Google OR-Tools

This repository contains samples and excercises for constraint optimization with Google OR-Tools using dotnet.

01_SimpleConstraints

This is no excercise but just a basic sample which was showed in the presentation.

It consists of a simple model containing variables with constraints and an optimization.

02_SudokuSolver

This is an excercise to create a solver for Sudokus, as showed in the presentation.

- **SudokuSolver**: contains the solver with helper-classes. Your task is to program the model in `SudokuConstraintsSolver.cs`
- **SudokuSolverTests**: contains unit-tests which are currently failing. Use them to guide you through the implementation of the solver and make them green - one by one.

In case you don't know Sudoku: https://de.wikihow.com/Sudokus-l%C3%B6sen

In case you're stuck, you can have a look at the solution: SudokuSolver/Solution

03_MinesweeperSolver

This is a more advanced and more open excercise.

- **MinesweeperSolver**: contains the solver with helper-classes. Your task is to program the model in `MinesweeperConstraintsSolver.cs`
- **MinesweeperSolverTests**: contains unit-tests which are currently failing. Use them and add new ones to guide you through the implementation of the solver and make them green - one by one.
- **Minefield**: Provides useful methods to deal with the minefield
  - `Minefield.GetAllCells()` provides all cells
  - Cells next to mines are of the type `CellWithMineDetector`
  - `Minefield.GetCellsInDetectionRadius(cellWithMineDetector)` provides all cells in the radius of the given cellWithMineDetector
  - Once you found the solution, you can mark a field with a mine: `Minefield.PlaceMine(x, y)`

In case you don't know Minesweeper: https://de.wikihow.com/Minesweeper-spielen

In case you're stuck, you can have a look at the solution: MinesweeperSolver/Solution

Cheat-Sheet

Create the model: Variables, Constraints, Objectives

Code	Explanation
`var model = new CpModel()`	Creates a model
`var x = model.NewIntVar(0, 9, "x")`	Creates a new integer-variable in the model
`var constraint = model.Add(x == y)`	Adds a new constraint to the model: x should be equal to y
`var constraint = model.Add(LinearExpr.Sum(variables) == 7)`	Adds a new constraint to the model: The number of variables should be equal to 7
`model.Maximize(x)`	Sets the objective of the model to maximize x
`model.AddAllDifferent(variables)`	Adds a global constraint requiring all variables in the list to be different

Solve and evaluate

Code	Explanation
`var solver = new CpSolver()`	Creates a solver
`var status = solver.Solve(model)`	The solver will search for solutions to the model. The status tells whether none (INFEASIBLE), some (FEASIBLE) or all (OPTIMAL) solutions have been found
`solver.Value(x)`	After `Solve(model)` the solution can be inspected

Links

- Getting started with C# https://developers.google.com/optimization/introduction/dotnet
- Google OR-Tools on Github (including samples) https://github.com/google/or-tools
- API documentation (in Python) https://google.github.io/or-tools/python/ortools/sat/python/cp_model.html