Noname manuscript No.
(will be inserted by the editor)

Modeling and Solving a Real-Life Multi-Skill Shift Design Problem

Alex Bonutti · Sara Ceschia · Fabio De Cesco · Nysret Musliu · Andrea Schaerf

Abstract In this work, we consider the shift design problem and we define a novel, complex formulation arising from practical cases. In addition, we propose a new search method based on a fast Simulated Annealing (SA), that, differently from previous approaches, solves the overall problem as a single-stage procedure. The core of the method is a composite neighborhood that includes at the same time changes in the staffing of shifts, the shape of shifts, and the position of breaks. Finally, we present a statistically-principled experimental analysis on a set of instances obtained from real cases. Both instances and results are available on the web for future comparisons.

Keywords Shift design · Workforce scheduling · Simulated annealing · Local search

1 Introduction

Shift design is an important phase of the workforce management process (see, e.g., [13]). It consists in setting start time, length, and staffing level of all working shifts of an organization. The assignment of specific workers to shifts is done in the subsequent workforce scheduling phase, based also on constraints and preferences of both the organization and the workers.

According to labor regulations, in many industrial sectors, shifts must include *breaks* for employees for both resting and eating. Therefore, breaks must

A. Bonutti and F. De Cesco
EasyStaff s.r.l., Via Adriatica, 278 - 33030 Campoformido (UD), Italy.
E-mail: {alex,fabio}@easystaff.it

S. Ceschia and A. Schaerf
DPIA, University of Udine, Via delle Scienze 206, Udine, Italy.
E-mail: {sara.ceschia,schaerf}@uniud.it

N. Musliu
DBAI, Technische Universität Wien, Austria.
E-mail: musliu@dbai.tuwien.ac.at

be taken into account when designing shifts, in order to be able to meet precisely the staffing requirements and thus ensure the desired service level. For this reason, we address the *shift and break* design problem, rather than shift design only.

In addition, we consider the situation, arising from practical cases, in which workers have different skills, and a common set of shifts must be selected for all skills. To this aim, we propose a novel multi-skill formulation with breaks, that, to our knowledge, has not been addressed in the literature.

The problem consists in selecting a set of shifts, and the corresponding staffing level for each skill, so as to minimize the cost of the following objectives:

1. understaffing and overstaffing for each timeslot;
2. number of distinct shifts used;
3. discrepancy of the average length of shifts to given limits.

We design a new search method based on a fast Simulated Annealing (SA), that, differently from previous approaches (see [14]), solves the overall problem as a single-stage procedure. The core of the method is a composite neighborhood that includes, at the same time, changes in the staffing of shifts, the shape of shifts, and the position of breaks.

In addition, we collect a set of challenging instances coming from real world applications in call-centers and supermarkets.

The experimental analysis makes use of statistically-principled techniques for the tuning of the numerous control parameters of the search method. Indeed, they include both the parameters of SA and the distributions for the selection of the candidate moves out of the composite neighborhood.

The outcome is that a non-trivial combination of the neighborhoods turns out to be the best one to solve the proposed instances. In order to enable future comparisons, all instances and their best results are available on the web along with a validation tool at `http://bitbucket.org/satt/shift-design`.

The paper is organized as follows. Section 2 provides the problem formulation. Section 3 describes the related work. Our search method is explained in Section 4. Section 5 illustrates our experimental analysis, including a description of the instances. Finally, conclusions are drawn is Section 6 along with future work.

2 Problem formulation

The problem consists in designing a set of shifts and assigning the number of workers to them, relying on a set of *shift types* (or shift templates) and workforce requirements.

The problem formulation includes the following main entities:

Planning Granularity: The length of the indivisible step for planning (typically 10' or 15'), used to divide the planning period into discrete *timeslots*.

Planning Horizon: Number of days of planning (typically 7, from Monday to Sunday). The schedule is assumed cyclic, so that the shifts that cross

Name	Earliest start	Latest start	Minimum length	Maximum length	Shift granularity	Break	Availability
Morning	06:00	08:00	420	480	60	No	*
Day	09:00	12:00	420	540	60	Yes	*
Evening1	13:00	15:00	420	480	60	No	*
Evening2	13:00	15:00	420	540	60	Yes	Mon–Fri

Table 1 Shift types

Name	Break length	Min break distance from start	Min break distance from end	Min break start	Max break end
Day	60	120	120	12:00	17:00
Evening2	60	60	60	*	*

Table 2 Additional data for shift types with breaks

the midnight of the last day contribute to fulfill the requirements of the morning of the first day.

Requirements: The requirements specify for each skill, for each timeslot of each day of the planning horizon, the number of requested workers. This number is not strict, so that overstaffing and understaffing are allowed, but penalized in the objective function.

Shift Types: Each shift belongs to a shift type, which sets several constraints on the shape of the shift:

- Minimum and maximum length (including the break, if present).
- Earliest and latest start time.
- Granularity of the shift length, called *shift granularity*, that must be a multiple of the planning granularity.
- Set of planning days in which it can be used.
- Break presence (Boolean-valued).

For shift types with break, the following additional data is included:

- Break length.
- Minimum distance of the break from the beginning of the shift.
- Minimum distance of the break from the end of the shift.
- Minimum start time of the break.
- Maximum end time of the break.

In our setting, we consider only the long break for the meals (that normally lasts 1 hour), therefore there is at most one break for each shift. Shorter breaks, used for resting, are not planned, but rather assumed to be taken deliberately by the operator according to the current situation.

As an example, consider the shift types given in Table 1. The additional information for the shift types with break are supplied in Table 2. The character $*$ means that there is no limit.

