Shift Design with Answer Set Programming

M. Abseher[1], M. Gebser[2,3], N. Musliu[1], T. Schaub[3,4⋆], and S. Woltran[1]

[1] TU Wien, Austria
[2] Aalto University, HIIT, Finland
[3] University of Potsdam, Germany
[4] INRIA Rennes, France

Abstract. Answer Set Programming (ASP) is a powerful declarative programming paradigm that has been successfully applied to many different domains. Recently, ASP has also proved successful for hard optimization problems like course timetabling. In this paper, we approach another important task, namely, the shift design problem, aiming at an alignment of a minimum number of shifts in order to meet required numbers of employees (which typically vary for different time periods) in such a way that over- and understaffing is minimized. We provide an ASP encoding of the shift design problem, which, to the best of our knowledge, has not been addressed by ASP yet.

1 Introduction

Answer Set Programming (ASP) [4] is a declarative formalism for solving hard computational problems. Thanks to the power of modern ASP technology [8], ASP was successfully used in various application areas, including product configuration [13], decision support for space shuttle flight controllers [11], team building and scheduling [12], and bio-informatics [9]. Recently, ASP also proved successful for optimization problems that had not been amenable to complete methods before, for instance in the domain of timetabling [2].

In this paper, we investigate the application of ASP to another important domain, namely, workforce scheduling [3]. Finding appropriate staff schedules is of great relevance because work schedules influence health, social life, and motivation of employees at work. Furthermore, organizations in the commercial and public sector must meet their workforce requirements and ensure the quality of their services and operations. Such problems appear especially in situations where the required number of employees fluctuates throughout time periods, while operations dealing with critical tasks are performed around the clock. Examples include air traffic control, personnel working in emergency services, call centers, etc. In fact, the general employee scheduling problem includes several subtasks. Usually, in the first stage, the temporal requirements are determined based on tasks that need to be performed. Further, the total number of employees is determined and the shifts are designed. In the last phase, the shifts and/or days off are assigned to the employees. For shift design [10], employee requirements for a period of time, constraints about the possible start and length of shifts, and limits for the average number of duties per week are considered. The aim is to generate

⋆ Affiliated with Simon Fraser University, Canada, and IIIS Griffith University, Australia.

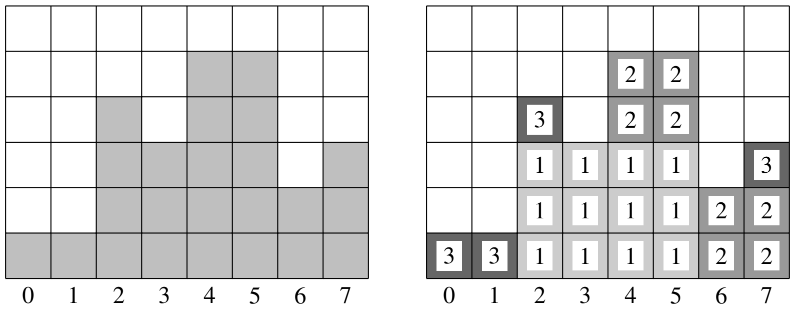

Fig. 1. Work demands over a day (left) and the unique optimal schedule (right) with shifts starting at slot 2, 4, or 7, respectively, indicated by different boxes, while other kinds of shifts are unused

solutions consisting of shifts (and the number of employees per shift) that fulfill all hard constraints, while minimizing the number of distinct shifts as well as over- and understaffing. This problem has been addressed by local search techniques, including a min-cost max-flow approach [10] and a hybrid method combining network flow with local search [6]. These techniques have been used to successfully solve randomly generated examples and problems arising in real-world applications. A detailed overview of previous work on shift design is given in [7].

Although the aforementioned state-of-the-art approaches for the shift design problem are able to provide optimal solutions in many cases, obtaining optimal solutions for large problems is still a challenging task. Indeed, for several instances the best solutions are still unknown. Therefore, the application of exact techniques like ASP is an important research target. More generally, it is interesting to see how far an elaboration-tolerant, general-purpose approach such as ASP can compete with dedicated methods when tackling industrial problems. Our ASP solution is based on the first author's master thesis [1] and relies on sophisticated modeling and solving techniques, whose application provides best practice examples for addressing similarly demanding use-cases. In particular, we demonstrate how order encoding techniques [5] can be used in ASP for modeling complex interval constraints. Experimental results and further details are provided in the full version[5] of this paper.

