Omega

Robust shift scheduling with discretionary rest breaks

--Manuscript Draft--

Manuscript Number:	OMEGA-D-23-01570
Article Type:	Research Paper
Section/Category:	Research Paper - Production Management, Scheduling and Logistics
Keywords:	shift scheduling; break assignment; bilevel programming; robust optimization; heuristics
Abstract:	Shift scheduling is a key task in workforce management which consists in covering the staffing needs by feasible work shifts at minimum labour cost. Work shifts must include rest breaks according to National regulations and collective agreements, which significantly impacts the expected Level of Service (LoS) provided to the customers. Therefore, managers tend to preserve the LoS by rigidly scheduling breaks at centralized level. This affects the well-being of employees, who, especially in large organizations, require more discretion in the choice of rest breaks to reduce work stress. This paper presents a modelling framework to handle discretionary break assignment based on a two-stage robust optimization model. Despite its theoretical complexity, we develop an efficient heuristic algorithm able to find high quality shift schedules under staffing demand patterns common to several service systems. The methodology allows to assess the cost of discretion and to finely determine the best trade-off among discretion, LoS and labor cost.

Robust shift scheduling with discretionary rest breaks

Sara Mattia[1], Fabrizio Rossi [2] and Stefano Smriglio [2]

[1]Consiglio Nazionale delle Ricerche (CNR)
Istituto di Analisi dei Sistemi ed Informatica, Rome (Italy)
email: sara.mattia@iasi.cnr.it
[2]Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila, L'Aquila (Italy)
email: [fabrizio.rossi, stefano.smriglio]@univaq.it

December 6, 2023

Abstract

Shift scheduling is a key task in workforce management which consists in covering the staffing needs by feasible work shifts at minimum labour cost. Work shifts must include rest breaks according to National regulations and collective agreements, which significantly impacts the expected Level of Service (LoS) provided to the customers. Therefore, managers tend to preserve the LoS by rigidly scheduling breaks at centralized level. This affects the well-being of employees, who, especially in large organizations, require more discretion in the choice of rest breaks to reduce work stress. This paper presents a modelling framework to handle discretionary break assignment based on a two-stage robust optimization model. Despite its theoretical complexity, we develop an efficient heuristic algorithm able to find high quality shift schedules under staffing demand patterns common to several service systems. The methodology allows to assess the cost of discretion and to finely determine the best trade-off among discretion, LoS and labor cost.

keywords shift scheduling, break assignment, bilevel programming, robust optimization, heuristics

1 Introduction

Given a discrete time horizon, a demand for employees (*agents*) in each time slot and a set of *works shifts* with associated labor cost, the *shift scheduling problem* consists of deciding how many agents must be assigned to each work shift in order to cover the demand at minimum cost. Since the introduction of the first mathematical formulations [14, 37], this problem has been extensively

studied for a variety of industrial and service systems [2, 16]. In practice, work shifts include *rest breaks*, which are mandatory to reduce workers stress, caused, for instance, by many hours spent in front of computer monitors. National regulations and collective agreements establish feasible break patterns for a work shift in each industrial sector. These translate a complex body of knowledge about physiological and psychological risk factors related to work fatigue into practical restrictions [12]. These topics are currently being debated at National, as well as single organization, level. On the other hand, it is well-known that the impact of rest breaks on the *Level of Service* (LoS) provided to the customers is quite relevant [25]. Therefore, scheduling breaks within a regulatory framework so as to minimize the LoS degradation is an important issue.

Rest breaks can be scheduled off-line, that is, at planning level, or during operations (real-time). Break scheduling at the planning level is addressed in the literature by two types of mathematical programming models: *explicit* models, which associate a decision variable to each feasible pair shift-break, essentially extending the foundational shift scheduling models [14, 37]; and *implicit* models, which consider separate assignment variables for shifts and breaks, enforcing compatibility by suitable linking constraints. Implicit models are more compact than explicit ones, but they require an a posteriori algorithm to build the schedule [3, 5, 35, 41]. Examples of these formulations, along with a comparison of the two approaches, can be found in [4]. Finally, some studies address the break scheduling problem with given shift schedule [46, 6].

An in-depth comparison of off-line and real-time break scheduling policies from a management perspective is reported in [43]. The Authors consider several service systems and their experiments show that, when work shifts are scheduled ignoring breaks, possible periods of surplus staffing do not provide enough capacity to accommodate all of the expected breaks at real time level, resulting in a costly shortage of personnel (understaffing). They also observe that the recovery of additional capacity through the relaxation of time restrictions typically causes frustration among employees.

Overall, the experiments in [43] show that scheduling breaks in advance provides several benefits, while real-time break assignment may significantly affect the LoS even under strict centralised management. On the other hand, in practice, enforcing rest breaks scheduled in advance is often not straightforward because of unpredictable demand variability as well as contingencies on agents side, which can only be handled in real-time. In this case, scientific studies involve the use of simulation or statistical/machine learning methodologies [25]. Additional agent flexibility is sometimes required with actions like sending employees to or recalling them from break [42]. In [22] such a flexibility is exploited within a stochastic programming model in order to design shift schedules which are robust to unpredictable demand fluctuations.