For each shift type, there are many different shifts compatible with its limits. For example, assuming a planning granularity of 15', there are 18 shifts for the shift type Morning of Table 1. In fact, there are 9 possible start times, namely 6:00, 6:15, ..., 8:00, and 2 possible lengths: 420 and 480 minutes (the shift granularity is 60'). In presence of breaks, the number of shifts grows sig-

Timeslot	Mon	Tue	Wed	Thu	Fri	Sat	Sun
06:00–07:00	0/0	2/1	2/1	4/2	2/1	2/1	1/0
07:00–08:00	1/1	3/1	2/2	5/3	3/2	3/2	1/1
08:00–09:00	3/1	4/2	3/2	5/2	4/2	4/3	1/1
09:00–10:00	4/2	3/2	4/3	5/4	4/2	5/3	1/1
10:00–11:00	3/2	4/2	5/3	5/3	4/2	6/2	2/1
11:00–12:00	3/2	3/2	4/3	4/3	3/3	4/4	2/1
12:00–13:00	3/2	3/2	4/3	3/3	4/3	3/3	2/2
13:00–14:00	3/2	3/2	3/3	4/2	3/2	4/2	2/1
14:00–15:00	3/2	4/2	3/3	3/2	3/2	5/2	2/1
15:00–16:00	3/2	4/2	3/2	3/3	4/3	5/2	2/1
16:00–17:00	3/2	4/2	4/2	3/2	4/3	4/3	2/1
17:00–18:00	3/2	3/1	4/2	3/2	4/2	4/3	2/1
18:00–19:00	3/2	2/2	3/3	3/2	3/2	4/2	2/1
19:00–20:00	3/2	2/2	4/2	3/2	3/2	4/2	1/1
20:00–21:00	2/1	2/1	2/2	2/2	2/2	2/2	2/1
21:00–22:00	2/2	2/1	2/2	3/2	2/2	3/1	2/1

Table 3 Requirements for 2 skills

nificantly, due to all possible distinct positions of the break itself. For example, there are 140 and 162 shifts compatible with the types Day and Evening2, respectively. The possible positions of the break with respect to the start of the shift are set according to the shift granularity (and not to the general planning granularity). For example, if a Day shift starts at 11:15, the break can take place at 12:15 or 13:15, but not at 12:45 because in this case the shift granularity is 60'.

An example of requirements is given in Table 3 for a case with two skills with a planning granularity of 60 minutes. It can be observed that in this case there is no night work, and therefore the cyclicity of the schedule is not relevant. In some cases, the requirements are spread all around the 24 hours, and then it is necessary to take into account the coverage that stretches from one day to the morning of the next one, and also from the last day to the morning of the first one.

We consider the following four objectives:

UnderStaffing (US): The sum of the positive differences between the requirements and the staffing level for all timeslots, all days, and all skills.

OverStaffing (OS): The sum of the positive differences between the staffing level and the requirements for all timeslots, all days, and all skills.

NumberOfShifts (NoS): Number of distinct shifts used in the solution. Two shift are considered distinct if they have different start, different length, or different break position.

ShiftLengthDiscrepancy (SLD): For every skill, the average length of the shifts (ALS) is obtained by multiplying the length (excluding the break) of each shift by its total number of workers of that skill (for all days), summing up for all shifts, and then dividing the result by the total number of workers of that skill (for all days). The ALS is compared to the predefined minimum and maximum values and all possible discrepancies for all skills are summed up into the SLD.

Shift					Staffing						
Shift type	Start	End	Length	Break	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Morning	06:00	13:00	420	—	0/1	2/1	2/1	2/2	2/1	2/1	1/0
Morning	07:00	15:00	480	—	1/0	0/0	0/1	3/1	1/1	1/1	0/1
Morning	08:00	15:00	420	—	2/0	0/0	1/0	0/0	0/0	0/0	0/0
Day	09:00	16:00	420	12:00–13:00	0/1	2/1	1/1	0/1	1/0	2/0	1/0
Day	11:00	18:00	420	15:00–16:00	0/0	1/0	0/0	0/0	1/1	0/1	0/0
Day	12:00	21:00	540	15:00–16:00	0/1	0/1	1/1	0/0	0/0	1/0	1/0
Evening1	15:00	22:00	420	—	2/0	2/0	1/1	2/2	2/2	3/2	1/1
Evening2	15:00	22:00	420	20:00–21:00	1/1	0/1	2/0	1/0	1/0	0/0	0/0

Table 4 Solution for the instance obtained from Tables 1– 3

A possible solution to the instance described in Tables 1–3 is shown in Table 4. The evaluation of this solution reveals that it has 1560 minutes of OS, 1440 of US, summed up for the two skills. Figure 1 shows an excerpt of the difference between the coverage and the requirements for skill 1 on Monday.

The number of distinct shifts is 8, which, multiplied by the corresponding weight, represents the cost of the objective NoS.

The average shift length is 414 minutes (6h 54') for skill 1 and 420 minutes (7h) for skill 2 (break excluded). This solution has been obtained setting the maximum shift length to 7h for both skills (the minimum is set to 6h), therefore it has 0 violations for the objective SLD.

We can notice that the staffing of the shift of type Evening2 is 0 for both skills on the last two days (Saturday and Sunday). This is not incidental, but due to the fact that this shift type is available only from Monday to Friday.

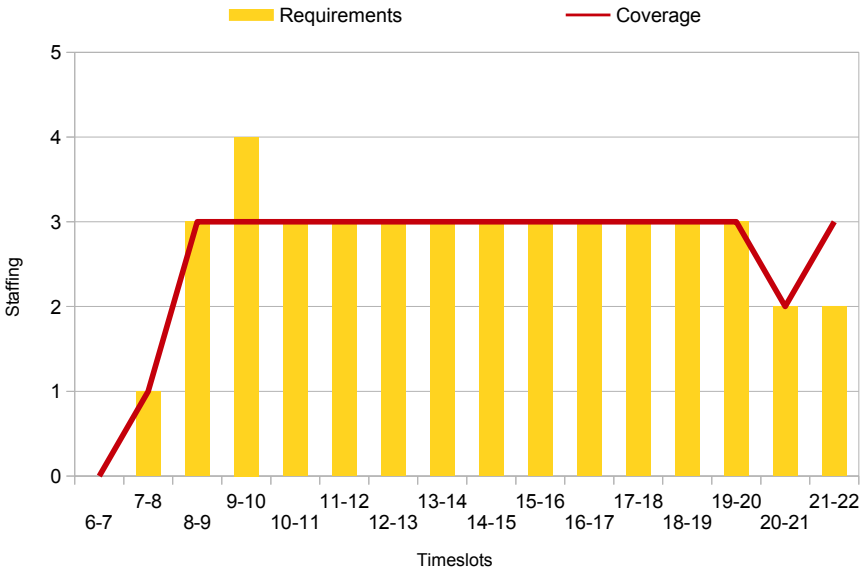

Fig. 1 Requirements and coverage for the solution in Table 4 for skill 1 on Monday

