CS4402 Constraint Programming

Assignment: **Practical 1: Automated Warehouse Picking**

Credits: 50% of the coursework component

Deadline: 28/10/2024 - 21:00

(NB MMS is the definitive source for deadlines and weights)

Aim / Learning objectives

This practical tests your ability to construct and evaluate constraint models. Your goal is to find a sensible approach and to produce a clear, concise, understandable model and text documenting your effort.

Submission

The deliverables consist of a *single zip file* containing, *at least*:

- A report, the contents of which are specified below.
- A file named `model.eprime`, containing the Essence Prime constraint model for "The Automated Warehouse" problem specification described below.

Please read this document *carefully*. It describes several requirements for the problem being modelled, and the way in which your solution and experiments should be reported.

The Automated Warehouse

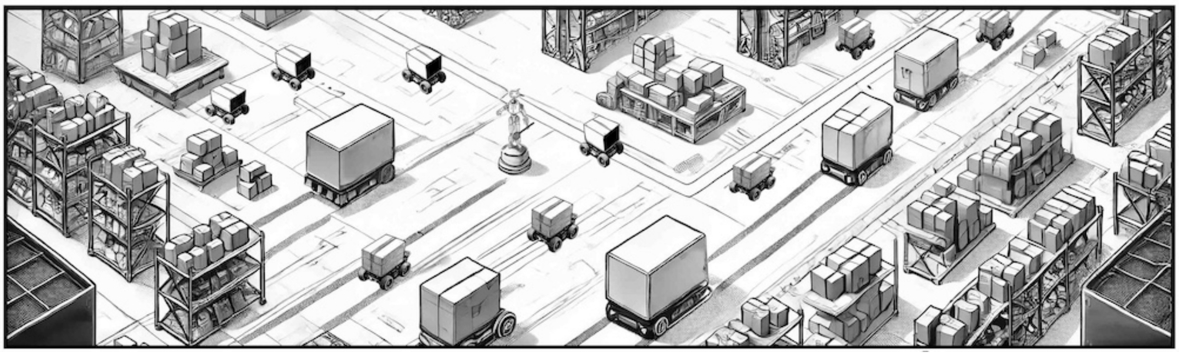


Figure 1: Image from https://www.leagueofrobotrunners.org/

As an optimization and scheduling expert, you have been hired as a consultant by a well-known logistics company that operates internationally. The company is trying to automate their warehouse operations and aims to invest in a fleet of robots. The task you are going to solve is to manage a fleet of robots that pick products from a warehouse and bring them to the people packaging them.

In essence, regarding the map:

- The operations happen in a pre-mapped warehouse, which is divided into a grid of size M x N where M are the number of rows and N the number of columns.
- In the map, empty cells (which are traversable) are represented as a 0, while impassable cells are marked with a -1.
- The different piles of objects stored in the warehouse are identified with positive integers.

The rest of the constraints:

- You receive a list of tasks. In each task, a robot must pick up an item from the specified pile and deliver it to the target location to complete that task.
- Each robot has a starting position.
- Robots move in one of four directions: up, down, left and right.
- Robots do not need to move in each time step; they can wait by remaining in the same cell.
- You are given a maximum time limit. That is, the maximum number of moves **any** robot can do.
- Robots can carry only one object at a time.
- Robots start with their inventory empty.
- Once picked up, a robot cannot drop an object unless it is in the task target location.

Note that robots can move through cells containing object piles, regardless of whether they pick up an item or not (see the example).

A **solution** is characterized as, for each robot, the list of positions (the path) of where it is in each time step.

The task is split into 3 different parts, increasing in complexity:

Part 1: Basic Order Assignment and Routing:

In this part, your task is to create a model for the previously described problem.

Part 2: Complex movement:

In this part, we will also consider the following constraint:

- Robots must avoid collisions with each other.

There are two kinds of possible collisions:

Cell conflict: A cell conflict occurs if two robots occupy the same position at the same step. That is, no two robots can be in the same cell at the same step.

Swapping conflict: A swapping conflict between two robots occurs when two robots swap locations in a single time step. That is, if robot A is in <1,2> and robot B is in <2,2> in timestep 4, a conflict arises if in timestep 5 robot A is scheduled to be in <2,2> and robot B in <1,2>.

Part 3: Optimal Robot Order Assignment and Routing:

In this part, we will also consider the following constraint:

- The solution should minimise the sum of travel distances of **all** robots.