In general, whatever the policy adopted for break scheduling, both practical approaches and scientific literature prioritize mitigation of LoS degradation and do not consider work-related fatigue, which is a fundamental aspect in human resource management. A comprehensive survey on rest breaks from work has been recently published by European Union [12]. It starts from National regulations

and collective agreements, analyses debates in Member States and illustrates the state-of-the-art of studies concerning the fundamental impact of rest breaks on health and productivity of workers. This survey clearly documents that, although workers are strongly involved in determining their rest breaks, time pressure often prevents them from enjoying their breaks in a satisfactory way. At the same time, significant benefits are observed when rest breaks are spent with some comfort and flexibility (see also [21]). In [11, 44], the relationship between the workday fatigue and the degree of autonomous choice associated with the lunch break is investigated. In [13] a thorough analysis of the impact of autonomy of airport security officers in deciding rest breaks is presented. In [36] a formal economic model is presented to assess the benefits of discretionary break periods perceived by workers and an extensive quantitative study is documented in the context of New York taxi drivers. The results of this study noticeably show that workers value discretionary breaks highly, as they would require a 23% increase in daily revenue in order to be induced to accept a fixed break schedule. Moreover, a degradation in labor supply is measured when a mandatory break policy is implemented. The study illustrated in [10], carried out at the General Hospital of Vienna, shows that self-determined rest breaks were effective in reducing acute fatigue of physicians during work. Finally, we have gathered multiple witnesses, as part of industrial projects, of the strong interest of contact center workers for trading flexible and restorative rest breaks. This is getting more and more critical in several organizations and often requires negotiation between management and trade unions.

A comprehensive review of the operational research literature on this highly multi-disciplinary topic has been recently presented in [47]. Several papers address shift/break scheduling problems where the impact of fatigue on agents productivity is taken into account and rest breaks give some recovery effect. Most of these papers focus on leveraging flexibility in break length or timing so as to minimize the impact of workers fatigue on system performance. For an insightful survey of these studies we refer the reader to [47], particularly, sections 4.2 and 4.3. The authors argue that the development of fatigue mitigation models calibrated to individual workers is the next research frontier. However, to the best of our knowledge, no papers have addressed shift/break scheduling problems under the perspective of workers well-being.

Following this line, in this paper, we investigate optimization models which preserve workers well-being by allowing them to decide their break timing in complete autonomy. In detail, we propose a new methodology devoted to design shift schedules able to preserve high LoS independently from the break assignment policy. In fact, such a decoupling provides a system performance guarantee certificate, so as to pave the way to implementing any real-time break assignment policy. This methodology is based on *robust optimization* [7], which turned out to be able of leveraging system flexibility to improve staffing and agent scheduling decisions under uncertain arrival rates [27, 28, 34]. We redirect this potential toward the design of shifts that can handle not only the assignment of breaks in real time, but even the case where the agents themselves decide when to rest, subject only to the standard regulations. Interpreting this

as a source of uncertainty, we define a two-stage robust optimization model which is hard from both theoretical and practical perspectives. In fact, unlike the model of [34], the standard Benders-like reformulation cannot be applied. To cope with this complexity we formulate the problem as a bilevel program and develop a *nested sequential* type heuristic, exploiting a key problem feature represented by the existence of multiple solutions with the same labor cost but significantly different robustness to agents autonomy. We illustrate a computational experience documenting the performance of this heuristic along with its practical impact. This shows that our methodology can actually translate system flexibility of standard scenarios into a nice compromise between personnel cost and full agents autonomy.
The paper is organized as follows. Section 2 describes the decision process and the corresponding optimization problems. In particular, §2.1 refers to the standard centralized decision-making setting while §2.2 introduces the scenario where both central managers and agents contribute to scheduling decisions. Sections 3 and 4 illustrate the corresponding mathematical developments. In Section 5 the bilevel programming heuristic is detailed and in Section 6 the computational achievements are documented. Finally, in Section 7, some conclusions and future directions are drawn.

2 Decision process and problems definition

Staffing (or *workforce forecasting*) decisions in workforce management are generally taken at the interval-level [2, 25]. This yields a discrete time horizon $T = \{1, \dots, \mathcal{T}\}$ and, for every $t \in T$, a required *staffing level* b_t, representing the number of agents who must be on duty at t in order to guarantee the desired LoS (for instance, an adequate average speed of answer in contact centers). These levels, typically determined by queuing models or simulation [2, 15, 25], have to be covered by a set J of *work shifts*, each of which, say j, corresponds to a sequence of consecutive time periods $T_j = \{s_j, \dots, f_j\}$. The shift duration $f_j - s_j$ depends on the underlying labor contract, which also determines an associated labor cost c_j. Work shifts must also accommodate rest breaks, whose allocation is constrained by National regulations as well as collective agreements [12]. A *break pattern* B_i^j is a subset of T_j where breaks can actually be taken, in compliance with such constraints. The collection of the feasible break allocation patterns for each shift $j \in J$ is denoted by $\mathcal{B}^j$.

