An Interactive Modelling Environment for Designing Warehouse Layouts Based on ASP

Andre Thevapalan, Gabriele Kern-Isberner, Marco Wilhelm
Department of Computer Science, TU Dortmund University, 44227 Dortmund, Germany
andre.thevapalan@tu-dortmund.de, gabriele.kern-isberner@cs.tu-dortmund.de, marco.wilhelm@tu-dortmund.de

Pascal Kaiser, Moritz Roidl
Chair of Material Handling and Warehousing, TU Dortmund University, 44227 Dortmund, Germany
pascal3.kaiser@tu-dortmund.de, moritz.roidl@tu-dortmund.de

Abstract

In this paper, we apply answer set programming (ASP) to the task of planning warehouse layouts within the logistical domain. Warehouse layout planners have to take into account a vast number of aspects to come up with feasible layouts, especially the placement of logistical elements under specific conditions like the accessibility of warehouse elements. Also optimization criteria, e. g., minimizing the pathways, have to be considered. Out of a generally large number of feasible layouts, the layout planner has to decide which layout fits best. As this can be quite difficult and time-consuming, we propose an interactive modelling environment which creates all feasible layouts for a given planning problem by exploiting a logistical knowledge base as well as specific case data provided by the user, who can interactively review all optimal layouts and look for the most suited one. It allows for taking soft constraints and vague expert knowledge into account while leaving room for proposing novel ideas for layouts.

Introduction

When planning layouts of warehouses for order picking, logistics experts are tasked with positioning a given set of elements like storage equipment of different sizes, a base (the start and endpoint of each order) and conveyor belts inside a warehouse while considering various restrictions (e. g., the accessibility of all structure elements) and target values (e. g., optimal material flow, minimal pathways). It is characterized by many trade-offs between conflicting objectives, i.e., it is a convoluted task, often resulting in a large number of feasible layouts. A crucial goal of the layout generation task is to find an optimal or at least "good" feasible layout of the storage area (Grosse, Glock, and Neumann 2015; de Koster, Le-Duc, and Roodbergen 2007), thus reducing the operational costs (Arnold and Furmans 2007; Kovács 2017).

Answer set programming (ASP) (Baral 2003; Gelfond and Lifschitz 1991) constitutes a declarative programming paradigm which allows the description of real-world problems by *extended logic programs* using *rules* r of the form $h \,\text{:-}\, b_1, \ldots, b_m, \mathit{not}\, b_{m+1}, \ldots, \mathit{not}\, b_n$. where h and all b_i's are *literals*. Rules with $n = 0$ are *facts*, rules with an empty head are *constraints*. Intuitively, an ASP rule states if $b_1, \ldots, b_m$ are true and $b_{m+1}, \ldots, b_n$ are not known to be true, then h has to be true. A solution of $\mathcal{P}$ is called an *answer set* which is a minimal set of literals that obeys all rules in $\mathcal{P}$ simultaneously. We use the ASP system *clingo* (Gebser et al. 2019) to integrate imperative programming.

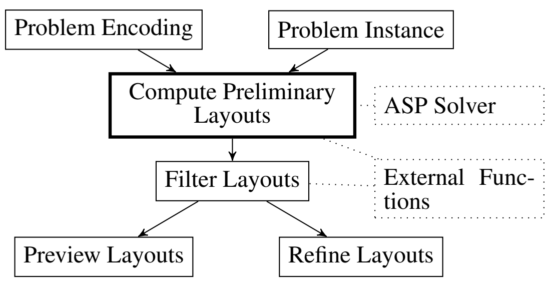

Figure 1: Framework overview

```
r1 elem_type(rid,"rname",rx,ry).
r2 reach(X,Y,TID):-reach(X+1,Y,TID)
     coords(X,Y), not occupied(X,Y).
r3 elem_reach(ID,T,h,X,Y)
     :-elem(ID,T,_,h,X,Y),reach(X,Y+1,T).
r4 :- elem(ID1,_,_,_,X,Y),elem(ID2,_,_,_,X,Y),
     ID1!=ID2.
r5 elem_quantity(rid,rq).
r6 blocked(bx,by).
```

Listing 1: Program (excerpt)

Due to its declarative paradigm, ASP is suitable for modelling complex problems such as the designing of warehouse layouts. We thus present an ASP-based interactive modelling framework that supports knowledge experts in finding all feasible warehouse layouts that meet all technical requirements and optimization objectives. We focus on an interactive approach where the logistician can provide various input data to define the parameters of the individual use case. Although the generated layouts possess the mandatory characteristics of warehouse layouts, they are not primarily based on traditional layouts, as these can impose quite significant restrictions. Instead, the modelling allows for taking soft constraints and vague expert knowledge into account and, thereby, leaves room for proposing novel ideas for layouts.

