Casual Employee Scheduling with Constraint Programming and Metaheuristics

Nikolaus Frohner, Stephan Teuschl, and Günther R. Raidl

Institute of Logic and Computation, TU Wien, Austria
{nfrohner|raidl}@ac.tuwien.ac.at, e0608934@student.tuwien.ac.at

Abstract. We consider an employee scheduling problem where many casual employees have to be assigned to shifts defined by the requirement of different work locations. For a given planning horizon, locations specify these requirements by stating the number of employees needed at specific times. Employees place offers for shifts at locations they are willing to serve. The goal is to find an assignment of employees to the locations' shifts that satisfies certain hard constraints and minimizes an objective function defined as weighted sum of soft constraint violations. The soft constraints consider ideal numbers of employees assigned to shifts, distribution fairness, and preferences of the employees. The specific problem originates in a real-world application at an Austrian association. In this paper, we propose a Constraint Programming (CP) model which we implemented using MiniZinc and tested with different backend solvers. As the application of this exact approach is feasible only for small to medium sized instances, we further consider a hybrid CP/metaheuristic approach where we create an initial feasible solution using a CP solver and then further optimize by means of an ant colony optimization and a variable neighborhood descent. This allows us to create high-quality solutions which are finally tuned by a manual planner.

Keywords: Employee Scheduling · Constraint Programming · Ant Colony Optimization · Multi-Objective Optimization · Variable Neighborhood Descent

1 Introduction

We consider an employee scheduling problem that arises as a real-world problem in an Austrian association. It deals with assigning employees to shifts at work locations within a given planning horizon so that certain hard constraints are fulfilled and the violation of soft constraints regarding demand satisfaction of the locations, fairness, and preferences of the employees is minimized. The problem falls into the broad class of personnel scheduling [1] with strong ties to the nurse rostering problem [2] but has some distinguishing features. One is the substantial fluctuation of employees and the high variance of their availabilities over different planning horizons and within each; therefore employees are coined "casual". Another specialty is that employees specify individual maximum numbers of

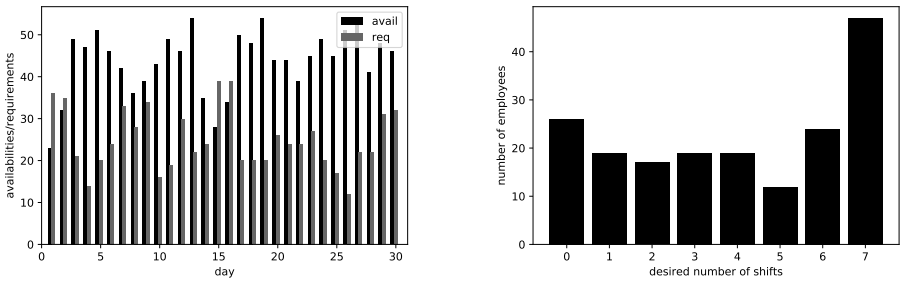

Fig. 1. (a) Availabilities of employees to serve a shift vs. requirements of locations over an exemplary month. (b) Distribution of maximum number of shifts employees offer to serve in the considered month.

shifts they desire to work. The variance of the ratio of actual shifts assigned divided by the desired shifts should be minimized to balance the fulfillment of the employees' desires. Likewise, the fulfillment ratio of the locations' requirements shall be balanced as well. Figure 1a shows the availabilities of employees to serve shifts on given days over a month compared with the requirements by the locations. This exemplary month starts with a weekend where a lack of employees is evident, whereas on other days there is substantial overcapacity. A hard constraint is that employees cannot be assigned every day they are available since they offer a desired number of shifts for a month which also acts as a hard upper limit. The corresponding distribution can be seen in Fig. 1b. Typically, in our application there is always a shortage of workers which makes it highly desirable to distribute this shortage evenly over the shifts.

In the following Section 2, we will formally define the optimization problem including its hard and soft constraints. This formulation gives rise to an exact approach by means of Constraint Programming (CP) which we will describe in Section 3. In Section 4, we introduce a hybrid algorithm that makes use of this CP-model, Ant Colony Optimization, and Variable Neighborhood Descent to be able to tackle large problem instances. In Section 5 we conduct a computational study for both approaches on artificially generated data and real data provided by an Austrian association, after which we conclude in Section 6.

