Solvinery – Manual :

Basic Overview

The project enables anyone to easily access powerful tools to solve planning problems such as scheduling, task management, and more. Choose a scenario from the shared public library, edit it by enabling or disabling constraints, adjusting priorities, and editing the static data, and send it to the server who uses a powerful engine to find the optimal solution. The project will initially feature a number of select scenarios that will cover many of the most common problems. However, anyone willing to learn how can create a new one from scratch and help expand our ecosystem by sharing it.

Technical overview:

Solvinery is a linear programming interface designed to allow users to change existing linear programming model data without actually looking at the problem itself. It does so using parts of SCIP Optimization Suite, specifically, it uses Solving Constraint Integer Programs (SCIP) as the solving engine, and Zuse Institute Mathematical Programming Language (Zimpl) to define the problem for the engine. Read about them here: https://scipopt.org/index.php#welcome It has three main features for each registered user: viewing, editing and solving owned images, viewing, searching and downloading new images from the public repository and creating new images from scratch, using raw zimpl code as a starting point. A new image's quality depends mostly on the quality of the zimpl code it's made from, thus it's not recommended to create new images unless no image in the existing repository solves your issue.

Usage Guide

Using Images

Every image consists of the following components:

- **Profile**

its own profile (name, description, etc.), sets, parameters, constraints, and preferences

- **Sets**

An image set defines data that one can edit in the problem. A set is an unordered list of text,integer, decimal, or tuple elements. A tuple is itself an ordered list of the same element types a set supports, which means a set can consist of a collection of lists. A set is displayed in the image in a table format, where each column's head holds the name of the component below it.

- **Parameters**

An image parameter defines a single element, which may be an integer, decimal, or text value. Like sets, a parameter's value may have a name of its own(i.e., a parameter called Salary with its value called Dollars)

- **Constraints**

A logical constraint that is applied to the problem when it's solved. May be toggled on or off

- **Preferences**

A part of the actual problem that's being optimized. Each preference may have a 0–1 scale, meaning one can define a priority for each preference compared to others.

Every image may be solved at any point, that means it's sent to the server to have the engine solve it and return its solution, if such exists. An image may also be edited at any point, be warned though, one can easily break an existing image with the wrong change, so at least rudimentary knowledge is recommended before editing any logical component of an image. An image may also be published and deleted at all times.

Solution VIew

Once an image is solved, a solution is shown on the screen. Every solution consistest of a list of topics that were solved, each topic consists of either a list of solutions or a numeric value. If there's a list of solutions, and there are exactly two or three columns, they may be pivoted into a table view, where each column in the solution may be assigned to a row, column, or cell of the table. A three-column solution may also be viewed as a graph.

Shared Images

By default, the first 5 images, sorted by creation date, are displayed. The user may filter, according to an image name, description, and author, and view images created before or after a specific date. An image viewed here may at any point be downloaded, at which point it will be added to the user's image list.

Image Creation

- **Zimpl code**

An image is initially made from parsed zimpl code. As of now we do not support parsing the full score of zimpl's syntax, thus you must adhere to the following rules: Only "primitive" sets and parameters will be parsed and available in the image. Sets that won't be parsed include: sets with more the one layer deep recursive tuples, indexed sets, indexed parameters, and any of their derivatives. As a general rule of thumb, if you try to break our parser, you'll probably succeed in doing so — so try to keep the syntax simple. Most of zimpl's complex data structures may be defined in a simpler way with multiple sets.

- **Image definition**

The most important part of an image is its variables — these are the actual topics displayed in the solution. Each variable may be selected, where unselected variables are not displayed in the solution, have an alias, overriding its original name, and a structure alias, overriding the names of the parts making up each section of its solution. Next, there are sets and parameters; like a variable, they may have an alias and structure alias. Finally, we have constraints and preferences — for each of those one can to create a module, where each module is made up of a name, description and the actual constraints and preferences that are managed by it. THe preferences and constraints making up modules are actual code snippets from the underlying zimpl code.