Reasoning Agents project

Davide di Ienno • XX.02.2022

Outline

- Topic of the project
- Parser
- Game generator
- Solver

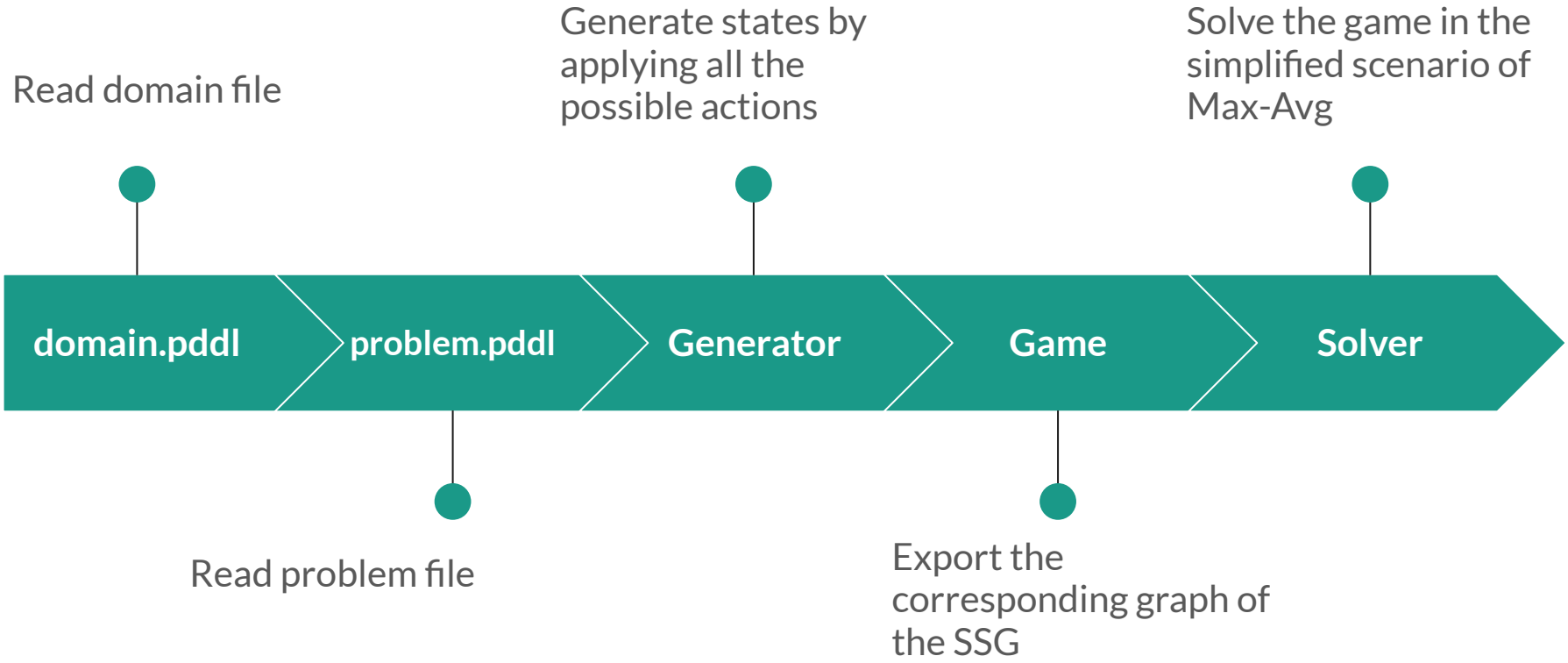
Topic of the project

From Planning

Starting from a probabilistic planning problem (main reference were prob-benchmarks of PRP).

To Games

Obtain from it a Simple Stochastic Game, in the simplified case of only one player, and solve it.


Read domain file
Generate states by applying all the possible actions
Solve the game in the simplified scenario of Max-Avg
domain.pddl
problem.pddl
Generator
Game
Solver
Read problem file
Export the corresponding graph of the SSG

Parser - Domain

Predicates

Read the predicates and their key characteristics.

Predicates that are not related to objects make it easier to check preconditions

Actions

- Parameters
- Preconditions
- Effects

Parser - Domain: Actions

Parameters

Definition of the parameters involved in preconditions and effects

Preconditions

All conditions that must be verified in the given state to apply the action.

Not all the keywords of pddl were implemented.

Parser - Domain: Actions

Effects

They can have a probability attached to them.

Sometimes, only a subset of the effects are probabilistic.

They can also have only effects with probability $p < 1$, implying that there is a probability 1-p that the state will remain unchanged.

Parser - Problem

Objects

All the objects in the world are defined here (they are key to check preconditions).

Init state

The initial state is defined using predicates and objects.

Goal

The goal condition is defined too, in a similar way to the init state.

Generator - Actions

Verify Preconditions

Given a state, verify if there are actions which can be applied; if there are any, return the corresponding objects.

Order

Speed up the process by checking conditions in ascending order of complexity.

Generator - States

New states

Starting from the initial state, generate new ones by applying possible action.

The Algorithm is basically a breadth-first.

Unique

Duplicate states are not accepted.

If a duplicate is discovered, the old one is used instead.

Generator - States

Max

Each state has a corresponding Max node in the resulting SSG.

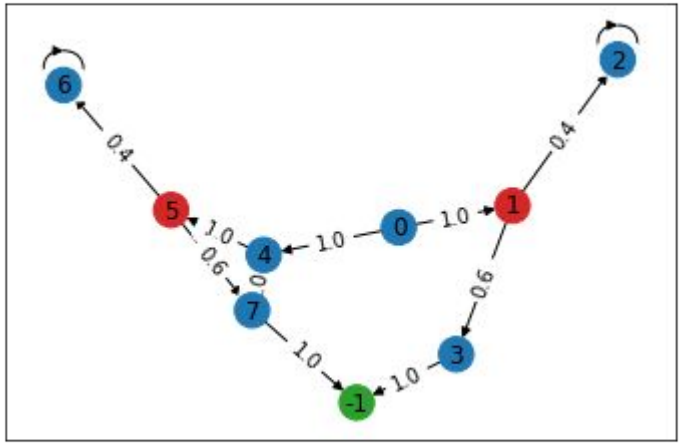
Average

If an action have probabilistic effects, the choice of do such action is modelled through an Average Node.

Generator - SSG

Climber

Output for the climber problem represented with the aid of NetworkX.



Solver - Max and Avg

Input : A simple stochastic game $G = (V, E)$ with only AVE and MAX vertices
Output: The optimal value vector v_{opt}

begin

Minimize $\sum_{x \in V} v(x)$,

subject to the constraints:

$$
\begin{array}{ll}
v(x) \geq v(y) & \text{if } x \in V_{\text{MAX}} \text{ and } (x, y) \in E \\
v(x) \geq \frac{1}{2} \cdot v(y) + \frac{1}{2} \cdot v(z) & \text{if } x \in V_{\text{AVE}} \text{ and } (x, y), (x, z) \in E \\
v(x) = p(x) & x \in \text{SINK} \\
v(x) \geq 0 & x \in V
\end{array}
$$

Solve this linear program to obtain an optimal value vector v. Output v.

end

From: Elena Valkanova. Algorithms for simple stochastic games. Master's thesis, University of South Florida, May 2009

Solver - Max and Avg

Generalization

More general approach for Average nodes: not limited to two options with 0.5 probability.

PuLP

Implemented thanks to PuLP library, dedicated to solving Linear Programming problems.

Thanks for the attention