3 Related work

The formal definition of the shift design problem is given in [21]. In that paper a local search algorithm and different neighborhood operators have been proposed to solve this problem. The proposed algorithm has been included in the scheduling system Operating Hours Assistant and it has been applied for solving shift design problems in different companies. In [13] the complexity of the shift design problem has been analyzed and the local search technique of [21] has been further extended by a new strategy which combines different neighborhood relations. Further, a greedy algorithm that deployed a min-cost max-flow procedure has been developed and the hybridization of this technique with local search has been investigated.

The integration of breaks into shift design has been considered in [16], where a simple greedy strategy has been applied to schedule three or four breaks per shift after the generation of shifts. The scheduling of larger number of breaks (as a separate phase after shift design) has been investigated for applications in airports ([7] and [30]) and call centers [6]. Solving the shift design problem together with break scheduling by a method that combines constraint programming and local search has been investigated in [17]. An overview of previous work on shift design and break scheduling is provided in [14]. All the works mentioned above deal with the single-skill employees. To the best of our knowledge, the multiple-skills have not been considered for the shift design problem yet.

A problem related to shift design, called shift scheduling, has been investigated extensively in the literature. The main differences between these two problems is that in shift scheduling usually the shifts are generated for a single day, whereas in shift design the whole week is considered and the problem is cyclic. In addition, the minimization of the number of shifts and the deviation of the average number of duties per week to given limits (in this paper this objective is achieved by minimizing the discrepancy of the average length of shifts to given limits) is only considered in shift design.

The shift scheduling problem has been solved mainly by exact approaches based on Integer Programming (IP) models. A set-covering formulation for shift scheduling traces back to Dantzig [12]. This formulation is based on enumeration of all feasible shifts based on possible start times, duration, breaks, and time windows for breaks. Following that early work, many other IP formulations for shift scheduling have been proposed in the literature (see, e.g., [3–5, 23, 26]).

Shift scheduling in multi-skill call centers has been considered by Bhulai *et al* [8], using an approach based on both a heuristic method and an IP model. In [22] and [10] shift scheduling problems with a planning period of one day, and at most three breaks (two 15 minutes breaks and a lunch break of 1 hour) have been considered. The application of automata and context-free grammars to formulate constraints on sequences of decision variables is shown in [10]. Quimper and Rousseau [22] make use of regular and context-free languages to model the regulations for the shift scheduling problem and apply Large

Neighborhood Search to solve the overall problem. In addition to the previous model, the authors apply their methods in single-activity and multiple-activity shift scheduling problems. A new implicit formulation for multi-activity shift scheduling problems using context-free grammars has been proposed by Côté *et al* [11]. A Tabu Search algorithm for the shift scheduling problem has been investigated by Tellier and White [25].

Another related problem, called tour scheduling, has been also extensively investigated in the literature. The tour scheduling problem combines shift and days-off scheduling. The planning period consists of several days and the tours generated must optimize different criteria. An example of use of Benders decomposition for tour scheduling is given in [24]. A survey of different solving approaches for tour scheduling problems is given by Alfares [2]. In this paper we do not consider days-off scheduling and, as we mentioned above, our problem has several differences to the shift scheduling problem that is solved separately or as a part of the tour scheduling problem.

4 Search method

Our solution method is based on local search, whose main features are presented in the following sections.

4.1 Search space and initial solution

As search space, we use a direct representation of the solution, which is composed by a set of shifts with the corresponding staffing level for all days and skills (like in Table 4). However, similarly to previous work (see, e.g., [14]), the states of the search space are not composed only of the shifts actually used by workers, as in Table 4, but also by a set of so-called *inactive* shifts. Inactive shifts have no workers, but can become active by means of moves that increase the number of workers or transfer a worker from another shift. When an inactive shift becomes active, a new random inactive shift of the same type is generated and inserted in the current state.

The initial solution is obtained by creating a random number of distinct active and inactive shifts for all types. The designated active shifts are populated with random workers up to a level such that the total staffing is equal to the total requirements for all timeslots.

4.2 Neighborhood relations

We implement a search method that uses a neighborhood relation that extends and adapts to our case the one proposed in [13] for the simpler problem. In detail, we consider the union of the following five basic neighborhoods:

AR: Add or remove one worker from a shift.

Tr: Transfer one worker from a shift to another shift of the same type.
ES: Enlarge or shrink a shift by one timeslot either at the beginning or at the end.
MB: Move the break forward or backward by one unit of shift granularity.
Me: Merge two shifts of the same type, transferring all workers of the first shift to the second one.

For neighborhoods ES and MB that modify the shift, a move belongs to the neighborhood only if the obtained shift still belongs to the type of the original one. In addition, a move of these two neighborhoods can be applied only to active shifts. Similarly, the first shift of a Me move must be an active one.

4.3 Simulated Annealing

As a metaheuristic technique, we make use of Simulated Annealing, which was introduced in [20] and [29]. In literature, many variants of SA have been presented and discussed (see, e.g., [1,28]); in this section, we describe the features of our implementation.