2 The Shift Design Problem

To begin with, let us introduce the shift design problem. Our problem formulation follows the one in [10]. As input, we are given the following:

- consecutive *time slots* sharing the same length. Each time slot is associated with a number of employees that should be present during the slot.
- *shift types* with associated parameters *min-start* and *max-start*, representing the earliest and latest start, and *min-length* and *max-length*, representing the minimum and maximum length of a shift.

[5] `www.dbai.tuwien.ac.at/proj/Rota/ShiftDesignASP.pdf`

$$\left\{\begin{array}{l}
time(0,0), time(1,1), \ldots, time(7,7), next(0,1), next(1,2), \ldots, next(7,0),\\
work(0,1), work(1,1), work(2,4), work(3,3), work(4,5), work(5,5),\\
work(6,2), work(7,3), exceed(1), shorten(1), opt(shortage,3,1),\\
opt(excess,2,1), opt(select,1,1), range(2,2,1), \ldots, range(2,2,4),\\
range(2,3,1), \ldots, range(2,3,5), range(2,4,1), \ldots, range(2,4,5),\\
range(4,2,1), \ldots, range(4,2,5), range(4,3,1), \ldots, range(4,3,5),\\
range(4,4,1), \ldots, range(4,4,5), range(5,2,1), \ldots, range(5,2,5),\\
range(5,3,1), \ldots, range(5,3,5), range(5,4,1), \ldots, range(5,4,5),\\
range(6,2,1), \ldots, range(6,2,3), range(6,3,1), \ldots, range(6,3,3),\\
range(6,4,1), \ldots, range(6,4,3), range(7,2,1), \ldots, range(7,2,3),\\
range(7,3,1), \ldots, range(7,3,3), range(7,4,1), \ldots, range(7,4,4)
\end{array}\right\}$$

Fig. 2. ASP facts specifying an instance of the shift design problem

The aim is to generate a collection of k shifts $s_1, \ldots, s_k$. Each shift s_i is completely determined by its *start* and *length*, which must belong to some shift type. Additionally, each shift s_i is associated with parameters indicating the number of employees assigned to s_i during each day of the planning period. Note that we consider cyclic planning periods where the successor of the last time slot is equal to the first time slot. An example of employee requirements and a corresponding (optimal) schedule are shown in Figure 1.

In analogy to [6], we investigate the optimization of the following criteria: sum of *shortages* of workers in each time slot during the planning period, sum of *excesses* of workers in each time slot during the planning period, and the *number of shifts*.[6] Traditionally, the objective function is a weighted sum of the three components (although this kind of aggregation is not mandatory with ASP).

3 Shift Design in ASP

An instance like the one shown in Figure 1 is specified by facts as in Figure 2. Facts of the form $time(S,T)$ associate each slot S with a day time T. Our instance includes one day, divided into eight slots denoted by the times $0, \ldots, 7$. Instances of $next(S', S)$ provide predecessor or successor slots, respectively, where S is usually $S'+1$, except for the last slot whose successor is 0. (When another day is added, the slots $8, \ldots, 15$ would also be mapped to day times $0, \ldots, 7$, $next(7,0)$ would be replaced by $next(7,8)$, and $next(15,0)$ would connect the new last slot to 0 instead.) For each slot S, a fact $work(S,N)$ gives the number N of desired employees, and $exceed(E)$ as well as $shorten(F)$ may limit the amount of employees at duty to at most $E+N$ or at least $N-F$, respectively. For instance, we obtain the upper bound 4 and the lower bound 2 for employees engaged in slot 7. Facts of the form $range(S,L,1), \ldots, range(S,L,M)$ provide potential amounts of shifts of length L that can start from slot S, where M is the maximum number of desired employees over all slots within the horizon of the shift. For shifts starting from slot 7, those of length 2 or 3 stretch to slot 0 or 1, respectively, and the corresponding maximum number of desired employees is 3 in slot 7 itself; unlike that, shifts of length 4 also include slot 2 in which 4 employees shall be at duty.

