John Wu jwu175 awang343
Alan Wang CookedPotato null

Project – IV

As a team of two people, we spent approximately 24 person-hours.

Model Description

Setup

- **Tests** - This was an array of binary variables representing whether or not we decide to test each symptom
- **XOR Matrix**: We created a 3d array of dimension `numDisease` by `numDisease` by `numTest`. Effectively, for each pair of diseases, we had a vector representing which tests could differentiate between those two diseases. This is precomputed.

Constraints and Objective

- **All Differentiable** - For each unique pair of diseases, we had to check that the dot product of which tests are activated with the XOR vector in the XOR Matrix was ≥ 1. This guarantees that there is at least one active test that differentiates every pair of diseases.
- **Minimized Cost** - We took the dot product of the active tests vector and the cost vector to obtain the cost for a given assignment of tests. Our model aimed to minimize this.

Relaxation to LP

We relaxed the binary variable for tests into a continuous variable from 0 to 1. This model produces a lower bound on the cost of testing, and we were able to use it at every node in our search tree to get a cost lower bound given a set of forced assignments.

B&B Strategy

To convert our LP solution into an ILP solution, we implemented multiple variants of the Branch and Bound (BnB) algorithm to search for the optimal integer-feasible solution.

Branch and Bound

We explored two primary BnB strategies: Depth-First Search (DFS) and Best-First Search (BFS), each enhanced with simple but effective heuristics.

1. **Depth-First Search (DFS)** — In our DFS-based BnB, we used a stack to manage the incremental modifications needed at each node, allowing us to efficiently update the LP model with newly fixed variable constraints. This approach avoided reinitializing and resolving the entire LP from scratch at each step.

   When we encountered an integer-feasible solution, we updated the incumbent cost and aggressively pruned subtrees that could not yield better solutions. To prioritize more promising branches, we employed a greedy strategy that explored nodes with lower LP relaxation costs earlier in the search.

   For variable selection, we experimented with several simple heuristics, including:

- Selecting the variable whose LP value is **closest to 0.5**, targeting variables with the highest uncertainty to reduce the branching factor early.
- Selecting the variable with the **smallest LP value**, to quickly confirm variables that are likely to be zero and potentially simplify constraints.
- Other variations like furthest from 0.5, largest value, etc., though they were less effective.

We found the "closest to 0.5" and "smallest value" heuristics most effective in practice.

2. **Best-First Search (BFS)** — In our BFS variant, we maintained a priority queue ordered by the LP relaxation cost, always exploring the node with the lowest objective value next. This required us to store the full LP state and objective at each node, as we could no longer rely on efficient in-place LP updates like in DFS.

   For variable selection, we reused the same heuristics as in the DFS strategy to maintain consistency in how we navigated the decision tree.

Observations

1. We were able to get full coverage with these simple heuristics and with DFS + BFS.
2. Since we only need 1 test to differentiate between diseases, we posited that the LP solution values would be relatively close to 0. In other words, we would get more information about what tests *shouldn't* be included compared to what should. This is why the heuristics of smallest LP performs better on more instances than highest LP.
3. There seemed to be a CPLEX bug where the solution to an LP with constraint $x_1 = 1$ would allow for solutions where $x_1 = 0.999999998$. In other words, there seemed to be a floating point variable storage error. This forced us to redesign our heuristic calculations to be robust because we didn't want to branch on an already assigned variable that wasn't equal to 1.
4. One thing we noticed was that recently, Brown's CS Dept machines updated their Python version to 3.11.2. This is incompatible with CPLEX, and seems to have broken our compile.sh. There is no global Python 3.9 to copy anymore when installing a new environment with uv.