The solution procedure starts by creating a random initial state, as described in Section 4.1. Then at each iteration, a neighbor of the current state is randomly selected. However, given that we use a composite neighborhood, we first select the basic neighborhood and then, the specific move within the neighborhood. The latter sampling is done in a uniform way, while the former is driven by five given probability values p_{AR}, p_{Tr}, p_{ES}, p_{MB}, and p_{Me} (with $p_{\mathsf{AR}} + p_{\mathsf{Tr}} + p_{\mathsf{ES}} + p_{\mathsf{MB}} + p_{\mathsf{Me}} = 1$).

A move is always accepted if the difference in the cost between the new state and the current one is less than or equal to 0. Conversely, worsening moves are accepted with probability $e^{-\Delta f/T}$, where Δf is the difference of the cost and T is a parameter of SA, called the *temperature*.

The temperature T is initially set to a high value T_0 and is subsequently updated with a *non-geometric cooling scheme*, already used in [19] and based on *cut-offs*. In practice, instead of sampling a fixed number ns of neighbors at each temperature level, the temperature can be prematurely decreased using the classical geometric cooling scheme ($T_n = \alpha \times T_{n-1}$, where α is the *cooling rate*), if na moves have been already accepted at the current level of temperature (with $na < ns$). This allows to speed-up the search in the initial stages of the solution process, thus saving iterations that can be used after for intensification in the final phase.

The control parameters of the procedure are the cooling rate (α), the number of neighbors sampled at each temperature (ns), the number of neighbors accepted before each cooling step (na), the starting and final temperatures (T_0 and T_{min}), and the five neighborhood probability values (p_*).

5 Experimental analysis

Our solver is written in C++ using the framework EasyLocal++ (v. 3) [15]. All the experiments are executed on an Ubuntu Linux 14.04 machine with 8 Intel® Core™ CPU i7-4770 (3.40 GHz) physical cores, hyper-threaded to 16 virtual cores. A single virtual core has been dedicated to each experiment.

For all experimental analyses and comparisons, we have used the tool JSON2RUN [27], which samples parameter configurations from the *Hammersley point set* [18], and compares them using the F-Race procedure [9]. F-Race is based on the Friedman test to prove statistical significance, using the threshold of confidence set to 95% (p-value = 0.05).

5.1 Instances

Instances have been obtained starting from real data, but combining them in different ways in order to obtain many realistic test cases with different structure. The outcome is a set of 28 challenging instances, divided into four families based on the shift types that they offer, and a toy one used as example in Section 2. All instances are available online at `http://bitbucket.org/satt/shift-design`.

Table 5 reports the main features of the instances in terms of planning granularity, number of skills, number of shift types, presence of the night shift, average slot requirements (per skill) and variation of the requirements (per skill) moving from a slot to the following one. Notice that only the first family (instances MS01–MS04) has 24h requirements (night shift), whereas all the others are based on a 6:00–22:00 work pattern.

The weights of the four components of the objective function are generally set by the operator on any single application. For this experimental analysis we fix them to the values reported in Table 6.

5.2 Neighborhood comparison

The first solver that we test is a Simulated Annealing that uses the union of all neighborhoods. We name this solver as $SA(\mathsf{AR} \oplus \mathsf{Tr} \oplus \mathsf{ES} \oplus \mathsf{MB} \oplus \mathsf{Me})$. However, an analysis of the executed moves revealed that the neighborhood Me is very rarely useful. This observation leads us to consider the solver $SA(\mathsf{AR} \oplus \mathsf{Tr} \oplus \mathsf{ES} \oplus \mathsf{MB})$, excluding the neighborhood Me. This solver, however, turns out to deliver solutions that are not local minima with respect to the neighborhood Me, so that they could be easily improved. We therefore devise the solver $SA(\mathsf{AR} \oplus \mathsf{Tr} \oplus \mathsf{ES} \oplus \mathsf{MB}) \triangleright SD(\mathsf{Me})$ that uses a *steepest descent* as a post-processing step, but using the neighborhood Me alone.

For comparison, we have also performed experiments on the pure steepest descent method $SD(\mathsf{AR} \oplus \mathsf{Tr} \oplus \mathsf{ES} \oplus \mathsf{MB} \oplus \mathsf{Me})$. The outcome is that SD is clearly outperformed by all the SA-based methods (around 70% gap), therefore we do not report its results.

Table 5 Main features of the instances

Instance	Gran.	Skills	Shift types	Shifts	Night shift	Average requirements	Variation requirements
Toy	60	2	4	110	F	2.03/1.3	0.62/0.5
MS01	10	2	2	2272	T	3.00/2.12	0.57/0.53
MS02	10	2	2	2272	T	2.96/2.09	0.55/0.57
MS03	10	2	2	2272	T	2.93/2.09	0.54/0.53
MS04	10	2	2	2272	T	3.13/2.16	0.63/0.59
MS05	15	2	3	527	F	23.62/15.61	3.9/2.8
MS06	15	3	3	527	F	19.61/11.72/8.14	3.4/2.3/1.8
MS07	15	2	3	527	F	11.37/7.68	2.1/1.7
MS08	15	3	3	527	F	9.77/5.54/3.56	1.9/1.4/1
MS09	15	2	3	527	F	12.70/8.43	2.4/1.8
MS10	15	3	3	527	F	10.5/6.51/4.26	2.2/1.6/1.2
MS11	15	2	3	527	F	10.8/7.36	2/1.6
MS12	15	3	3	527	F	9.05/5.54/3.41	1.8/1.4/0.97
MS13	15	2	4	8877	F	23.6/15.6	3.9/2.8
MS14	15	3	4	8877	F	19.6/11.7/8.15	3.4/2.3/1.8
MS15	15	2	4	8877	F	11.4/7.69	2.1/1.7
MS16	15	3	4	8877	F	9.77/5.54/3.56	1.9/1.4/1
MS17	15	2	4	8877	F	12.7/8.43	2.4/1.8
MS18	15	3	4	8877	F	10.5/6.51/4.26	2.2/1.6/1.2
MS19	15	2	4	8877	F	10.8/7.36	2/1.6
MS20	15	3	4	8877	F	9.05/5.54/3.41	1.8/1.4/0.97
MS21	15	2	10	32196	F	23.6/15.6	3.9/2.8
MS22	15	3	10	32196	F	19.6/11.7/8.15	3.4/2.3/1.8
MS23	15	2	10	32196	F	11.4/7.69	2.1/1.7
MS24	15	3	10	32196	F	9.77/5.54/3.56	1.9/1.4/1
MS25	15	2	10	32196	F	12.7/8.43	2.4/1.8
MS26	15	3	10	32196	F	10.5/6.51/4.26	2.2/1.6/1.2
MS27	15	2	10	32196	F	10.8/7.36	2/1.6
MS28	15	3	10	32196	F	9.05/5.54/3.41	1.8/1.4/0.97