The quality of a shift plan depends on two cost functions. The first represents the labor cost of the selected shifts, while the second penalizes the deviations of the actual coverage from the required levels b_t. Specifically, for every $t \in T$, a penalty ϕ_t (ψ_t), is given, representing an economical loss due to *understaffing* (*overstaffing*) at period t. The second cost function is then the sum of such penalties for all periods. In specific contexts, allowing such a coverage flexibility may also require to enforce an upper bound on the level of understaffing and overstaffing, denoted by U_t and O_t, respectively. In the extreme case where

$U_t = 0$, $t \in T$, one just enforces that the demand for employees must be satisfied at each period (eventually, with some overstaffing).

In a traditional decision-making approach, break scheduling decisions are fully controlled by the managers, who construct a shift plan along with the associated break assignment, while employees just receive their daily break pattern at the beginning of each working day. This is referred to as *single decision maker management* (§2.1). We contrast such an approach with a *non-cooperative management* framework, where managers elaborate shift plans, while rest breaks are autonomously decided by the employees, according to in-place regulations (§2.2).

2.1 Single decision maker management

Here we assume that all the decisions are taken by the managers, while the agents play no role at the decision level. In practical settings, the managers either compute break scheduling together with the shift plans (*one-step* approach) or the former is delayed to a later moment, that is, the problem is decomposed into two sequential steps calculating first shift plans and then the scheduling of breaks (*two-step* approach). Variants of these problems arise in a wide variety of application contexts and we refer to [20] for a survey. The problem addressed in the one-step approach is the *Shift-and-Break Scheduling* (SBS) problem:

Problem 2.1. *The SBS problem consists of assigning a number of employees to each shift $j \in J$ and scheduling the required rest breaks so as to minimize the sum of labor and under/overstaffing costs.*

The SBS problem has been extensively studied in the literature and several integer programming models have been proposed to solve it [3, 5, 35, 41].

Differently, the two-step approach builds solution (possibly suboptimal) to the SBS problem 2.1 by sequentially solving two separate problems: the *Shift Planning* (SP) problem and the *Break Assignment* (BA) problem.

Problem 2.2. *The SP problem consists of assigning a number of employees to each shift $j \in J$ so as to minimize the sum of labor and under/overstaffing costs.*

Problem 2.3. *Given a shift plan, the BA problem consists of scheduling the required rest breaks so as to minimize the under/overstaffing costs.*

For example, shifts may be decided at the beginning of the week solving SP, because they represent a *strategic decision*, while rest breaks are assigned at the beginning of each working day through BA, because they correspond to an *operational decision*.
Also the SP problem is traditionally modelled as an integer program (see §3). As the BA problem is concerned, one can find a classification in [17] while a complexity (NP-hardness) proof is given in [46].

Summarizing, the one-step approach leads to better labor deployment and higher LoS but is computationally harder to solve. On the contrary, the two-step approach may produce inefficient schedules [43] but is typically preferred by managers for several reasons: (i) they tend to evaluate the personnel costs by neglecting rest breaks; (ii) it often corresponds to the standard practice, splitting strategic and operational decisions; (iii) deferring break allocation allows to take into account typical contingencies, such as personnel unavailability and demand updates. Note that even in the two-step approach, the whole decision process is controlled by managers, while agents are not involved.
The mathematical models for the problems illustrated above are formally discussed in §3.

2.2 Non-cooperative management

We now introduce a *non-cooperative management* framework, where both agents and managers contribute to scheduling decisions. In this framework, managers decide the work shifts while agents can schedule their rest break as they prefer in real time, according to some restrictions. This is the organization preferred by employees, who perceive a significant benefit from a discretionary management of their break periods [36]. However, it requires some negotiation between managers and employees.
In many contexts, employees are free to ask for a break, within prescribed time windows, and these requests can be accepted or rejected by a supervisor, who is in charge of preserving the expected LoS. The supervisors action is typically exercised by allowing the break only if the current understaffing level does not rise beyond a given threshold. This understaffing control avoids putting undue pressure on working employees and it is in the interest of both managers and employees.
This decision process requires managers to assign agents to work shifts without predetermining their breaks and without knowing in advance when the agents might decide to have a rest break. If work shifts are chosen by solving the SP problem 2.2 and completely ignoring possible agents decisions, the LoS can be greatly degraded by such autonomous choices. Hence, we investigate a non-cooperative framework where the agent decisions are regarded as a source of randomness in the system and the manager has to compute work shifts that are robust to this uncertainty. Then, a *robust optimal solution* is a solution which guarantees the manager that, whatever the decisions taken by the agents, the corresponding realization cost will never exceed a worst case cost calculated a priori. The robust solution can be determined by solving the following *Discretionary Breaks Shift Planning* (DBSP) problem.