2 Problem Formulation

Given locations L and a planning horizon D consisting of days $d \in D$, the tuples $(d, l) =: s \subset D \times L$ constitute shifts. Each shift s has a requirement $R_s \in \mathbb{Z}_+$ of employees that should ideally serve it. Each employee $w \in W$ chooses certain shifts $S_w \subset S$ which he could potentially serve and a desired number of shifts N_w, which also acts as upper limit of the shifts w will be assigned to. The locations are comprised of houses H that have shifts on a more regular basis and events E whose shifts are more sparsely distributed over the planning horizon

but possibly with higher requirement peaks. Furthermore, there are two special locations: *standby*, denoted by b, which is used in case someone becomes sick, and *floating*, denoted by f, for employees that are assigned dynamically to a house on the very day of the shift. Only employees whose numbers of shifts N_w is above a given threshold are eligible for standby and floating shifts. If an employee selects a shifts (d, l) where $l \in H$, then all the other houses are selected automatically for that day as well, so that there is enough flexibility for the planners. However, the employees provide a nonempty preference list of houses $H_w \subset H$. Events on the other hand can be selected separately.

The goal is to find an assignment of employees to shifts, which we denote by the sets of shifts $A_w \subset S_w$ each worker $w \in W$ is assigned to, that satisfies a number of hard constraints and minimizes violations of a number of soft constraints. The most relevant hard constraints are that employees have to be assigned at least once, at most in accordance to their desired/maximum number of shifts, and at each day at most once; each shift has a time duration and the total duration for each worker must stay within legal bounds. Furthermore, the standby and floating shifts must be fully covered, otherwise their purpose of being a backup would be defeated.

We are thus facing a multi-objective optimization problem where soft constraint violations are modeled as different objectives with different priorities. Since shifts provide service to paying customers, the fulfillment of the corresponding requirements is by far the most important objective. Given an assignment A, each shift has a relative shortage $u_s = 1 - |\{A_w \in A \mid s \cap A_w \neq \emptyset\}|/R_s$ that should be kept small and balanced over all shifts, which we implement by minimizing the mean squared error of the vector $\mathbf{u} = (u_s)_{s \in S}$ with respect to the desired optimum $\mathbf{u}^* = \mathbf{0}$. Next priority is to distribute the shifts over employees as fair as possible, taking their numbers of desired shifts into account. To achieve this, we minimize the variance of the sum of the assigned shift durations divided by the number of desired shifts multiplied by the maximum shift duration over the employees. The shift duration is denoted as Δ_s. The fractions of floating shift hours over the assigned hours should also be distributed evenly among all workers. Last but not least, we aim at keeping the ratios workers are assigned to non-preferred houses small and balanced over all workers, for which again minimizing the corresponding mean squared error is deemed suitable. Putting everything together yields the vector-valued objective function $\mathbf{f}\colon \mathcal{A} \to [0,1]^4$:

$$\mathbf{f}(A) := (g_u(A), g_f(A), g_{ff}(A), g_{np}(A))^T \tag{1}$$

$$g_u(A) := \frac{1}{|S|} \sum_{s \in S} \left(1 - \frac{|\{A_w | s \cap A_w \neq \emptyset\}|}{R_s}\right)^2 \tag{2}$$

$$g_f(A) := \mathrm{Var}\left[\frac{\sum_{s \in A_w} \Delta_s}{N_w \max_{s \in S} \Delta_s}\right], g_{ff}(A) := \mathrm{Var}\left[\frac{\sum_{s \in A_w \wedge l(s) = f} \Delta_s}{\sum_{s \in A_w} \Delta_s}\right] \tag{3}$$

$$g_{np}(A) := \frac{1}{|W|} \sum_{w \in W} \left(\frac{|\{s \in A_w | l(s) \in H \setminus H_w\}|}{|A_w|}\right)^2 \tag{4}$$

3 Exact Solution Approach

We model the problem as a constraint program using MiniZinc [5]. We consider different formulations. The first one is based on a set formulation and the second one on binary decision variables. In the set formulation, the main decision variable is a two-dimensional array of size $|L| \cdot |D|$ containing sets of integers representing an assignment $A_{(l,d)} \subset W,\ \forall (d,l) \in S$ of specific employees to shifts. Table 1 shows a simplified assignment example of three employees, being assigned to three houses with different requirements over three days where every employee is assigned the desired number of shifts.