Component	weight	measure
US	1	per minute
OS	1	per minute
NoS	60	per shift
SLD	100	per minute

Table 6 Weights of the objectives

The comparison of the performances of the three SA-based solvers, each one tuned as explained in the next section, is shown in Figure 2 for instance MS13. Similar behavior is observed for the other instances. This outcome is confirmed by the Friedman test on the whole set of instances that states that $SA(\mathsf{AR} \oplus \mathsf{Tr} \oplus \mathsf{ES} \oplus \mathsf{MB}) \triangleright SD(\mathsf{Me})$ is indeed statistically superior to both the others.

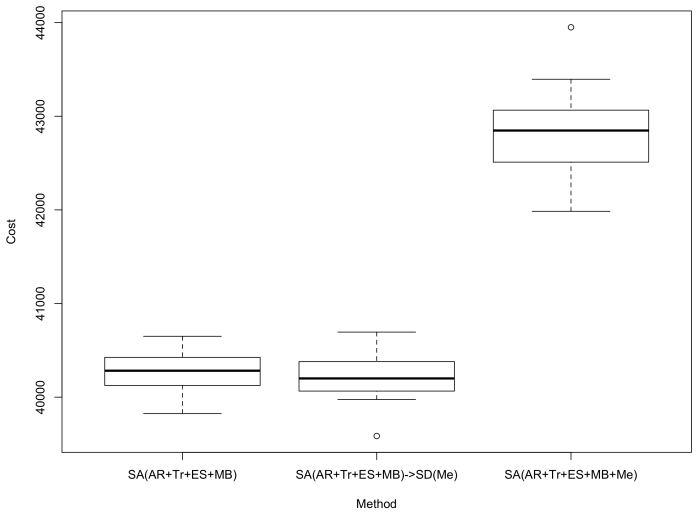

Fig. 2 Boxplot of the results of the three solvers on the MS13 instance

5.3 Parameter tuning

We now describe the parameter tuning phase only for the solver $SA(\mathsf{AR} \oplus \mathsf{Tr} \oplus \mathsf{ES} \oplus \mathsf{MB}) \triangleright SD(\mathsf{Me})$, which resulted to be the winning of the F-Race procedure.

Our SA solver has many parameters to fix and each specific configuration results in a different running time. In order to have the same running time for each configuration, we fix the total number of iterations I and compute the parameter ns from the other parameters, using the following formula

$$ns = I/ - log_{\alpha}(T_0/T_{min})$$

This way the number of iterations is fixed and thus the running time is approximately the same.

For our solver, the iteration budget has to be divided between the SA and the $SD(\mathsf{ME})$ phases so that we have approximately the same computational time as the other SA solvers. In detail, we grant to SA a number of iterations equal to $I \cdot 0.92$, given that it has been experimentally shown that $SD(\mathsf{ME})$ takes about 8% of the CPU time.

For the selection of the parameter configurations, we have used the bound ranges reported in Table 7.

Preliminary experiments showed that in this setting α is not significant, we therefore set α to the fixed value 0.99 and we focused on the other control parameters T_0, T_{min} and na. This is not surprising because in our setting ns is a function of I and the other parameters, and therefore α only determines

Table 7 Parameter setting (HP for Hammersley point set)

Name	Role	fixed by	#points	range	value
T_0	Start temperature	HP + F-Race	30	[30–5]	18.33
$\delta = T_0/T_{min}$	Expected temperature range	HP + F-Race	30	[100–1000]	128.125
α	Cooling rate	preliminary experiments	–	–	0.99
I	Total iterations	–	–	–	10^8
ns	Iterations per temperature	from I, T_0, T_{min}, and α	–	–	207095
$\gamma = na/ns$	Neighbors accepted ratio	HP + F-Race	30	[0.05–0.3]	0.19
p_{Ar}	Probability of Ar moves	HP + F-Race	100	[0.01–0.3]	0.054 (0.073)
p_{Tr}	Probability of Tr moves	HP + F-Race	100	[0.01–0.6]	0.563 (0.766)
p_{Es}	Probability of Es moves	HP + F-Race	100	[0.01–0.3]	0.085 (0.116)
p_{MB}	Probability of MB moves	HP + F-Race	100	[0.01–0.2]	0.033 (0.045)

the entity of the single step in the temperature and not the actual slope of the cooling trajectory, which is determined by $\delta = T_0/T_{min}$.

We decided to tune δ instead of T_{min} and $\gamma = na/ns$ instead of na, because this turned out to provide a better distribution of the configurations than using T_{min} and na directly.

The winning configuration is shown in Table 7. For the probabilities, we report in parentheses also the normalized value. Unsurprisingly, the Tr neighborhood is the most selected one, given that it is the most atomic and least disruptive change. It is also interesting to observe that the start temperature is relatively low. At this temperature a move that activates one shift (without other effects) would have a probability of acceptance of only $e^{-60/18.33} = 0.0378$.

All of the following experiments were performed using this configuration.