General advice: debugging constraint models can be difficult because a conflicting set of constraints will cause the solver to return no solution without there being an obvious cause. For this reason, you are encouraged to build up the set of variables and constraints in your model **incrementally**, continually testing them on small instances.

In addition, it would be wise to proceed from part 1 to part 3 incrementally, and only attempting a part once you have a working solution of the previous part.

If you find your output hard to debug, you might consider investing time in building a simple way of visualizing the paths.

Input Format

The data is given to you in the following format (the particular values here are for illustration):

```
language ESSENCE' 1.0
letting nrows be 4
letting ncols be 4

letting nrobots be 1
letting robot_pos be [[1, 2]]

letting ntasks be 2
letting task_obj  be [2, 1]
letting task_dest be [[4,4],
                      [4,4]]

letting time_limit be 11

letting map be [[ 0, 0, 0, 0],
                [-1, 1, 2, 0],
                [-1, 3, 1, 0],
                [-1, 0, 0, 0]]
```

A description of the parameters follows:

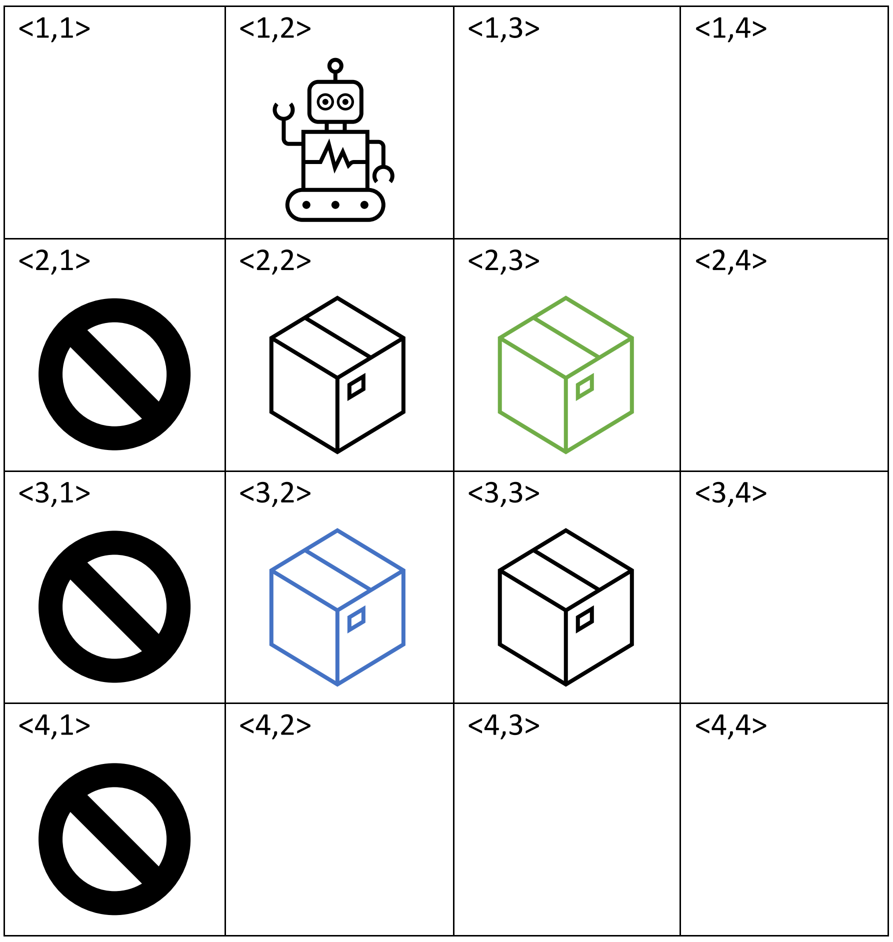
- `nrows` and `ncols` tells us the number of rows and columns the map has.

- `nrobots` tells us the number of robots the instance has.
- `robot_pos` is a two-dimensional matrix, with two columns and as many rows as robots. It lists the initial row and column of each robot. You can think about it as a list of tuples <X, Y>, where X is the row and Y the column.
- `ntasks` tells us the number of tasks the instance has.
- `task_obj` a one-dimensional matrix with one position per task. It describes the object that must be picked up for each task.
- `task_dest` a two-dimensional matrix (again, a list of positions <X, Y>) which has the coordinates to where the object of each task must be brought.
- `time_limit` shows the maximum number of steps (moves) any robot can do.
- `map` gives us a map of the warehouse. Remember that a 0 marks a traversable cell, a -1 an untraversable cell and the object piles are numbered from 1 onwards.