Problem 2.4. *The DBSP problem consists of assigning a number of employees to each shift $j \in J$ so as to minimize the sum of labor and the worst case under/overstaffing costs, assuming that rest breaks are self-decided by the employees.*

Observe that, although involved in the decision process, the agents do not have

the same decision power of the managers. That is, our non-cooperative management is a hierarchical decision making process, where the managers decide first (they set the shifts, making a strategic decision), while the agents react to the decisions made by the managers (they choose the breaks, making an operational decision).
The mathematical model for the DBSP problem 2.4 is formally discussed in §4.

3 Models for single decision maker

We refer to pairs (j, B_i^j), $j \in J, B_i^j \in \mathcal{B}^j$, as *work patterns* and denote by I_j the set of those associated with work shift $j \in J$. We also let $I = \bigcup_{j \in J} I_j$. Each work pattern $i = (j, B_i^j) \in I$ is associated with cost $c_{j(i)} = c_j$, that is, it inherits the cost of the corresponding work shift $j(i)$. Finally, we introduce the following integer decision variables:

y_j : number of agents assigned to work shift $j \in J$;
x_i : number of employees associated with work pattern $i \in I$;
$o_t(u_t)$: overstaffing (understaffing) level at period $t \in T$.

Under single decision maker management, if both the shifts and the breaks are computed at the same time (one-step approach), the manager directly finds a suitable allocation of the agents $\mathbf{x}$. If shifts and breaks are computed into separate phases (two-step approach), the first step determines the shift assignment $\mathbf{y}$, while the second one maps the computed $\mathbf{y}$ into a work pattern assignment $\mathbf{x}$ that is suitable for the considered $\mathbf{y}$, that is, into a $\mathbf{x} \in S(\mathbf{y})$, which is the set described below.

Definition 3.1. *Set $S(\mathbf{y})$ describes the feasible work pattern assignments $\mathbf{x}$ for a given work shift assignment $\mathbf{y}$, that is, the vectors $\mathbf{x}$ such that the agents y_j assigned to shift $j \in J$ are split among work patterns $i \in I_j$.*

$$S(\mathbf{y}) = \left\{ \mathbf{x} \in \mathbb{Z}_+^{|I|} : \sum_{i \in I_j} x_i = y_j, j \in J \right\}$$

3.1 Models for the two-step approach

The two-step approach requires the solution of the SP problem 2.2 and the BA problem 2.3 in sequence.
Problem SP was originally formulated by Dantzig [14], who introduced covering constraints $\sum_{j \in J: t \in T_j} y_j \geq b_t, \quad t \in T$, enforcing that the required staff is actually present in each slot. This model has been used in several successive applications, such as [37]. However, it is well-known that it may yield in practice highly "suboptimal" schedules [25]. To overcome this drawback, a flexible version of the Dantzig model has been introduced in [34] (a similar idea is also exploited in [27]) by allowing understaffing at slot level but penalizing it in the

objective function. Here, we further refine the flexible model by imposing upper bounds on the understaffing or overstaffing which can occur at each time slot. Our flexible model reads as follows.

$$\begin{aligned}
\text{SP-IP} \quad & \min \sum_{j\in J} c_j y_j + \sum_{t\in T}(\phi_t u_t + \psi_t o_t) \\
& \sum_{j\in J: t\in T_j} y_j + u_t - o_t = b_t \qquad t \in T \qquad (1)\\
& u_t \le U_t \qquad t \in T \qquad (2)\\
& o_t \le O_t \qquad t \in T \qquad (3)\\
& \mathbf{y} \in \mathbb{Z}_+^{|J|}, \mathbf{u}, \mathbf{o} \in \mathbb{Z}_+^{|T|}
\end{aligned}$$

Constraints (1) guarantee that the demand is satisfied by agents assigned to work shifts covering slot t, possibly causing overstaffing or understaffing. Constraints (2) and (3), imposing upper bounds on the over/understaffing at each time slot, have not been considered in [34]. These make the model even more adherent to the practice without affecting its computational tractability.
As observed in [34], SP can be reformulated as a min-cost flow problem on a suitable graph. It is not hard to see that introducing constraints (2) and (3) translates into arc capacities in such a graph, which keeps safe the min-cost flow reformulation [1]. Then, Theorem 2 of [34] implies the following result.

Lemma 3.1. *SP can be solved in time* $O((|T|+|J|)^2 \log|T| + |T|(|J|+|T|)\log^2|T|)$.

Notice that such a complexity still holds if we place lower bounds on $\mathbf{o}$ and $\mathbf{u}$, instead of just upper bounds, and if bounds are imposed (also or only) on the $\mathbf{y}$ variables.