Notice that we refer to the "expected" temperature range because the cutoff-based cooling scheme, allows us to reach T_{min} some iterations in advance depending on the cost landscape and on the ratio γ. These "saved" iterations then are used to carry out further (intensification) moves at the end of the search process.

5.4 Results

Table 8 reports the average values of the different cost components for the $SA(\mathsf{AR} \oplus \mathsf{Tr} \oplus \mathsf{ES} \oplus \mathsf{MB}) \triangleright SD(\mathsf{Me})$ solver, and the average running time (in seconds) on our machine. These results are obtained from 30 repetitions of the solver with the configuration described in Table 7. In addition, on our website we publish the best solution of each singular instance, along with a solution *validator* which allows to certify its cost.

Looking at Table 8, we notice that a large part of the total cost comes from the staffing components (US and OS). For families MS01–MS04, MS05–MS12 and MS21–MS28 the values of these two components are comparable, with the OS cost usually smaller than US. For family MS13–MS20, instead, the gap between the two components is much larger; and this is related to the structure of the shift types, that are characterized by strict limits on the shift length.

Table 8 Results of the $SA(\mathsf{AR} \oplus \mathsf{Tr} \oplus \mathsf{ES} \oplus \mathsf{MB}) \triangleright SD(\mathsf{Me})$ solver

Instance	NoS	US	OS	SLD	Tot. cost	Time (secs.)
Toy	834	858.00	1178.00	0	2870.00	377.37
MS01	910	3070.00	2448.00	0	6428.00	678.55
MS02	834	3311.33	2261.33	0	6406.67	669.75
MS03	848	3066.67	2360.67	0	6275.33	667.45
MS04	856	3601.33	2653.33	0	7110.67	675.44
MS05	1992	15593.00	13484.00	0	31069.00	564.55
MS06	2198	24372.00	18908.00	0	45478.00	588.17
MS07	1466	11053.50	10892.50	0	23412.00	555.76
MS08	1588	14323.00	12025.00	0	27936.00	580.71
MS09	1638	14581.00	14129.00	0	30348.00	556.88
MS10	1532	21556.00	16556.00	0	39644.00	573.68
MS11	1600	11226.00	9860.00	0	22686.00	560.72
MS12	1674	14620.00	11562.00	0	27856.00	586.23
MS13	4996	30854.50	4371.00	0	40221.50	538.56
MS14	5592	31691.00	5120.50	0	42403.50	563.47
MS15	3040	17576.50	3156.50	0	23773.00	542.45
MS16	3430	18892.00	4425.50	0	26747.50	570.92
MS17	3014	23622.00	3534.00	0	30170.00	551.31
MS18	3444	25853.50	4929.00	0	34226.50	574.11
MS19	2968	16030.00	3235.00	0	22233.00	547.99
MS20	3338	17310.50	4046.50	0	24695.00	570.44
MS21	6022	4379.00	4626.00	0	15027.00	568.13
MS22	6718	6036.00	5787.00	0	18541.00	598.06
MS23	3792	3454.00	3280.00	0	10526.00	570.93
MS24	4082	4936.50	4542.50	0	13561.00	597.16
MS25	3870	4586.50	3694.50	0	12151.00	570.62
MS26	4258	6054.50	5352.00	0	15664.50	597.69
MS27	3716	3165.00	3294.00	0	10175.00	565.80
MS28	3980	4507.50	4319.50	0	12807.00	588.33

The cost of the NoS cost components is not negligible and it obviously increases with the number of available shift types. However, we can observe that for family MS05–MS12, the number of shifts generated from the explosion of the shift types is significantly smaller than for others families, but its cost is comparable.

Regarding the component SLD we notice that it is never violated in our analysis on the given instances. Experiments with the weight ten times smaller than the one in Table 6 reveal that it is violated significantly only for the family MS05–MS12. This behavior demonstrates that only for this family the "optimal" average length of the shifts falls outside the given limits.

5.5 Shift types with break vs. without break

In previous works (e.g., [13]), the shift design problem was addressed using shift types without breaks. In this section, we want to show that breaks are not only necessary in practice to let workers to get some rest or have a meal, but may also be useful to obtain a better coverage of the requirements.

Table 9 Comparison of the values of the objectives for family MS13–MS20 with and without (MS*-nb) shift types with breaks

Instance	NoS	US	OS	Tot. cost
MS13	4996	30854	4371	40221
MS13-nb	3792	33659	6979	44431
	-24%	9%	59%	10%
MS14	5592	31691	5120	42403
MS14-nb	4374	35036	8599	48010
	-21%	10%	67%	13%
MS15	3040	17576	3156	23773
MS15-nb	2346	19557	4913	26816
	-22.83%	11%	55%	12%
MS16	3430	18892	4425	26747
MS16-nb	2572	21205	5736	29513
	-25%	12%	29%	10%
MS17	3014	23622	3534	30170
MS17-nb	2098	27247	5872	35217
	-30%	15%	66%	16%
MS18	3444	25853	4929	34226
MS18-nb	2622	30337	6955	39914
	-23%	17%	41%	16%
MS19	2968	16030	3235	22233
MS19-nb	2230	18029	4921	25180
	-24%	12%	52%	13%
MS20	3338	17310	4046	24695
MS20-nb	2652	19553	5806	28011
	-20%	12%	43%	13%

In order to show this, we create a new testbeds identical to the one described in Section 5.1, but we replace all the shift types with breaks with coherent ones without break. We named these instances with the suffix *-nb* (no-break).

Results in Table 9 for family MS13–MS20 demonstrate that removing breaks entails on the one hand a clear reduction on the number of shift, but on the other hand an increase of staffing cost; in particular the OS component has an average variation of about +43% causing an average increase in the total cost of 13%.

This is due to the fact that the staff demands are relatively ragged (see Table 3), and therefore breaks can be used to match small "hollows" in the requirements, that otherwise would result in overstaffing situations.

5.6 Analysis of the weights

In Table 6, we fix the weight of NoS to 60, meaning that the use of one additional shift costs as much as 60' of discrepancy in the staffing.