Table 1. Small example of a assignment with three houses, three days and three employees.

w	N_w	H_w
0	3	$\{2\}$
1	2	$\{1,2,3\}$
2	1	$\{0\}$

$R_{(l,d)}$	0	1	2
0	1	1	1
1	0	1	1
2	2	1	0

$A_{(l,d)}$	0	1	2
0	$\{0\}$	$\{2\}$	$\{1\}$
1	$\{\}$	$\{1\}$	$\{0\}$
2	$\{\}$	$\{0\}$	$\{\}$

For the minimization of the soft constraint violations we choose the weighted sum approach, where the vector-valued objective function is condensed to a real-valued function:

$$f(A) := \lambda_u g_u(A) + \lambda_f \cdot g_f(A) + \lambda_{ff} \cdot g_{ff}(A) + \lambda_{np} \cdot g_{np}(A) \tag{5}$$

We designed the soft constraint violations to yield values between zero and one, therefore no special re-scaling is necessary. To put more weight on the unassigned shifts objective and keep the others equally weighted, we use the weights $\lambda_u = 10$ and $\lambda_f = \lambda_{ff} = \lambda_{np} = 1$. Since we make use of floating point variables, we solve the MiniZinc models by two different, float-capable solver backends, namely Gecode and JaCoP.

In the approach where we greedily extend a small initial solution as described in the next section, an alternative CP model is used to only satisfy the hard constraints. It is based on $|L| \cdot |D| \cdot |W|$ binary decision variables, stating whether an employee w is assigned to a shift (d,l). It does not consider the soft constraints and is thus only used for pure hard constraint satisfaction. This model can then also be solved by the solvers Chuffed and Gurobi that as of the time of writing neither support sets nor floats in combination with MiniZinc.

4 Hybrid Approach

First tests with MiniZinc using real-world instances indicate that our problem instances are too big to be solved to optimality within reasonable time. This gives rise to a hybrid approach, where we use CP to create an initial solution

that satisfies the hard constraints and which is then fed into a metaheuristic for further improvement. We propose two different combinations of CP and metaheuristics, where both our CP-models, the one with soft constraint optimization and the one with hard-constraints only, come into play:

1. The MiniZinc optimization model, which also considers the soft constraints, is solved until a first feasible solution is obtained in order to obtain a rather "complete" solution from CP, which is then passed to a variable neighborhood descent (VND) for possible further improvement.
2. The MiniZinc hard constraint satisfaction model is used to create a small feasible solution. This solution typically has a high objective value and can be substantially improved by assigning further employees. This is done by a successively applying an Ant Colony Optimization (ACO) [3, 4] with a min/max pheromone model [6]. Each ant starts from the initial solution and iteratively extends it according to the ACO's usual probabilistic principles in combination with certain greedy criteria. The so far best solution is used for pheromone update. After we hit a time limit, we take the best solution and try to improve it further by the variable neighborhood descent (VND), either up to local optimality or until an overall time limit is hit.

In the first variant, the CP solver is used in optimization mode to create a rather complete (many assignments), initial solution, whereas in the second variant the binary decision variable model is used in satisfaction mode to create a rather small (few assignments) solution. The latter is realized by using the value choice heuristic *indomain_min* which prefers to set decision variables to zero.

For the VND, we use neighborhood structures induced by the following operations, in the given order: *MoveEmployeeInDay*: For a given employee w and a day d, change the location of his assignment, *AssignEmployee*: Assign a shift with shortage to an employee, *ReassignShift*: Unassign the shift from an employee and assign it to a different employee, *ReassignEmployee*: Unassign an employee from a shift and assign the employee to a different shift, and *SwapShifts*: Swap shifts of two employees. All these neighborhoods are searched in a first improvement fashion.

Each operator has impact on a different set of soft constraints. For example, swapping the assignment of two employees keeps the distribution fairness among locations unchanged but may improve the preference satisfaction of the employees.