A note on indexing: Note that the indexing scheme goes from 1 to the number of columns/rows. For example, `ncols` being 4 means that columns go from 1 to 4.

Example

When considering the previously described instance, the map can be represented as follows.



A note on the map representation: The mapping between integers and item colours (1: black, 2: green, 3: blue) in the warehouse figure is purely for illustrative purposes.

If we consider the two tasks given to us:
```
letting ntasks be 2
letting task_obj  be [2, 1]
letting task_dest be [[4,4],[4,4]]
```

We need to bring a green and a black item to the <4,4> cell.
A possible valid path is to go <2,2>, picking up a black item. Then move <3,2>, <4,2>, <4,3> and finally <4,4>, delivering the black item. Then move all the way to <2,3> to pick up a green item and go back to <4,4> to deliver the final object. That would take us a total of 11 steps.

Note that although valid, we can do better. The minimal travelling distance can be achieved with 9 steps by doing: <1,3>,<2,3>,<2,4>,<3,4>,<4,4>,<3,4>,<3,3>,<3,4>,<4,4>

Instances

There are various instances given to you of increasing size. Generally, the bigger the instance the harder it is to solve, but it is not always the case. These instances are given so you can experimentally evaluate the limits of what your model can do. Note that it is **not** expected for your model to be able to solve every instance within a reasonable amount of time.

Extra performance

You can try to improve the performance of your model by adding symmetry breaking, implied or dominance constraints.

Report

- **Variables, domains, and Constraints**: Start by giving a general idea of how your model works. Describe how you chose the variables and their respective domains. In addition, describe each of the (sets of) constraints you have added to your model, and their purpose. To aid your explanation, feel free to copy-paste the constraints you used and explain them in detail in the report or use diagrams. If you have channelled two or more models, also explain them.
- **Additional Constraints (optional)**: If you have added any additional constraints such as symmetry breaking or implied constraints, explain them.
- **Basic Empirical Evaluation**: Describe your experimental setup, and record and analyse the output of the empirical evaluation described above. Evaluate the performance of your model on the provided instances supplied with this practical specification. In doing so use the default settings of Savile Row, such as:

```
./savilerow -run-solver model.eprime X.param
```

You should, *at least*, record in your report for each instance: the **SolverNodes**, **SolverSolveTime**, **Savile Row TotalTime**. If you have attempted part 3, include also the **optimal value** found. As discussed in Tutorial 1, these are available in the .info file after your model has been run, irrespective of whether the instance is solvable.

There are some ideas for a more extensive empirical evaluation below.

- **Additional Empirical Evaluation**: You might wish to evaluate how any of the following affects the performance:
  - Extensions to the model such as implied or symmetry breaking constraints.
  - A custom search heuristic (SR manual Section 2.5).
  - The difference in performance between various Savile Row backends (SR manual Section 1.2).

Key points

- The focus of the practical is on correctness. That is, your main worry should be that the solutions given by the model are valid. It is not expected that your model should solve all instances.
- The report should be especially concise and informative.
- If you wish you can expand your work in many ways, such as performance improvements via implied constraints, finding tighter bounds on the number of steps or by considering a good comparison of different backend solvers and preprocessing options that Savile Row offers.

Assessment Criteria

The practical will be marked following the standard mark descriptors as given in the Student Handbook (see link below). Further guidance as to what is expected follows:

- To achieve a mark of 7 or higher: A model for part 1, mostly complete, or complete with a few flaws. Good evaluation and report.
- To achieve a mark of 11 or higher: A model for part 2, mostly complete, or complete with a few flaws. Well evaluated and reported.
- To achieve a mark of 14 or higher: A correct model for part 3, at most with a few flaws. Well evaluated and reported.
- To achieve a mark of 17: A fully correct model for part 3. In addition, the report should be very well evaluated and reported, additionally exploring the problem more in depth via some of the suggested topics.
- To achieve a mark greater than 17: A fully correct model for part 3 which is performant, concisely and elegantly expressed. The report should be outstanding, with an in-depth and insightful evaluation, and further exploration of the subject.

Policies and Guidelines

Your attention is drawn to the following:

Marking
See the standard mark descriptors in the School Student Handbook:
http://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_Descriptors

Good academic practice
The University policy on Good Academic Practice applies:
https://www.st-andrews.ac.uk/students/rules/academicpractice/

Lateness

https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#lateness-penalties