Looking at the values of the cost components reported in Table 8, this value seems to be reasonable given that the number of used shifts is inline with the

ones used in practical situations. Nevertheless, we would like to investigate the impact of changing the weight of NoS on the other cost components, to see if there is room for general improvement.

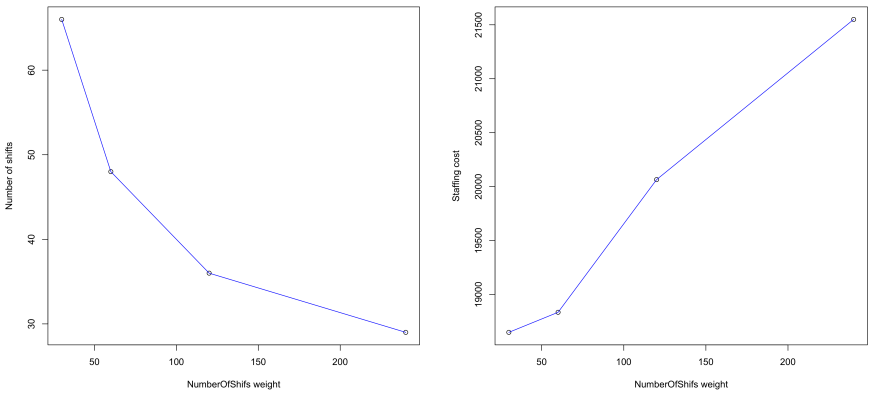

Fig. 3 Impact of the weight of NoS on the staffing cost components (US and OS) and on the number of used shifts

Figure 3 shows the cost obtained for the two staffing components and the resulting number of shifts, for the weight on NoS equal to 30, 60, 120, and 240, respectively. Not surprisingly, both are changed linearly with the weight, but obviously in different directions.

Finally, we investigate on the relative weight of the two staffing components. Indeed, in Table 6 the weight of US and OS is equal. In some cases, however, it might be more reasonable to give more importance to US, given that this could lead to degradation of the quality of service.

Consequently, we decided to perform an additional experiment to study the behavior the staffing components US and OS for higher weights of US.

Figure 4 shows the average values of the US and OS components for a value of the weight of US in the range $[1 - 4]$.

As might be expected, increasing the weight of the US implies a reduction of the violations of the corresponding cost component. Conversely, it has a inverse impact on OS, which increases with the weight. This happens because the solver prefers to guarantee a sufficient coverage on high peaks of demand which can produce some overstaffing on adjacent periods.

5.7 Results on instances with perfect staffing solution

In order to try to assess the absolute quality of the solutions produced by our solver, we developed a generator able to create instances for which the cost of

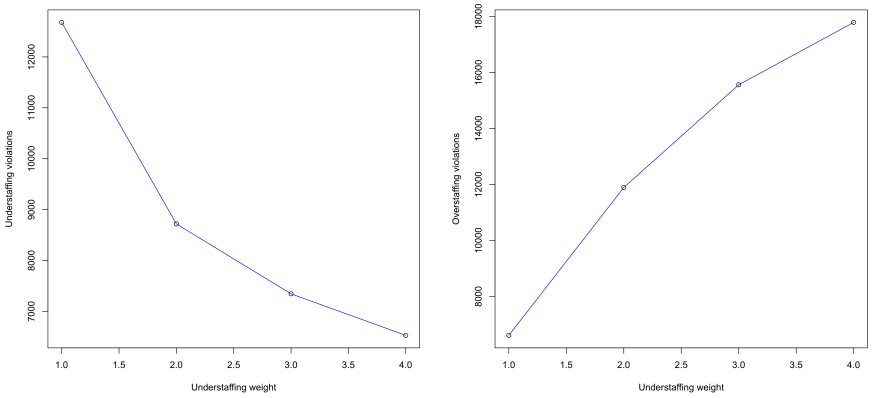

Fig. 4 Impact of the weight of US on the corresponding cost component and on OS

the two staffing components (OS and US) and SLD is zero. This is obtained by starting from generating the solution and creating the requirements matched by the solution itself. We generated 10 instances, named R1–R10 with different shift types and staffing levels.

Table 10 Results on instances with perfect staffing solution

	Average					Best					Optimum
Instance	NoS	US	OS	SLD	Tot. cost	NoS	US	OS	SLD	Tot. cost	NoS
R1	606.6	1353.3	1484.7	0	3444.6	420	0	0	0	420	300
R2	1107.6	1058.9	1575.8	0	3742.2	900	90	90	0	1080	600
R3	1057.2	1940.0	2581.1	0	5578.2	660	60	30	0	750	420
R4	2890.8	1840.5	2317.8	0	7049.1	2040	360	420	0	2820	780
R5	681.0	517.5	522.3	0	1720.8	480	0	0	0	**480**	480
R6	685.8	1002.6	1144.5	0	2832.9	480	0	0	0	**480**	480
R7	1122.6	1890.9	2580.3	0	5593.8	1020	120	360	0	1500	420
R8	979.8	1265.0	1924.7	0	4169.4	660	120	0	0	780	420
R9	615.6	1647.5	1886.9	0	4149.9	540	0	1470	0	2010	300
R10	607.8	1837.8	2219.1	0	4664.7	540	990	0	0	1530	240

Table 10 reports the average value of the cost function for 30 runs and the best solution found by our solver; in addition we show also the cost of the optimal solution, which refers only to the NoS component given that it is the only one not null. We can notice that the solver has been able to find the optimal solution in two cases (R5, R6) and for one instance (R1), it obtained a solution with the two staffing components and SLD equal to 0. Although for other 7 cases there is still a space for improvement, the obtained solutions are useful for typical practical applications.

6 Conclusions and future work

We have proposed a complex SA-based search method for the shift and break design problem. Thanks also to a principled parameter tuning phase, the method has been able to solve the proposed instances to a satisfactory level of quality.

