Summary of the fourth paper

Moritz Feuerpfeil

May 2018

Abstract

This is a summary of the paper: ”Experimenting with robotic intra-logistics domains” written by Gebser, Obermeier, Otto, Schaub, Sabuncu, Van Nguyen and Tran Cao Son for the class ”Agent-technology” at the University of Potsdam.

1 Summary of the content

This paper introduces the *asprilo* framework that allows the user to model complex scenarios in the domain of robotic intra-logistics. *Asprilo* is an open-source project that aims to fill the gap of publicly available research material in this domain due to the privatization of similar projects by big companies, e.g. Amazon, Swisslog and GreyOrange. *Asprilo* allows not only to visualize ones solutions, but also to generate customized warehouse scenarios. A very important part of the framework is the benchmarking ability, which is quite relevant for research purposes. I will not go into syntactic detail because that will be understood and trained in the practical exercises.

1.1 The Components of Asprilo

The benchmark environment *asprilo* consists of four parts:

- a benchmark generator
- a solution checker
- a benchmark and solution visualizer
- different reference encodings

1.2 Problem domains

To put it in a very simple way, the domain can be boiled down to a warehouse and some robots that move shelves around with the goal of fulfilling a set of orders. The warehouse is modelled as a two-dimensional grid of equally sized squares. A square has one of the following functions:

- *highway*, where no robot or shelf shall rest
- *picking station*, where orders are processed

- *storage location*, where shelves can be put down and stored

To be more precise about what a robot can do at a time step, here is a list of available actions with their restrictions:

- *move* to an adjacent square (a robot carrying a shelf shall not move to a square containing a stored shelf and two robots shall not swap positions)
- *pick up* a shelf (if the robot does not carry one currently)
- *put down* a shelf (if the robot is at a storage square and currently carries one shelf)
- *deliver* a shelf to a picking station

When the authors talk about *orders*, they mean non-empty sets of *order lines*, which are basically requests for a single product in a set quantity.
In *asprilo* there are four different selected problem domains. In the general domain **A**, the focus lies on product quantities, which means that deliveries update the available products on a shelf and an order line may exceed the quantity of an product that is available on a single shelf.
The domain **B** ignores product quantities altogether but keeps the concept, that only one product line can be dealt with at a time.
The last domain that still deals with product delivery is the domain **C**, where one delivery action deals with all the order lines, requesting products available at the shelf, at once.
Last but not least, there is the domain **M**, which really only deals with multi-agent path finding (MAPF) scenarios. The goal condition in this domain is to find a plan that positions each robot under a certain shelf with only *move* actions.
Since there is quite a big gap between scenarios **A**, **B**, **C** and **M**, the authors came up with the scenarios $\mathbf{A^M}$, $\mathbf{B^M}$ and $\mathbf{C^M}$. These special domains stay pretty much the same as their "relatives" with one exception: there are only singleton orders and shelves hold unique products.

1.3 Experimental evaluation

The authors of this paper now used *asprilo* to benchmark examples of the domains $\mathbf{A^M}$, $\mathbf{B^M}$, $\mathbf{C^M}$ and **M**. This was done with three questions in mind:

- What is the impact of different representations of grid positions?
- What is the impact of increasingly complex domains?
- What is the impact of decoupling sources of combinatorics?

Since I did not really understand the differences between the grid position representations (and I am not sure how important this is for out future discussion), I will only present the outcome of the last two questions.
The authors pre-calculated the makespan with a time limit of 8 hours for the three types of grid sizes: small(11x6), medium(19x9) and large(46x15). They also differed in the quantity of storage squares, which were 16 for small, 60 for medium and 320 for large. To get a better overview, the number of robots were

incremented three times for each grid size, calculating 30 instances per increment.
For the second question one can say that the impact of the increasing complexity becomes less relevant when we go from complex to more complex. The biggest jump in runtime was noted when going from the M domain to the C domain. The jump was that big, that in domain C all 120 instances timed out for medium and large grid size. The differences between C and B were not that big and even smaller as the barely noticeable differences between B and A, which is to be expected because the increase in complexity from B to A is not that big (A only deals with singleton products).
One of the (at least for me) most interesting findings was the huge impact of prior task assignment. Even though the large grid size remained unsolvable, the medium grid size now became solvable in quite a reasonable time. That indicates better scalability with a run time decrease of at least an order of magnitude. The assignment even halves the number of variables and does not even lead an increase of the smallest makespan (except for C where it increases from 20 to 21), which is made possible by the optimized task assignment.

1.4 Conclusion

Asprilo is an ambitious project that offers benchmark scenarios for real-world applications with the goal to further improve Answer Set Programming and our understanding of it. The framework can be extended in many ways that allows for customizable domains within a certain range. Future work with *asprilo* can reveal more challenges to ASP-based approaches for solving logistic problems like the ones presented in terms of scalability and efficiency.

2 Comments

Unfortunately I missed reading this paper on time for our discussion, but I could follow the presentation nevertheless. When reading the paper I noticed that the presentation covered nearly all of it already, so I did not have a hard time understanding most of it. The one thing that I did not yet understand is the difference between the various "versions" of *clingo*. I am very much looking forward to experimenting with this framework!