Given a shift solution vector $\mathbf{y}$ of problem 2.2, the BA problem 2.3 can be written as

$$\min_{\mathbf{x}\in S(\mathbf{y})} \sum_{t\in T}(\phi_t u_t + \psi_t o_t) \qquad (4)$$

Denoting by $W(t)$ the set of work patterns $i \in I$ in which the slot t is a working period, i.e., $W(t) = \{i \in I : t \in T_{j(i)} \setminus B_i^j\}$, the corresponding mathematical program reads as

$$\begin{aligned}
\text{BA}(\mathbf{y})\text{-IP} \quad & \min \sum_{t\in T}(\phi_t u_t + \psi_t o_t) \\
& \sum_{i\in W(t)} x_i + u_t - o_t = b_t \quad t \in T \qquad (5)\\
& \sum_{i\in I_j} x_i = y_j \qquad j \in J \qquad (6)\\
& \mathbf{x} \in \mathbb{Z}_+^{|I|}, \mathbf{u}, \mathbf{o} \in \mathbb{Z}_+^{|T|}
\end{aligned}$$

Constraints (6) ensure that, if y_j agents are assigned to work shift $j \in J$, each of them receives a break assignment corresponding to a valid work pattern $i \in I_j$. Constraints (5) is used to compute the over/understaffing corresponding to the chosen $\mathbf{x}$. This formulation is close to the *Set Partitioning* model presented in [17], with the difference that shifts can be assigned to multiple agents (as stated by the given $\mathbf{y}$ vector) instead of just one. The BA problem 2.3 has been first proved to be NP-hard in [46], even when the possible $\mathbf{x}$ vectors to be associated with the provided $\mathbf{y}$ are explicitly listed and polynomially many. A nice classification of BA problems along with complexity results are illustrated in [17].
As for the practice, a common policy is to improve schedule adherence by assigning rest breaks in real time only at slots with overstaffing. However, it has been shown that this may not allow to accommodate all of the required breaks [43].

3.2 A model for the one-step approach

Consider now the case where the managers directly compute shifts with breaks, that is, a feasible $\mathbf{x}$ vector. The formulation of the SBS problem 2.1 reads as follows.

$$
\begin{aligned}
\text{SBS-IP} \quad & \min \sum_{i \in I} c_{j(i)} x_i + \sum_{t \in T} (\phi_t u_t + \psi_t o_t) & & \\
& \sum_{i \in W(t)} x_i + u_t - o_t = b_t & t \in T & \quad (7) \\
& u_t \leq U_t & t \in T & \quad (8) \\
& o_t \leq O_t & t \in T & \quad (9) \\
& \mathbf{x} \in \mathbb{Z}_+^{|I|}, \mathbf{u}, \mathbf{o} \in \mathbb{Z}_+^{|T|} & &
\end{aligned}
$$

Service constraints (7) ensure that the demand b_t at period t is either exactly covered by the agents on duty or it is satisfied accepting some overstaffing or understaffing, as in SP-IP. Constraints (8) and (9) enforce upper bounds on the amount of under/overstaffing at each time slot. SBS-IP does not contain explicitly the shift assignment variables $\mathbf{y}$ that can be trivially determined as $y_j = \sum_{i \in I_j} x_i$.
Unlike the SP problem, the SBS problem 2.1 cannot, in general, be reduced to a network flow problem or solved in polynomial time. Nevertheless, it keeps some SP properties. Indeed, the min-cost flow reformulation of SP allows to regard all variables of SP as continuous. Although this argument does not apply to SBS, one can still prove that $\mathbf{u}$ and $\mathbf{o}$ can be treated as continuous variables, by generalizing to SBS-IP the optimality condition on $\mathbf{u}$ and $\mathbf{o}$ given in [34].

Lemma 3.2. *For every optimal solution to SBS-IP, $o_t \cdot u_t = 0$, for all $t \in T$.*

Proof. Let $(\bar{\mathbf{x}}, \bar{\mathbf{o}}, \bar{\mathbf{u}})$ be an optimal solution such that there exists $\bar{t}$ with the property that $\bar{o}_{\bar{t}} > 0$ and $\bar{u}_{\bar{t}} > 0$. Let $(\bar{\mathbf{x}}, \hat{\mathbf{o}}, \hat{\mathbf{u}})$ be the solution obtained from $(\bar{\mathbf{x}}, \bar{\mathbf{o}}, \bar{\mathbf{u}})$ replacing $\bar{o}_{\bar{t}}$ and $\bar{u}_{\bar{t}}$ by $\hat{o}_{\bar{t}} = \max\{0, \bar{o}_{\bar{t}} - \bar{u}_{\bar{t}}\}$ and $\hat{u}_{\bar{t}} = \max\{0, \bar{u}_{\bar{t}} - \bar{o}_{\bar{t}}\}$.

Since $\phi_{\bar{t}} > 0$ and $\psi_{\bar{t}} > 0$, $(\bar{\mathbf{x}}, \hat{\mathbf{o}}, \hat{\mathbf{u}})$, which is feasible for SBS-IP, has a better objective value than $(\bar{\mathbf{x}}, \bar{\mathbf{o}}, \bar{\mathbf{u}})$, which is, therefore, not optimal. It follows that, once $\mathbf{x}$ is chosen, the $\mathbf{o}$ and $\mathbf{u}$ variables are automatically determined by the following formulas.

$$o_t = \max\{0, \sum_{i \in W(t)} x_i - b_t\} \quad t \in T$$
$$u_t = \max\{0, b_t - \sum_{i \in W(t)} x_i\} \quad t \in T$$

If such a pair does not satisfy constraints (8) and (9), then no feasible solution with the chosen $\mathbf{x}$ exists. □

From the proof of the Lemma 3.2, one can observe that $\mathbf{o}$ and $\mathbf{u}$ variables do not require integrality restrictions, as u_t and o_t will be either zero or equal to the difference between two integer values (b_t and $\sum_{i \in W(t)} x_i$).