The solver has been successfully tested in a few applications in supermarkets and call-centers, in conjunction with an industrial workforce scheduler, called Workforce Planner, developed by EasyStaff S.r.l.

Future work will aim at the following objectives:

- devise and compare new search techniques and new neighborhood combinations, in order to further improve the current results;
- collect a large set of instances so as to test the methods in a more comprehensive way;
- further investigate on the definition of the suitable weights for the cost components, and the corresponding trade-offs, in order to help the final user to set them.

Acknowledgements Nysret Musliu has been supported for this work by the Austrian Science Fund (FWF): P24814-N23.

References

1. Emile H. L. Aarts and Jan Korst. *Simulated Annealing and Boltzmann Machines*. John Wiley & Sons, New York, 1989.
2. Hesham K Alfares. Survey, categorization, and comparison of recent tour scheduling literature. *Annals of Operations Research*, 127(1-4):145–175, 2004.
3. Turgut Aykin. Optimal shift scheduling with multiple break windows. *Management Science*, 42:591–603, 1996.
4. Turgut Aykin. A comparative evaluation of modelling approaches to the labour shift scheduling problem. *European Journal of Operational Research*, 125:381–397, 2000.
5. Stephen E. Bechtold and Larry W. Jacobs. Implicit modelling of flexible break assignments in optimal shift scheduling. *Management Science*, 36(11):1339–1351, 1990.
6. Andreas Beer, Johannes Gärtner, Nysret Musliu, Werner Schafhauser, and Wolfgang Slany. Scheduling breaks in shift plans for call centers. In *Proceedings of The 7th International Conference on the Practice and Theory of Automated Timetabling*, Montreal, Canada, 2008.
7. Andreas Beer, Johannes Gärtner, Nysret Musliu, Werner Schafhauser, and Wolfgang Slany. An AI-based break-scheduling system for supervisory personnel. *IEEE Intelligent Systems*, 25(2):60–73, 2010.
8. Sandjai Bhulai, Ger Koole, and Auke Pot. Simple methods for shift scheduling in multiskill call centers. *Manufacturing & Service Operations Management*, 10(3):411–420, 2008.
9. Mauro Birattari. *The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective*. PhD thesis, Université Libre de Bruxelles, Belgium, 2004.
10. Marie-Claude Côté, Bernard Gendron, Claude-Guy Quimper, and Louis-Martin Rousseau. Formal languages for integer programming modeling of shift scheduling problems. *Constraints*, 16(1):55–76, 2011.
11. Marie-Claude Côté, Bernard Gendron, and Louis-Martin Rousseau. Grammar-based integer programming models for multiactivity shift scheduling. *Management Science*, 57(1):151–163, 2011.

12. George B. Dantzig. A comment on Eddie's traffic delays at toll booths. *Operations Research*, 2:339–341, 1954.
13. Luca Di Gaspero, Johannes Gärtner, Guy Kortsarz, Nysret Musliu, Andrea Schaerf, and Wolfgang Slany. The minimum shift design problem. *Annals of Operations Research*, 155(1):79–105, 2007.
14. Luca Di Gaspero, Johannes Gärtner, Nysret Musliu, Andrea Schaerf, Werner Schafhauser, and Wolfgang Slany. Automated shift design and break scheduling. In *Automated Scheduling and Planning*, pages 109–127. Springer, 2013.
15. Luca Di Gaspero and Andrea Schaerf. EasyLocal++: An object-oriented framework for flexible design of local search algorithms. *Software—Practice and Experience*, 33(8):733–765, 2003.
16. Johannes Gärtner, Nysret Musliu, and Wolfgang Slany. A heuristic based system for generation of shifts with breaks. In *Proceedings of the 24th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge*, 2004.
17. Luca Di Gaspero, Johannes Gärtner, Nysret Musliu, Andrea Schaerf, Werner Schafhauser, and Wolfgang Slany. A hybrid LS-CP solver for the shifts and breaks design problem. In *Hybrid Metaheuristics*, pages 46–61, 2010.
18. John Michael Hammersley, David Christopher Handscomb, and George Weiss. Monte Carlo methods. *Physics today*, 18:55, 1965.
19. David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon. Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning. *Operations Research*, 37(6):865–892, 1989.
20. Scott Kirkpatrick, Daniel Gelatt, and Mario Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.
21. Nysret Musliu, Andrea Schaerf, and Wolfgang Slany. Local search for shift design. *European Journal of Operational Research*, 153(1):51–64, 2004.
22. Claude-Guy Quimper and Louis-Martin Rousseau. A large neighbourhood search approach to the multi-activity shift scheduling problem. *Journal of Heuristics*, 16(3):373–391, 2010.
23. Monia Rekik, Jean-François Cordeau, and François Soumis. Implicit shift scheduling with multiple breaks and work stretch duration restrictions. *Journal of Scheduling*, 13:49–75, 2010.
24. Monia Rekik, Jean-François Cordeau, and François Soumis. Using benders decomposition to implicitly model tour scheduling. *Annals of Operations Research*, 128(1-4):111–133, 2004.
25. Pascal Tellier and George White. Generating personnel schedules in an industrial setting using a tabu search algorithm. In E. K. Burke and H. Rudova, editors, *The 5th International Conference on the Practice and Theory of Automated Timetabling*, pages 293–302, 2006.
26. Gary Thompson. Improved implicit modeling of the labor shift scheduling problem. *Management Science*, 41(4):595–607, 1995.
27. Tommaso Urli. json2run: a tool for experiment design & analysis. *CoRR*, abs/1305.1112, 2013.
28. Peter J. M. van Laarhoven and Emily H. L. Aarts. *Simulated annealing: theory and applications*. Kluwer Academic Publishers, 1987.
29. Vladimir Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(l):41–51, 1985.
30. Magdalena Widl and Nysret Musliu. The break scheduling problem: complexity results and practical algorithms. *Memetic Computing*, 6(2):97–112, 2014.