Size (in m)	Amount	Number of Layouts w/o orient.	w/ orient.
1x1	47	474	653,440
1x3	13	163	169
1x5	6	22	32
1x6	5	826	1,340

Table 1: Case Study Details

Interactive Modelling Environment

The components of the interactive modelling environment are illustrated in Figure 1. The encoding of structural warehouse elements, the definition of general dependencies, and constraints regarding their positioning is called the *problem encoding*. For instance, rule r_1 in Listing 1 encodes a rack type called $rname$ with reference id rid that occupies an area of rx meters by ry meters. To ensure that every rack is reachable, the program includes rules like r_2 and r_3 by which it obtains all reachable cells and defines that a rack is accessible if the cell above is reachable. To omit layouts that violate crucial conditions, constraints are used, e. g., r_4 states that a layout cannot contain two different racks on the same position. To encode the input parameters, the user adds the corresponding facts (*problem instance*) to the program, e. g., with r_5, every layout has to include rq racks of type rid and via r_6, no element shall is placed on position (bx, by). Using the encoding and the problem instance provided by the expert, the solver generates the answer sets that represent all feasible layouts. Further algorithms then filter out the optimal ones, e. g., w.r.t. the total distances. The layouts are visualized as 2D graphics and can be refined by the user.

Case Study

Assume that a warehouse planner has to position racks inside an existing warehouse for order-picking tasks. The warehouse has a basic layout of 20x20m and already contains a base and a conveyor belt that connects the base with a border of the warehouse. Table 1 shows the different types of racks, their sizes and the amount of each type that has to be positioned in a layout. Since, from a logistical viewpoint, it is not advantageous to mix different kinds of racks, each type of rack has to be positioned in a separate quadrant of the warehouse. Besides the base and the belt, there are areas where racks cannot be placed. Thus, positioning elements using traditional layouts as templates is not a viable option (Roodbergen and Vis 2006; Sancakli et al. 2022). The *optimization* directive is the minimization of the rack-to-rack and rack-to-base distances.

To compute all distinct layouts, each rack is either oriented horizontally or vertically. The generation of all layouts takes about 51 seconds. To get an optimal solution w.r.t. the total travel distance, the travel distances between every two racks and between the racks and the base are computed. For this, each rack must be cardinally oriented. We reach the total execution time of ~30 hours, while the calculation of the travel distances takes 50-200 milliseconds per layout. The amount of layouts for each subarea is shown in Table 1. We obtain one optimal layout w.r.t. total travel distance (Figure 2). It demonstrates that via ASP, arrangements are computed that resemble traditional layouts with parallel rows of racks (top half) as well as non-traditional layouts that fit the specific geometry of the warehouse (bottom half).

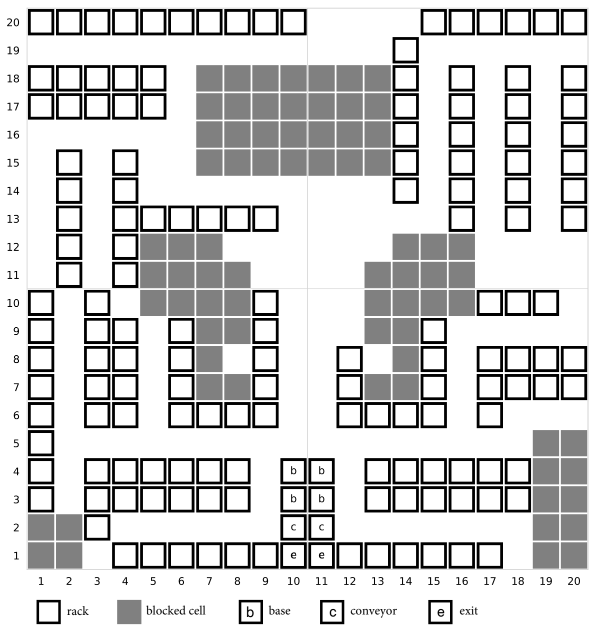

Figure 2: Optimal Layout

Conclusion and Future Work

We proposed an interactive modelling environment based on answer set programming (ASP) that aims at assisting logistical experts during the process of designing warehouse layouts. ASP poses a powerful tool to enhance the layout planning workflow by presenting an extended logic program, which generates feasible layouts when structural elements have to be positioned in a constrained warehouse meeting predefined optimization directives. The feasible layouts are visualized to the user, who can then refine specifications manually. ASP allows for effortlessly modifying the set of constraints as the order of rules is not relevant. Besides the cognitively adequate modelling of logistical tasks in general, it can dramatically reduce the time it takes to design warehouse layouts. In future work, we want to improve the calculation of travel distances, and we additionally plan to embed this approach into a hybrid inference framework that also takes preferences into account.

Acknowledgments. This work was supported by the Research Grant KE 1413/14-1 of the German Research Foundation (DFG) awarded to Gabriele Kern-Isberner.

References

Arnold, D., and Furmans, K. 2007. *Materialfluss in Logistiksystemen: mit 19 Tabellen*. Springer.

Baral, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press.

de Koster, R.; Le-Duc, T.; and Roodbergen, K. J. 2007. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2019. Multi-shot ASP solving with clingo. *Theory Pract. Log. Program.* 19(1):27–82.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Gener. Comput.*

Grosse, E. H.; Glock, C. H.; and Neumann, W. P. 2015. Human Factors in Order Picking System Design: A Content Analysis.

Kovács, G. 2017. Warehouse Design – Determination of the Optimal Storage Structure. *Acta Technica Corviniensis-Bulletin of Engineering*.

Roodbergen, K. J., and Vis, I. F. A. 2006. A model for warehouse layout. *IIE Transactions*.

Sancakli, E.; Dumlupinar, I.; Akcin, A. O.; Cinar, E.; Geylani, I.; and Düzgit, Z. 2022. Design of a Routing Algorithm for Efficient Order Picking in a Non-traditional Rectangular Warehouse Layout. In *Digitizing Production Systems*.