4 Models for non-cooperative management

Consider here the non-cooperative management framework, where agents can autonomously choose their breaks (within given guidelines) and the manager has to compute work shifts that are resilient to their choices. Modelling the DBSP problem (2.4) leads to the following formula:

$$\min_{\mathbf{y}} \left\{ \sum_{j \in J} c_j y_j + \max_{\mathbf{x} \in S(\mathbf{y})} \min_{\mathbf{o},\mathbf{u}} \sum_{t \in T} (\phi_t u_t + \psi_t o_t) \right\} \tag{10}$$

Note that the outer minimization makes the inner $\min_{\mathbf{o},\mathbf{u}}$ redundant in SP-IP and SBA-IP. Here, the presence of the $\max_{\mathbf{x} \in S(\mathbf{y})}$ require to explicitly keep the inner minimization.
Problem (10) can be interpreted as a two stage problem, where the manager controls first stage variables $\mathbf{y}$ (strategic decisions), while the agents control second stage variables $\mathbf{x}, \mathbf{u}$ and $\mathbf{o}$ (decisions at operational level).
The inner problem

$$\lambda(\mathbf{y}) = \max_{\mathbf{x} \in S(\mathbf{y})} \min_{\mathbf{o},\mathbf{u}} \sum_{t \in T} (\phi_t u_t + \psi_t o_t) \tag{11}$$

can be regarded as an *Adversarial Break Assignment* (ABA) problem, modeling the worst case impact of discretionary rest breaks on the LoS. The value $\lambda(\mathbf{y})$ of an optimal solution to the ABA will be referred to as *cost of unsupervised break choices*. Under this perspective, the framework defined by (10) is a *two stage robust optimization* model [8] where $S(\mathbf{y})$ represents the *uncertainty set*, that is, the realizations of the possible choices of the agents that are feasible for a given $\mathbf{y}$.

Thanks to Lemma 3.2, Problem 11 can be modeled as the following Mixed Integer Quadratically Constrained Program:

$$\begin{aligned}
&\lambda(\mathbf{y}) = \max \sum_{t \in T} (\phi_t u_t + \psi_t o_t) \\
&\sum_{i \in W(t)} x_i + u_t - o_t = b_t && t \in T \\
&\sum_{i \in I_j} x_i = y_j && j \in J \\
&o_t \cdot u_t = 0 && t \in T \\
&\mathbf{x} \in \mathbb{Z}_+^{|I|}, \mathbf{o}, \mathbf{u} \in \mathbb{Z}_+^{|T|}
\end{aligned} \tag{12}$$

Taking upper bounds O_t and U_t into account, the quadratic term $o_t \cdot u_t = 0$ can be linearized by introducing a binary variable α_t and considering the constraints:

$$\begin{aligned}
&o_t \leq O_t \alpha_t && t \in T \\
&u_t \leq U_t (1 - \alpha_t) && t \in T \\
&\boldsymbol{\alpha} \in \{0,1\}^{|T|}
\end{aligned} \tag{13}$$

The resulting Mixed Integer Linear Program is referred to as ABA($\mathbf{y}$)-MILP. Note that constraints (13) also imply that variables $\mathbf{o}$ and $\mathbf{u}$ are integral in optimal solutions to ABA($\mathbf{y}$)-MILP, thus integrality on $\mathbf{o}$ and $\mathbf{u}$ can be relaxed. Apparently, problem (10) has the same structure of problem (3) in [34], which was rewritten in a Benders-like form (see §3.3 in [34]). However, the two problems are profoundly different, making it impossible to apply the same approach here. The key difference is that the uncertainty set $S(\mathbf{y})$ of problem (10) depends on the current solution $\mathbf{y}$ and different solutions must be protected against different realizations of the uncertainty. This is a crucial difference with respect to traditional robust optimization models, where the uncertainty set does not depend on the current solution [8, 9, 30, 33, 34].

The characteristic of the uncertainty set $S(\mathbf{y})$, which depends on the current solution $\mathbf{y}$, prevents the use of a classic Benders decomposition approach, as the Benders-like inequalities that can be derived are not *globally* valid, unlike those derived for problem (3) in [34]. It follows that the DBSP must be formulated as a (computationally more difficult) *bilevel program*, as illustrated below.

$$\begin{aligned}
\text{DBSP-BLP} \quad & \min \sum_{j \in J} c_j y_j + \lambda \\
& \mathbf{y} \in \mathbb{Z}_+^{|J|} \\
& \lambda = \max \sum_{t \in T} (\phi_t u_t + \psi_t o_t) \\
& \qquad \sum_{i \in W(t)} x_i + u_t - o_t = b_t \qquad t \in T
\end{aligned}$$

$$\sum_{i \in I_j} x_i = y_j \qquad j \in J$$
$$o_t \leq U_t \alpha_t \qquad t \in T$$
$$u_t \leq O_t(1 - \alpha_t) \qquad t \in T$$
$$\mathbf{x} \in \mathbb{Z}_+^{|I|}, \mathbf{o}, \mathbf{u} \in \mathbb{R}_+^{|T|}, \boldsymbol{\alpha} \in \{0,1\}^{|T|}$$

