Search and Planning 2022/23
Project – MAPF with CSP

19/09/2022

Overview

The Search and Planning project is to develop an encoding for solving the Multi-Agent PathFinding (MAPF) problem as a constraint satisfaction problem (CSP).

Problem Specification

MAPF finds a wide range of practical applications, that include robot routing, automated planning, network routing, GPS navigation, combinatorial problems (including puzzles), etc. Different variants of MAPF exist, and MAPF can also be referred to as cooperative pathfinding or multi-robot path planning, among other names.

The variant of MAPF considered in this project is described as follows.

1. Let $G = (V, E)$ be an undirected graph.
2. Let A be a set of agents, with $|A| < |V|$.
3. The injective function $\pi_S : A \rightarrow V$ denotes the start position.
4. The injective function $\pi_G : A \rightarrow V$ denotes the goal position.
5. Moves of the agents are organized in time steps.
6. At each time step i each agent can only move to an adjacent empty vertex at time step $i-1$.
7. Any number of agents can move (to empty positions) at any given time step.
8. The goal of MAPF is to minimize the *makespan*, i.e. to compute the minimum number of time steps that are required to reach the goal position from the start position.

Project Goals

You are to implement a tool, or optionally a set of tools, invoked with command `proj`, that should take two command line arguments:

1. The first argument denotes the name of a file representing a graph;

2. The second argument denotes the name of a file representing a scenario.

This set of tools should use the MiniZinc CSP solver (version 2.6.4 available from https://www.minizinc.org/) to compute the makespan given the target graph and scenario, and output a sequence of moves for the agents to move from the start position to the goal position.

The tool is expected to be executed as follows:

```
proj <graph-file-name> <scenario-file-name> > solution.txt
```

The tool should write to the standard output, which can then be redirected to a file `solution.txt`.

The programming language to use is Python version 3.10.5.

Formats

For the input formats, a number of lines at the beginning of the files may denote comments, consisting of lines that start with character '#'.

Graph Format

A standard text representation of graphs is used. The first line consists of one integer $|V|$ denoting the number of vertices. The second line consists of one integer denoting the number of edges $|E|$. Each of the following lines consists of two integers, representing each of the $|E|$ edges.

An example of an undirected graph is shown below.

```
5
4
1 2
2 3
3 4
2 5
```

Scenario Format

The file with the scenario consists of three parts, in order. The first part is a single line, consisting of a single integer, that denotes the number of agents $|A|$. The second part starts with a line with string *START:*. Afterwards, there are $|A|$ lines, each consisting of two integers, the first representing one agent and the second representing the initial vertex where the agent starts. The third and final part starts with a line with string *GOAL:*. Afterwards, there are $|A|$ lines, each consisting of two integers, the first representing one agent and the second representing the goal vertex where the agent is to be positioned.

An example of a scenario for the example graph above is shown below.

```
3
START:
1 1
2 2
3 5
GOAL:
1 5
2 2
3 1
```

Output Format

The output is a sequence of $M+1$ lines, where M denotes the computed makespan. Each line starts with an indication of the time step, of the form 'i=<number> '. Afterwards, each line consists of a sequence of pairs of integers separated by ':', each separated by tabs or spaces, and where the first integer denotes the agent index and the second integer denotes the vertex whether the agent is positioned.

An example of the expected output format for the example graph and scenario above is shown below.

```
i=0    1:1  2:2  3:5
i=1    1:1  2:3  3:5
i=2    1:2  2:4  3:5
i=3    1:3  2:4  3:5
i=4    1:3  2:4  3:2
i=5    1:3  2:4  3:1
i=6    1:2  2:4  3:1
i=7    1:5  2:3  3:1
i=8    1:5  2:2  3:1
```

Examples

Example 01

The input graph is given by:

```
5
4
1 2
2 3
3 4
2 5
```

The input scenario is given by:

```
3
START:
1 1
2 2
3 5
GOAL:
1 5
2 2
3 1
```

An example of a valid output is the following:

```
i=0    1:1  2:2  3:5
i=1    1:1  2:3  3:5
i=2    1:2  2:4  3:5
i=3    1:3  2:4  3:5
i=4    1:3  2:4  3:2
i=5    1:3  2:4  3:1
i=6    1:2  2:4  3:1
i=7    1:5  2:3  3:1
i=8    1:5  2:2  3:1
```

Example 02

The input graph is given by:

```
# Example comment line
13
15
1 2
1 4
```

```
2 5
4 5
4 7
5 8
7 8
7 10
8 11
10 11
11 12
12 13
3 6
6 9
9 13
```

The input scenario is given by:

```
# Example comment line
4
START:
1 1
2 5
3 7
4 11
GOAL:
1 3
2 6
3 9
4 13
```

An example of a valid output is the following:

```
i=0    1:1  2:5  3:7  4:11
i=1    1:2  2:8  3:4  4:10
i=2    1:5  2:11  3:1  4:7
i=3    1:8  2:10  3:2  4:4
i=4    1:11  2:7  3:1  4:5
i=5    1:12  2:10  3:4  4:5
i=6    1:13  2:11  3:7  4:8
i=7    1:9  2:12  3:10  4:8
i=8    1:6  2:13  3:11  4:8
```

```
i=9    1:3  2:9  3:12  4:8
i=10    1:3  2:6  3:13  4:11
i=11    1:3  2:6  3:9  4:12
i=12    1:3  2:6  3:9  4:13
```

Additional Information

The project is to be implemented in groups of one or two students. Registration through Fenix is required.

A number of papers covering the MAPF problem are available on the course's website [1, 2, 3, 4].

The project is to be submitted through the course website. You should submit a zip archive (with name <group_number>.zip) with the source code of your solution and a one-page document summarizing the proposed solution should be included in the archive. Acceptable formats for this document are txt and PDF.

The evaluation will be made taking into account correctness given a reasonable amount of CPU time (70%) and efficiency (30%).

If needed, an oral evaluation for specific groups may take place.

The input and output formats described in this document must be strictly followed.

Code Plagiarism Detection

The same or very similar projects will lead to failure in the course and, eventually, to the lifting of a disciplinary process. The projects will also be tested against existing web solutions. Analogies found with web programs will be treated as fraud.

Project Dates

- Project published: 19/09/2022.
- Fenix group registration: 30/09/2022.
- Project due: 21/10/2022.

Omissions & Errors

Any required clarifications will be made available through the course's official website.

References

[1] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artif. Intell.*, 219:40–66, 2015.

[2] G. Sharon, R. Stern, M. Goldenberg, and A. Felner. The increasing cost tree search for optimal multi-agent pathfinding. *Artif. Intell.*, 195:470–495, 2013.

[3] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. S. Kumar, R. Barták, and E. Boyarski. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *SOCS*, 2019.

[4] J. Yu and S. M. LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *AAAI*, 2013.