Search and Planning 2023/24
Project – Patient Transportation Problem as a CSP

11/09/2023

Overview

The Search and Planning project is to develop an encoding for solving the Patient Transportation Problem (PTP) as a Constraint Satisfaction Problem (CSP).

Problem Specification

According to the specification of problem 082 in CSPlib [1], the patient transportation problem consists in transporting patients to medical appointments given a fleet of vehicles.

The fleet is heterogeneous and is mainly composed of ambulances and private drivers operating as volunteers. Each patient has a set of characteristics and is represented by a request. **The objective is to satisfy as many requests as possible** within a fixed horizon, which is typically bounded by the working hours.

Three aspects of decision are considered in the PTP: (1) selecting which requests to service, (2) assigning vehicles to requests and (3) routing and scheduling appropriately the vehicles.

Some specific characteristics must also be considered in the PTP. Here are some of them:

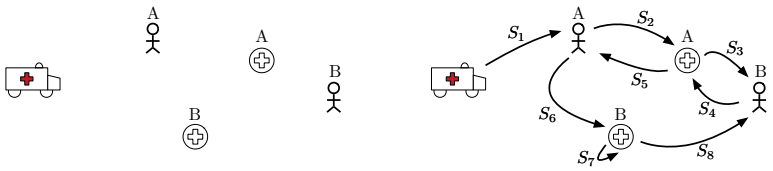
- Patients can have several constraints such as a maximum travel time or a maximum waiting time at the hospital. The time to embark and disembark a patient must sometimes be considered.
- The set of requests is heterogeneous. Some patients only require to go from their home to a health center, while some of them also need a return trip once the care has been delivered. In the latter case, they must be taken back home, or to another place if requested. It is also possible to have patients requiring only a return trip. Besides, requests can involve more than one passenger at once. For instance, a child can be accompanied by his parents.
- The vehicle fleet is heterogeneous. Vehicles can differ by their capacity, their initial/final location (typically a depot) and their availability. Some patients can only be taken by particular vehicles. For instance, patients in wheelchairs can only be transported by specific vehicles.

[1] https://www.csplib.org/Problems/prob082/

- Availability of vehicles can be non-continuous. For instance, they can be available from 9am to 1pm and from 3pm to 6pm.

Note that this version of PTP is static: the whole set of requests is known beforehand and no new request is added in real time.

An illustration of the PTP on a toy example with two patients (A and B) and a single vehicle is shown in Fig. 1. Note that patients A and B have to be displaced to different hospitals. A possible solution consists in the following sequence: taking A (S1), bringing A to hospital A (S2), taking B (S3), taking back A with B in the vehicle (S4), dropping A to its home (S5), bringing B to hospital B (S6), waiting for B (S7) and dropping B to its home (S8).



(a) The initial situation. (b) A possible solution.

Fig. 1: Illustration of the PTP with one vehicle and two patients.

Project Goals

You are to implement a program in Python in file `proj.py`, that should take as argument the name of a file representing the input data.

This program should use the MiniZinc CSP solver (version 2.7.6 available from https://www.minizinc.org/) to compute the number of patient requests to be satisfied given the input data, and output the number of requests satisfied and the corresponding sequence of actions. For solving the PTP, consider using the `cumulative` global constraint [2].

Your program is expected to be executed as follows:

```
python proj.py <input-file-name> <output-file-name>
```

For each specific input file, your program should write the solution to a JSON file. More information regarding the input and output format is provided in the next section.

The programming language to use is Python version 3.10.12.

[2] https://www.minizinc.org/doc-2.7.6/en/predicates.html#cumulative

Formats

Input Format

For the input, the data is provided as a JSON file having the following format:

- The "maxWaitTime" field contains the duration of the time windows during which the patient has to be transported. It is expressed as a string having the following format: "HHhMM"
- The "places" field contains all the distinct locations of the problem. For each of them:
  - "id" is the ID of the location. It corresponds to its position in the list of locations.
  - "category" is the category of the location. Its value can be 0 for a medical center, 1 for a vehicle depot and 2 for a patient location.
- The "vehicles" field contains all the vehicles available in the fleet. Each of them has the following fields:
  - "id" which contains the ID of the vehicle.
  - "canTake" which is a set of categories of patients that the vehicle can take.
  - "start" and "end" which are the IDs of the starting and ending depots of the vehicle. A -1 value indicates that the vehicle has no depot.
  - "capacity" which is the capacity of the vehicle.
  - "availability" which is a set of time windows when the vehicle is available. Each time window is encoded by a string having the following format: "HHhMM:HHhMM".
- The "patients" field contains all the requests. Each patient/request has the following fields:
  - "id" which contains the ID of the request.
  - "category" which indicates the category of the patient.
  - "load" which indicates the number of places required in the vehicle during the transport.
  - "start", "destination" and "end" which indicate the IDs of the starting location, medical center and return location for the patient, respectively. "start" or "end" can have a -1 value in case of single trip.
  - "rdvTime" and "rdvDuration" indicate the start of the appointment and its duration as strings under the format "HHhMM". In terms of constraints, the patient must be picked up at its starting location for its forward trip at or after rdvTime - maxWaitTime. They must be dropped at their destination before or at rdvTime. They must be picked up for their backward trip at the medical center after or at rdvTime + rdvDuration and must be dropped at their end location before or at

`rdvTime` + `rdvDuration` + `maxWaitTime`.

  - "`srvDuration`" indicates the time needed for the patient to embark/disembark the vehicle. It is encoded as a string under the format "`HHhMM`".

- The "`distMatrix`" field contains the distance (in minutes) matrix between the locations. It follows the IDs and order of the locations (`distMatrix[2][3]` is the distance from location 2 to location 3).
- The "`sameVehicleBackWard`" field indicates if the same vehicle has to be used for the two trips of a back and forth request.

Output Format

For the output, your program should provide a JSON file having the following format:

- The "`requests`" field is the objective of this problem and corresponds to the total number of satisfied requests.
- The "`vehicles`" field, a list containing all the vehicles available in the fleet. Each of them has the following fields:
  - "`id`" which contains the ID of the vehicle.
  - "`trips`" which is the list of trips made by the vehicle, in chronological order. Each trip has the following fields:
    * "`origin`" and "`departure`" which indicate the IDs of the origin and departure locations.
    * "`arrival`" which indicates the arrival time under the format "`HHhMM`".
    * "`patients`" which is a set of IDs associated with the transported patients during the trip.

A practical example, follows:

```
{
  "requests" : 11,
  "vehicles" : [ {
    "id" : 21,
    "trips" : [ {
      "origin": 1,
      "destination": 5,
      "arrival": "15h34",
      "patients": [ ]
      }, {
      "origin": 5,
      "destination": 10,
```

```
      "arrival": "15h50",
      "patients": [ 22, 25 ]
      },
    ...
    ] },
...
] }
```

Additional Information

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

Some documents covering the patient transportation problem are available on the course's website [1, 2].

The project code 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.

You should also submit a video with the maximum duration of 4 minutes. In that video, all the members of the group should appear and the work developed should be explained, including the output obtained for specific problem instances.

The evaluation will be made taking into account (i) correctness given a reasonable amount of CPU time (60%), (ii) quality of the solution (20%) and (iii) video (20%).

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: 11/09/2023.
- Fenix group registration: 22/09/2023.
- Project due: 20/10/2023 (code 17h00 and video 23h59).

Checkpoints

There will be a few checkpoints in the practical classes, although no evaluation will be given. The ckeckpoints are as follows:

1. Week 25-29 September: read input, define variables.
2. Week 2-6 October: implement constraints.
3. Week 9-13 October: guarantee correctness.
4. Week 15-20 October: improve performance.

Omissions & Errors

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

References

[1] Q. Cappart, C. Thomas, P. Schaus, and L. Rousseau. A constraint programming approach for solving patient transportation problems. In J. N. Hooker, editor, *Principles and Practice of Constraint Programming - 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings*, volume 11008 of *Lecture Notes in Computer Science*, pages 490–506. Springer, 2018.

[2] C. Thomas. *Advanced modelling and search techniques for routing and scheduling problems*. PhD thesis, Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2023.