the ILP used in Problem 1, where the integrality constraints $x_e \in \{0, 1\}$ are dropped and replaced by $x_e \geq 0$.

Implementation

I reused the parser for the Hyperbench CQ `.dtl` files from Problem 1 to obtain the set of vertices and hyperedges. For each hypergraph, I created one LP variable x_e for every edge e, added one covering constraint for every vertex, and set the objective to minimize the sum of all x_e. I used the PuLP library in Python together with the default CBC solver to solve each LP instance.

Experimental results on the CQ dataset

I ran the implementation on the first 70 hypergraphs in the CQ dataset (`1.dtl`–`70.dtl`). All of these instances solved quickly; in this range I did not encounter any timeouts.

For many hypergraphs, the optimal fractional edge cover coincides with the integral edge cover from Problem 1: all optimal weights are $x_e \in \{0, 1\}$ and the LP cost equals the minimum number of edges. For example:

- `1.dtl`: optimal fractional cost 6.0 with six edges of weight 1, matching the integral edge cover size 6.
- `10.dtl`: optimal fractional cost 2.0, again using two edges of weight 1.

However, there are also instances where the LP yields a strictly smaller fractional optimum than any integral edge cover. Typical examples are small symmetric graphs:

- `2.dtl`: the optimal fractional solution has $x_0 = x_1 = x_2 = 0.5$ and cost 1.5, while any integral edge cover needs at least 2 edges.
- `32.dtl` and `33.dtl` have similar symmetric structures and admit optimal fractional covers of total weight 1.5, again strictly below the corresponding integral edge cover sizes.

Overall, these experiments illustrate that the fractional edge cover LP is easy to solve on the CQ instances and that, as expected, its optimal value is always a lower bound on the size of an optimal (integral) edge cover, with equality for many but not all hypergraphs in the dataset.

Problem 3: Counterexample

I consider the hypergraph with vertices $\{0, 1, 2\}$ and edges

$$e_1 = \{0, 1\}, \quad e_2 = \{1, 2\}, \quad e_3 = \{0, 2\}.$$

This hypergraph has generalized hypertree width 1, but I show that it cannot have a width-1 GHD in which each hyperedge appears in exactly one bag.

Why this fails. If each bag contains exactly one hyperedge, then the bags for e_1, e_2, and e_3 must form a tree. However: - vertex 1 appears in the bags for e_1 and e_2, - vertex 0 appears in the bags for e_1 and e_3, - vertex 2 appears in the bags for e_2 and e_3.

To satisfy the running–intersection property, the bags containing each vertex must form a connected subtree. But here each pair of bags must be connected, producing a 3-cycle, and no tree can realize this. Therefore the required GHD does not exist.

ILP verification. I encoded the width-1 GHD constraints as an ILP (one binary variable per edge–bag assignment, plus connectivity constraints) and solved it using PuLP/CBC. The ILP was *infeasible*, confirming that the triangle hypergraph is a counterexample.

Conclusion. The claim is false: a hypergraph may have generalized hypertree width k but still fail to admit a width-k GHD where each hyperedge appears in exactly one bag.