[6] In [10], additionally, the average number of duties per week is considered.

$$
\begin{array}{rcll}
\{\mathit{run}(S,L,I)\} & \leftarrow & \mathit{range}(S,L,I) & (1)\\
\mathit{run}(S,L,I) & \leftarrow & \mathit{run}(S',L{+}1,I), \mathit{next}(S',S), 0 < L & (2)\\
\mathit{run}(S,L,I) & \leftarrow & \mathit{run}(S,L{+}1,I), 0 < L & (3)\\
\mathit{run}(S,L,I{+}J) & \leftarrow & \mathit{run}(S,L{+}1,I), \mathit{shift}(S,L,J) & (4)\\
 & \leftarrow & \mathit{run}(S,L,I{+}1), 0 < I, {\sim}\mathit{run}(S,L,I) & (5)\\
 & \leftarrow & \mathit{work}(S,N), \mathit{exceed}(E), \mathit{run}(S,1,N{+}E{+}1) & (6)\\
 & \leftarrow & \mathit{work}(S,N), \mathit{shorten}(F), F < N, {\sim}\mathit{run}(S,1,N{-}F) & (7)\\
\mathit{length}(S,L,I,1) & \leftarrow & \mathit{range}(S,L,I), \mathit{run}(S,L,I), {\sim}\mathit{run}(S,L{+}1,I) & (8)\\
\mathit{length}(S,L,I,J) & \leftarrow & \mathit{length}(S,L,I{+}1,J{-}1), 0 < I, {\sim}\mathit{run}(S,L{+}1,I) & (9)\\
\mathit{shift}(S,L,J) & \leftarrow & \mathit{length}(S,L,I,J) & (10)\\
\mathit{shift}(S,L,J) & \leftarrow & \mathit{shift}(S',L{+}1,J), \mathit{next}(S',S), 0 < L & (11)\\
\mathit{start}(S,L,J) & \leftarrow & \mathit{range}(S,L,J), \mathit{next}(S',S), \mathit{shift}(S,L,J), {\sim}\mathit{shift}(S',L{+}1,J) & (12)\\
W@P,S,I,\mathit{shortage} & \leftsquigarrow & \mathit{opt}(\mathit{shortage},P,W), \mathit{work}(S,N), I \in [1,N), {\sim}\mathit{run}(S,1,I) & (13)\\
W@P,S,I,\mathit{excess} & \leftsquigarrow & \mathit{opt}(\mathit{excess},P,W), \mathit{work}(S,N), \mathit{run}(S,1,I), N < I & (14)\\
W@P,T,L,\mathit{select} & \leftsquigarrow & \mathit{opt}(\mathit{select},P,W), \mathit{start}(S,L,J), \mathit{time}(S,T) & (15)
\end{array}
$$

Fig. 3. ASP encoding of the shift design problem

Moreover, facts $\mathit{opt}(\mathit{shortage},P,W)$, $\mathit{opt}(\mathit{excess},P,W)$, and $\mathit{opt}(\mathit{select},P,W)$ specify optimization criteria in terms of a priority P and a penalty weight W incurred in case of violations. The priorities in Figure 2 tell us that the desired number of employees shall be present in the first place, then the amount of additional employees ought to be minimal, and third the number of utilized shifts in terms of day time and length should be as small as feasible. Given that the criteria are already distinguished by priority, the penalty weight of a violation of either kind is 1, thus counting particular violations to assess schedules.

Our ASP encoding of the shift design problem is shown in Figure 3. For a slot S, the intuitive reading of the predicate $\mathit{run}(S,L,I)$ is that at least I shifts including S and $L{-}1$ or more successor slots are scheduled. This is further refined by $\mathit{length}(S,L,I,J)$, telling that $1,\dots,J$ of the scheduled shifts of exact length L may start from S, where $I{-}1$ shifts that include at least $L{-}1$ successor slots are scheduled in addition. The predicate $\mathit{shift}(S,L,J)$ expresses that at least J of the scheduled shifts stretch to S and exactly $L{-}1$ successor slots, and $\mathit{start}(S,L,J)$ indicates that the J-th instance of such a shift indeed starts from S. A schedule is thus characterized by the number of (true) atoms of the form $\mathit{start}(S,L,J)$, yielding the amount of shifts of length L starting from slot S. For example, the schedule displayed in Figure 1 is described by a stable model containing $\mathit{start}(2,4,1)$, $\mathit{start}(2,4,2)$, $\mathit{start}(2,4,3)$, $\mathit{start}(4,4,1)$, $\mathit{start}(4,4,2)$, and $\mathit{start}(7,4,1)$. When further scheduling a shift of length 2 to start from slot 6, it would be indicated by $\mathit{start}(6,2,3)$, as it adds to the two shifts from slot 4 stretching to slot 6 and 7 as well. However, the displayed schedule is the unique optimal solution, given that it matches the desired employees and uses a minimum number of shifts, viz. shifts of length 4 starting from slot 2, 4, or 7, respectively.