In the bilevel optimization terminology, the inner problem is called the *follower* problem, while the upper level problem is regarded as the *leader* problem. Similarly, the leader controls variables $\mathbf{y}$, while the follower is in charge of determining the values for variables $\mathbf{x}$ and, hence, $\mathbf{o}$ and $\mathbf{u}$. From the application point of view, as discussed in §2.2, it is neither in the interest of the leader nor of the follower to have large understaffing levels. The dualism leader/follower is fictionally used here to represent the worst case realization of the *unknown* behavior of the agents in choosing their rest breaks.

5 Multiple Solutions Search Heuristic for DBSP

Bilevel programming is NP-hard [23]. The problems where leader wants to minimize the maximum follower objective are known as *bilevel min–max optimization* problems [40], which include the broad class of the *interdiction* problems (see e.g., [31, 38]) as special cases. Here the leader seeks to forbid some choices to the follower so as to force a reduction in the maximum value of the follower objective. Solving bilevel problems coming from real-life applications is very difficult and even designing a good general purpose heuristic approach for a bilevel problem is not an easy task.

The methodologies for solving bilevel problems can be classified into four classes [39]: *single level* transformations, where the bilevel problem is transformed into a single level problem, but exact transformations are possible only under some assumptions [45]; *multi-objective* approaches, where the bilevel problem is transformed into a single level multi-objective problem, but exact transformations are possible only under some strict assumptions [19]; *co-evolutionary* approaches, where the leader and the follower problem are solved in parallel and exchange information to improve the solutions of both problems in the next round [26]; *nested sequential* approaches, where the leader and the follower problem are solved sequentially, with the solution of the leader problem being used in the follower one and the solution of the follower problem being used in solving the leader problem in the next round [18, 29, 32].

DBSP-BLP does not meet the conditions to be reformulated as a single level problem, with one or multiple objectives. In fact, the NP-hardness of the lower level problem [46] forbids to reformulate the bilevel problem as a single level (quadratic or mixed integer) problem by the complementarity conditions of linear duality.

Although fairly effective general bilevel solvers have been recently developed (see [24] for a survey), both numerical issues and slow convergence turned out to hamper the use of such tools for solving DBSP-BLP, even in small-sized cases. Therefore, we propose a nested-sequential matheuristic described in §5.1 that we call *Multiple Solutions Search* (MSS) heuristic.

5.1 The basic idea

As discussed in [34], the SBS problem 2.1 typically admits several optimal solutions $(\mathbf{x}, \mathbf{y})$, which are equivalent from the leader's (manager) perspective. Nevertheless, when discretionary break choices are allowed, values $\lambda(\mathbf{y})$ corresponding to cost equivalent $(\mathbf{x}, \mathbf{y})$ solutions may substantially differ. The following example illustrates this situation.

5.1.1 Example

We consider a typical scenario which will be also the basis of computational experiments in §6. The planning horizon T corresponds to 16 hours (from 6:30 to 22:30) subdivided into 192 time slots of 5 minutes. The set J contains shifts lasting 4 and 6 hours, which can start every 15 minutes. Thus, the first 4-hours shift may starts at 6:30 and the last one at 18:30, leading to a set J^4 of 49 possible shifts. Similarly, the first 6-hours shift may start at 6:30 and the last one at 16:30, leading to a set J^6 of 41 possible shifts. Overall, $J = J^4 \cup J^6$ contains 90 shifts. Breaks take 15 consecutive minutes for all shifts. According to the Italian regulations, shifts in J^4 accommodate only one break that can begin at least 100 and at most 120 minutes after the shift's start, corresponding to 5 workpatterns for each shift ($|I_j| = 5$, for $j \in J^4$). Shifts in J^6 have two breaks, the first can begin at least 100 and at most 120 minutes after shift's start, the second can begin at least 100 and at most 120 minutes after first break's end and, in general, the agent cannot work more than 120 minutes consecutively. This results in 22 work patterns for each shift ($|I_j| = 22$, for $j \in J^6$). Overall, $|I| = 1,147$ workpatterns. Shifts costs are $c_j = 50$ Euro for a 4-hours shift and $c_j = 70$ Euro for a 6-hours shift. The under(overstaffing) costs are $\phi_t = 15$ Euro ($\psi_t = 0$ Euro) for each $t \in T$.
In this example we set $U_t = 2$ and $O_t = 10$ for each $t \in T$ and focus on the following demand pattern: the staffing level b_t is equal to 5 for all time slots from 9:30 to 19:30 and zero in all remaining time slots (blue line in Figure 1). The $\mathbf{x}^*$ in the optimal solution to SBS-IP, whose cost is 740 Euro, corresponds to the following staff schedule (orange line in Figure 1):

4-hour shifts			6-hour shifts			
Agent	Start	Break	Agent	Start	1st Break	2nd Break
1	08:30	10:25	1	08:00	09:40	11:45
2	09:30	11:30	2	08:45	10:25	12:30
3	09:45	11:30	3	09:30	11:30	13:35
4	15:30	17:30	4	12:30	14:10	16:15
5	15:30	17:30	5	13:45	15:45	17:30
			6	14:00	15:45	18:00
			7	14:15	16:15	18:00

Observe that no understaffing occurs and total cost coincide with labor cost, as $\psi_t = 0$.

