Answer Set Solving in Practice

Prof. Torsten Schaub, Javier Romero

University of Potsdam — Winter Semester 2014/2015

Project 2 (Assembly Line)

Problem Description. The task of this project is to implement a workflow in a factory that assembles products in multiple steps. Each step consists of taking one or multiple parts and assembling or converting them into different parts. The goal is to complete all steps within a certain time frame with as few workers as possible.

The whole workflow consists of tasks (steps in the workflow) and parts that are created and consumed during a task. The factory has a set of workplaces that each have room for one worker and are connected by conveyor belts to move parts from one worker to the next. To simplify handling different kinds of assembly lines in one factory, not all of the workplaces are connected directly. Instead, conveyors are connected to sorters which assure that the parts will reach the corresponding workplace as long as there is a connection consisting of conveyors and sorters between the workplace producing a part and the workplace using it. Parts that are produced and consumed at the same workplace do not have to be transported. We assume that each task can be executed immediately at the beginning of the cycle without having to wait for required parts to be created. Conveyor belts only work in one direction. If workplaces or sorters are connected in both directions, then this is explicitly stated.

Implementing an assembly line requires you to assign each task to a workplace. Each workplace can handle multiple tasks as long as the time required to finish them does not exceed the length of the cycle. However, tasks cannot be split between multiple workplaces. A task can only be executed at a workplace if for every required part there is a connection to the workplace producing it. You're required to find an optimal solution for each workflow which means that there is no correct mapping from tasks to workplaces that uses less workplaces than your solution.

As an example we will consider the following setup:

- two tasks, $T1$ requiring 3 minutes and $T2$ requiring 2 minutes to complete;
- the time for one cycle being 4 minutes;
- one part A which is produced by $T1$ and consumed by $T2$;
- a factory with three workplaces, $W1$, $W2$, and $W3$;
- one sorter $S1$; and
- two conveyor belts, $C1$ moving parts from $W1$ to $S1$, and $C2$ from $S1$ to $W2$

Because both tasks combined take more than four minutes it is clear that at least two workers and thus workplaces have to be used. Since both tasks either produce or consume part A and cannot be executed at the same workplace, we cannot use workplace $W3$ because it has no connection. Which task is executed at which workplace is determined by the direction of the connection between $W1$ and $W2$. It is only possible to move parts from $W1$ to $W2$ which means that the only possible mapping from tasks to workplaces is $T1 \rightarrow W1$, $T2 \rightarrow W2$. Since there is only one solution this is also the optimal solution.

Representation in ASP. The working time in a cycle; the tasks and the parts involved; the workplaces, sorters, and the conveyor belts connecting them are represented by the following facts:

```
time(T).          % Working time per cycle
task(ID,T).       % Task and time required to complete it
produce(T,P).     % Part created in a task
consume(T,P).     % Part used in a task
workplace(W).     % Workplace
sorter(S).        % Sorter
conveyor(X1,X2).  % Conveyor belt connecting workplaces or sorters
```

For instance, the example is represented by the following facts:

```
time(4). task(1,3). task(2,2).
produce(1,a). consume(2,a).
workplace(1). workplace(2). workplace(3). sorter(s1).
conveyor(1,s1). conveyor(s1,2).
```

A solution – i.e. the mapping of tasks to workplaces – is represented as follows:

```
map_task(T1, W1) ... map_task(TN, WN)
```

Note that not all instances have solutions. The solution to the example is represented by the following atoms:

```
map_task(1,1) map_task(2,2)
```

Framework. In the `assembly.tar.gz` archive you will find the `assembly.lp` file and eight example instances. The file `assembly.lp` has to be completed with your ASP encoding. When you submit your encoding, the following lines have to be included in the `assembly.lp` file:[1]

```
#hide.
#show map_task/2.
```

The file containing your encoding has to be called `assembly.lp`. Apart from `map_task/2` no additional predicates must be included in the output!

Formalities. You can work on the solution alone or in groups of two people. Different groups have to submit different solutions, in case of plagiarism all groups involved will fail the project. Please submit your encoding by **April 5th, 2014** via YETI (All group members have to create a YETI account!). Be sure to submit your encoding in a file named `assembly.lp` containing only lowercase letters.

We will test your encoding with the eight provided instances as well as additional instances. Your encoding has to find the correct solutions for all instances. This will be tested automatically by YETI after you uploaded the encoding (with a slight delay). If your solution is not correct then YETI will display an error message. Please correct any errors that occur on your own or contact us if you get stuck.

[1]These instructions hide all atoms that are not part of the solution representation.

Tips:

- Make sure that your encoding generates correct solutions before integrating optimization to find optimal solutions.
- If you are stuck you can contact us. We will do out best to answer all your questions. You can send us questions and remarks via Moodle (forum/wiki) or send them via mail to `asp@lists.cs.uni-potsdam.de`.
- Start as soon as possible to avoid running out of time. (However, if you still realize that you have problems making it before the deadline, please contact us instead of copying another solution).