For the ACO we use a min/max pheromone update system as described in [6], which bounds the pheromone values to lie within the interval $[\tau^{\min}, \tau^{\max}]$. The pheromone matrix elements $\tau_{s,w}$ encode a bias for assigning shift s to employee w. Given an ant's current assignment A and $S_u(A) \ni s$ be the shifts with shortage and employees $W_s \ni w_s$ that are available for this shift and fulfill all the hard constraints, then we add the assignment $s \leftrightarrow w_s$ as extension to A resulting in A' with a probability depending on this bias and an attractiveness depending on the decrease in the objective value $\Delta f(A, A') = f(A) - f(A')$:

$$p_{s,w_s} \sim \tau_{s,w_s}^{\alpha} \cdot (1 + \Delta f(A, A'))^{\beta}$$

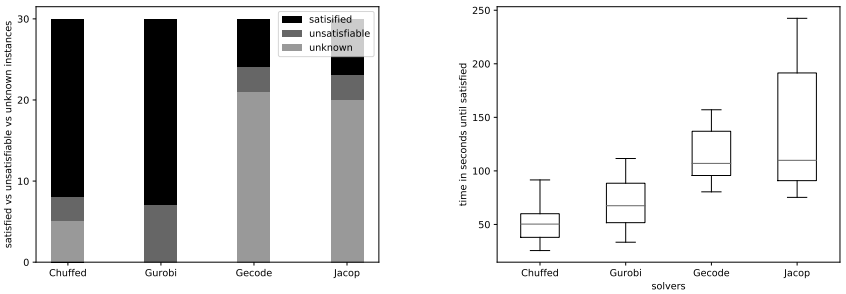

Fig. 2. Comparisons of CP solvers for 30 artificial test instances with a one hour time limit. Left: Feasibility statistics, where we see the stronger performance of Chuffed and Gurobi. Right: Boxplots for the times until a feasible solution was found for satisfied instances.

We start with a high objective value since many shifts are unassigned and are rewarded for assigning those. This is done until no more extensions are possible. $\Delta f(A, A')$ might also be slightly negative (increase in objective value) due to the fairness constraints, therefore we shift this difference by one. After each iteration when all ants have constructed their solutions, pheromone evaporation is performed, controlled by the parameter ρ, and the so-fast-best solution A_{bs} is used to increase the respective pheromones by $\Delta\tau = \frac{1}{f(A_{\text{bs}})}$.

5 Computational Study

We created 30 random artificial instances for a CP-solver benchmark and test the whole exact and hybrid approach on four real-world instances. For the artificial instances, we sampled the numbers of houses from $\{6, \dots, 8\}$, the numbers of events from $\{2, \dots, 11\}$, and the number of employees from $\{170, \dots, 249\}$. We considered a planning horizon of 30 days and created shift requirements following different load patterns (peaky, steady, weekend-only, etc.) and randomly sampled employees' desired shifts and availabilities following observations from the real-world instances. 23 of the artificial instances are satisfiable, seven not; all of the real-world instances are satisfiable.

All tests were conducted on an Intel Xeon E5-2640 processor with 2.40 GHz in single-threaded mode and a memory limit of 32GB. We used Python 3.6 for the implementation of the ACO, Java 11 for the implementation of the VND, and MiniZinc 2.2.3 with the backends Chuffed 0.10.3, Gecode 6.1.0, Gurobi 8.1.0, and JaCoP 4.6.

In Fig. 2 we see a comparison of the four different CP solvers we tested on the artificial instances. Chuffed and Gurobi using the binary formulation gave superior results in terms of number of instances they could satisfy or prove unsatisfiable within a CPU time limit of one hour, and the CPU time needed to satisfy an instance. We chose Gurobi as basis for the hybrid algorithm with

Table 2. Comparisons of exact and hybrid algorithms on four different real-world instances with resulting weighted objective values after a time limit of one hour.

instance	$\|W\|$	$\sum_s R_s$	$\|D\|$	t_f $[s]$	Exact	CSP	VND	ACO	ACO+VND
Sep 2018	184	928	30	123	1.290	7.206	0.418	0.530	**0.407**
Oct 2018	253	1079	31	238	-	6.901	-	0.553	**0.501**
Feb 2019	172	685	28	56	0.243	6.634	**0.093**	0.239	0.141
Apr 2019	170	947	30	124	2.547	7.218	1.018	0.849	**0.716**

the ACO, since it could satisfy every satisfiable instance within a couple of minutes. In the other variant, where we start from a complete solution provided by the CP-solver, we use JaCoP which performed better than Gecode in our experiments.