In more detail, the potential start of an instance I of a shift of length L from slot S is reflected by the choice rule (1) in Figure 3. Rule (2) propagates the start of a shift to its $L-1$ successor slots, where the residual length is decreased down to 1 in the last slot of the shift. For shifts with longer residual length L, rule (3) closes the interval between 1 and L, thus overturning any choice rules for potential starts of shifts of shorter length. Moreover, this allows for pushing the J-th instance of a shift stretching to slot S to the position $I+J$ when I instances of shifts longer than the residual length L are scheduled, as expressed by rule (4). The integrity constraint (5) asserts that the positions associated with scheduled shifts must be ordered by residual length. This condition eliminates guesses on instances I of starting shifts, and it also provides a shortcut making interconnections between positions of scheduled shifts explicit, which turned out as effective to improve search performance. The additional integrity constraints (6) and (7) are applicable whenever the deviation from numbers of desired employees is bounded above or below, respectively. Note that it is sufficient to inspect atoms of the form $run(S, 1, I)$ for appropriate positions I, given that residual lengths are propagated via rule (3).

In order to derive the amount of scheduled shifts of exact residual length L, rule (8) marks positions I, where instances may start, with 1 when the length L matches. Instances associated with smaller positions then count on by means of rule (9) unless their positions are occupied by shifts with longer residual length. By projecting the positions out, rule (10) yields that $1, \dots, J$ shifts of length L may start from slot S. In addition, longer shifts whose residual length decreases to L in S are propagated via rule (11). Finally, rule (12) compares instances that may start to propagated shifts and indicates the ones that indeed start from S. As a consequence, a stable model represents a schedule in terms of sequences of the form $start(S, L, m), \dots, start(S, L, n)$, expressing that $n+1-m$ instances of a shift of length L start from slot S. It remains to assess the quality of a schedule, which is accomplished by means of the weak constraints (13), (14), and (15) for the three optimization criteria at hand. The penalty for deviating from a number of desired employees is characterized in terms of the priority P and weight W given in facts, a position I pointing to under- or overstaffing in a slot S, and the corresponding keyword *shortage* or *excess*, respectively, for avoiding clashes with penalties due to the utilization of shifts. The latter include the keyword *select* and map the slot S of a starting shift of length L to its day time T, so that the penalty $W@P$ is incurred at most once for a shift with particular parameters, no matter how many instances are actually utilized.

A prevalent feature of our ASP encoding in Figure 3 is the use of closed intervals (starting from 1) to represent quantitative values such as residual lengths or instances of shifts. The basic idea is similar to the so-called *order encoding* [5], which has been successfully applied to solve constraint satisfaction problems by means of SAT [14]. In our ASP encoding, rule (4), (8), and (9) take particular advantage of the order encoding approach by referring to one value, viz. $L+1$, for testing whether any shift with longer residual length than L is scheduled. Likewise, the integrity constraints (6) and (7) as well as the weak constraints (13) and (14) focus on value 1, standing for any residual length, to determine the amount of employees at duty. That is, the order encoding approach enables a compact formulation of existence tests and general conditions, which then propagate to all target values greater or smaller than a certain threshold.

4 Discussion

In this work, we presented a novel approach to tackle the shift design problem by using ASP. Finding good solutions for shift design problems is of great importance in different organizations. However, such problems are very challenging due to the huge search space and conflicting constraints. Our work contributes to better understanding the strengths of ASP technology in this domain and extends the state of the art for the shift design problem by providing new optimal solutions for benchmark instances first presented in [6]. Below we summarize the main observations of our experiments, detailed in the full paper[5], regarding the application of ASP to the shift design problem:

- ASP shows very good results for shift design problems that have solutions without over- and understaffing. Our proposed ASP approach could provide optimal solutions for almost all such benchmark instances (DataSet1 and DataSet2 in [6]).
- The first results for problems that do not have solutions without over- or understaffing are promising. Although our current approach could not reproduce best known solutions for several problems, we were able to provide global optima for four hard instances (from DataSet3 in [6]), not previously solved to the optimum.
- Our experimental evaluation indicates that our approach could also be used in combination with other search techniques. For example, solutions computed by metaheuristic methods or min-cost max-flow techniques could be further improved by ASP.
- In general, the computational results show that ASP has the potential to provide good solutions in this domain. Therefore, our results open up the area of workforce scheduling, which is indeed challenging for state-of-the-art ASP solvers. This is most probably caused by the nature of the shift design problem, as there are few hard constraints involved that could help to restrict the search space.

Concerning related work, we mention an ASP implementation of a problem from the domain of workforce management [12], where the focus is on the allocation of employees of different qualifications to tasks requiring different skills. The resulting system is tailored to the specific needs of the seaport of Gioia Tauro. From the conceptual point of view, the main difference to our work is that the encoded problem of [12] is a classical allocation problem with optimization towards work balance, while the problem we tackle here aims at an optimal alignment of shifts.

As future work, we plan to tackle the problem of optimization in shift design by combining ASP with domain-specific heuristics in order to better guide the search, but also exploiting off-the-shelf heuristics is a promising target for further investigation. We are confident that ASP combined with heuristics is a powerful tool for tackling problems in the area of workforce scheduling. This fact is already underlined by significantly improved results obtained for branch-and-bound based optimization when activating particular off-the-shelf heuristics. By using customized heuristics, tailored to the specific problem at hand, the chance for further improvements is thus high.

Acknowledgments. This work was funded by AoF (251170), DFG (550/9), and FWF (P25607-N23, P24814-N23, Y698-N23).

References

1. Abseher, M.: Solving shift design problems with answer set programming. Master's thesis, Technische Universität Wien (2013)
2. Banbara, M., Soh, T., Tamura, N., Inoue, K., Schaub, T.: Answer set programming as a modeling language for course timetabling. Theory and Practice of Logic Programming 13(4-5), 783–798 (2013)
3. den Bergh, J., Beliën, J., Bruecker, P., Demeulemeester, E., Boeck, L.: Personnel scheduling: A literature review. European Journal of Operational Research 226(3), 367–385 (2013)
4. Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Communications of the ACM 54(12), 92–103 (2011)
5. Crawford, J., Baker, A.: Experimental results on the application of satisfiability algorithms to scheduling problems. In: Proceedings of AAAI'94, pp. 1092–1097. AAAI Press (1994)
6. Di Gaspero, L., Gärtner, J., Kortsarz, G., Musliu, N., Schaerf, A., Slany, W.: The minimum shift design problem. Annals of Operations Research 155, 79–105 (2007)
7. Di Gaspero, L., Gärtner, J., Musliu, N., Schaerf, A., Schafhauser, W., Slany, W.: Automated shift design and break scheduling. In: Automated Scheduling and Planning, pp. 109–127. Springer (2013)
8. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer Set Solving in Practice. Morgan & Claypool Publishers (2012)
9. Guziolowski, C., Videla, S., Eduati, F., Thiele, S., Cokelaer, T., Siegel, A., Saez-Rodriguez, J.: Exhaustively characterizing feasible logic models of a signaling network using answer set programming. Bioinformatics 29(18), 2320–2326 (2014)
10. Musliu, N., Schaerf, A., Slany, W.: Local search for shift design. European Journal of Operational Research 153(1), 51–64 (2004)
11. Nogueira, M., Balduccini, M., Gelfond, M., Watson, R., Barry, M.: An A-Prolog decision support system for the space shuttle. In: Proceedings of PADL'01, pp. 169–183. Springer (2001)
12. Ricca, F., Grasso, G., Alviano, M., Manna, M., Lio, V., Iiritano, S., Leone, N.: Team-building with answer set programming in the Gioia-Tauro seaport. Theory and Practice of Logic Programming 12(3), 361–381 (2012)
13. Soininen, T., Niemelä, I.: Developing a declarative rule language for applications in product configuration. In: Proceedings of PADL'99, pp. 305–319. Springer (1998)
14. Tamura, N., Taga, A., Kitagawa, S., Banbara, M.: Compiling finite linear CSP into SAT. Constraints 14(2), 254–272 (2009)