We encode both the Dominating Set and Hitting Set problems in CP-SAT by representing each element (in the Hitting Set problem) or each vertex (in the Dominating Set problem) as a Boolean variable, which is set to true if the corresponding element or vertex is selected.

In the Hitting Set problem, each input set is translated into a hard OR constraint over the variables corresponding to its elements. This ensures that at least one element from each set is selected—that is, each set is "hit."

In the Dominating Set problem, each vertex is associated with a hard OR constraint that includes the variable for the vertex itself and those of its neighbors. This guarantees that every vertex is either selected or adjacent to a selected vertex, thereby satisfying the domination condition.

To enforce minimality, we define a linear objective function that minimizes the total number of variables set to true. The CP-SAT solver is then used to find an exact, optimal solution that satisfies all hard OR constraints while minimizing the size of the selected set.

Our goal in this project is to explore the capabilities of the CP-SAT Solver, part of the OR-Tools package. As such, the only reduction applied is the removal of syntactically identical clauses. This simplification does not affect the solution space or the objective function for either the Dominating Set or Hitting Set problem, and therefore no further proof of correctness is required.

To solve the problems, we use only full subsolvers, which are guaranteed to explore the entire solution space and prove optimality. The solvers are configured with different settings depending on the problem type and instance behavior:

For both problems, we use the no_lp subsolver with linearization_level = 0, disabling the LP relaxation. Additionally, we enable optimize_with_core = true, which applies a core-based optimization strategy similar to MaxSAT solvers where it tries to increase the lower bound instead. This configuration solves approximately 80% of the instances optimally.

For the remaining unsolved Hitting Set instances, we apply the pseudo_costs subsolver, using linearization_level = 2 to enable a richer LP relaxation, search_branching = PSEUDO_COST_SEARCH for branching guided by historical pseudo-costs, and exploit_best_solution = true to prioritize branching in line with the last best-known solution. These settings work on the remaining instances, which is how we reach 100% on OPTIL.io.

Ideally, both subsolvers would be run in parallel, but since this is not allowed, we execute them in series, dividing the total runtime budget equally between them.

For the Dominating Set problem, we rely solely on the no_lp configuration. While additional settings are tested to improve runtime, they offer only marginal benefit in this context.

For the heuristic tracks of both problems, we use the solver’s built-in Large Neighborhood Search (LNS) strategy, which explores promising regions of the solution space by relaxing and re-optimizing subsets of variables.