In Table 2, the main results of our real-world instances are described. Every algorithm is given a time limit of one hour. We compare the exact approach using our constraint optimization model in the set formulation with JaCoP as backend solver with the hybrid variants. In ACO and ACO+VND we start from a small basic feasible solution provided by Gurobi, extended it by an ACO and possibly further improve it with the VND. Values t_f denote the times needed until a feasible solution was provided, the rest of the time is then used either by ACO or shared evenly among ACO and VND. In the other variant, we take the first feasible solution from JaCoP in optimization mode which is rather complete and feed it directly into the VND. We conducted the tests with six ants per iterations and ACO parameters $\alpha = \beta = 1$ and $\tau^{\min} = 1.0, \tau^{\max} = 10.0$, and $\rho = 0.9$. For the Feb 2019 instance, the CP+VND only approach gave the best objective value after one hour, for the others, CP+ACO+VND gave better results. For the Oct 2019 instance JaCoP could not find a feasible solution within the allowed time limit.

In Table 3, we take for each real-world instance the best solution and show the shift coverage compared to a theoretical upper bound and the unweighted soft constraints. In the February instance there is high availability of workers which allows for a high shift coverage, in the September and October instance, we get close to the theoretical upper bounds. Since the VND always hits the time limit, further improvements are expected to be possible; a corresponding analysis of converged solutions will be provided in the master thesis of Stephan Teuschl [7].

6 Conclusion

We introduced a casual employee scheduling problem arising in an Austrian association, where employees have to be assigned to shifts to satisfy demands at locations for a given planning horizon where a number of hard constraints has to be met and violations of fairness and preference soft constraints shall be minimized. We created two different MiniZinc constraint programming models for

Table 3. Shift coverage and unweighted soft constraint objective values for best solutions of real-world instances, where u is the number of unassigned shifts, c the shift coverage, and u_c the upper bound of the shift coverage, calculated by the sum of the number of shifts of the employees divided by the total requirements.

instance	best f	c	u_c	u	g_u	g_f	g_{ff}	g_{np}
Sep 2018	0.407	0.873	0.888	117	0.153^2	0.243^2	0.172^2	0.292^2
Oct 2018	0.501	0.868	0.918	142	0.174^2	0.319^2	0.130^2	0.281^2
Feb 2019	0.093	0.988	1.168	8	0.022^2	0.262^2	0.092^2	0.107^2
Apr 2019	0.716	0.767	0.817	221	0.227^2	0.303^2	0.161^2	0.291^2

this problem. The first model is used only for satisfying the hard constraints by assigning a small number of employees to shifts, which is then the basis for further extensions by an ant colony optimization algorithm together with a variable neighborhood descent. We compared four different backend solvers on 30 artificial benchmark instances for our problem, and Chuffed and Gurobi turned out to be the best choices. In the second model, the float-capable backend solvers JaCoP and Gecode are used to solve the optimization model considering the soft constraints up to the first feasible solution, which is then passed to the VND for further improvement. We compared the exact and the hybrid algorithms on four real-world instances with a CPU time limit of one hour and found that the hybrid approaches provided superior results. Further research is to be conducted to make use and measure the impact of different initial solutions, different orderings of neighborhoods in the VND, and parameter tuning of the ACO. An in-depth study of the presented casual employee scheduling problem and its solution methods will be given in the master thesis of Stephan Teuschl [7].

References

1. Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., De Boeck, L.: Personnel scheduling: A literature review. European journal of operational research **226**(3), 367–385 (2013)
2. Burke, E.K., De Causmaecker, P., Berghe, G.V., Van Landeghem, H.: The state of the art of nurse rostering. Journal of Scheduling **7**(6), 441–499 (2004)
3. Dorigo, M., Stützle, T.: Ant colony optimization: overview and recent advances. In: Handbook of metaheuristics, pp. 311–351. Springer (2019)
4. Gutjahr, W.J., Rauner, M.S.: An ACO algorithm for a dynamic regional nurse-scheduling problem in austria. Computers & Operations Research **34**(3), 642–666 (2007)
5. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: MiniZinc: Towards a standard CP modelling language. In: International Conference on Principles and Practice of Constraint Programming. pp. 529–543. Springer (2007)
6. Stützle, T., Hoos, H.H.: MAX–MIN ant system. Future generation computer systems **16**(8), 889–914 (2000)
7. Teuschl, S.: Casual Employee Scheduling with Constraint Programming and Metaheuristics. Master's thesis, TU Wien, Austria (2019), to appear