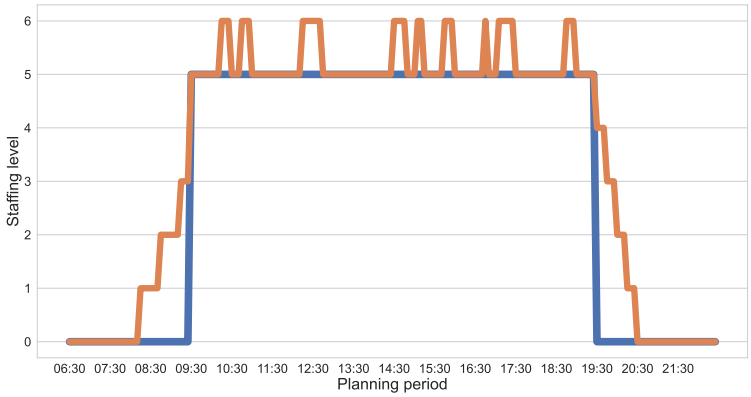


Figure 1: Optimal solution to SBS-IP

Starting from $\mathbf{x}^*$, one can calculate the shift assignment $\mathbf{y}^*$ ($y_j^* = \sum_{i \in I_j} x_i^*$) and estimate the impact of discretionary rest breaks by evaluating the unsupervised break choices cost $\lambda(\mathbf{y}^*) = 485$ Euro. Suppose now to identify a set $\mathcal{K}$ of solutions of SBS-IP with the same cost of $\mathbf{x}^*$ and to evaluate $\lambda(\mathbf{y}^k)$, for each $(\mathbf{x}^k, \mathbf{y}^k) \in \mathcal{K}$. In this example, $\mathcal{K}$ contains 485 solutions whose associated $\lambda(\mathbf{y}^k)$-s are reported in Figure 2.

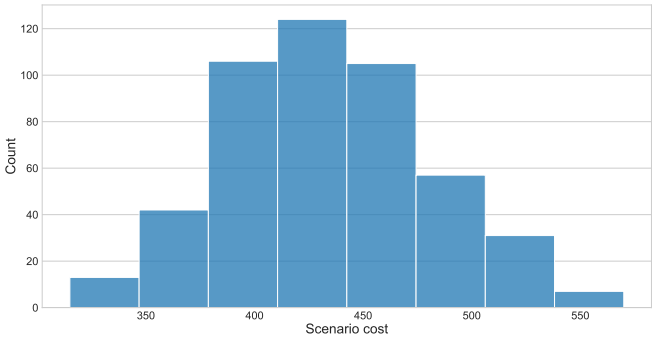
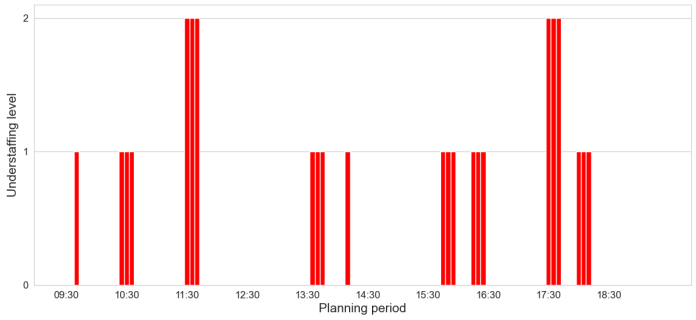


Figure 2: Histogram of the costs $\lambda(\mathbf{y}^k)$ of 485 SBS solutions of equal cost

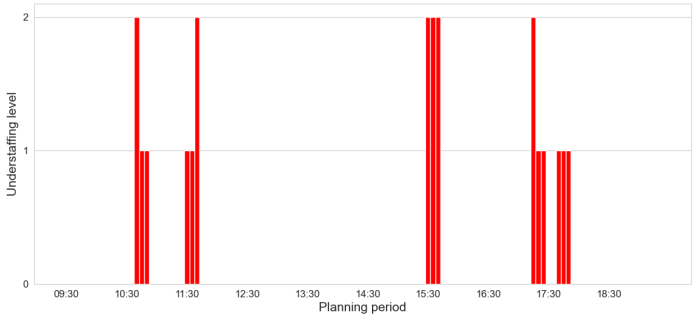
The distribution shows that the range of costs of unsupervised break choices is quite large (from 315 Euro to 570 Euro) and the leftmost SBS solution, say $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, induces a cost $\lambda(\bar{\mathbf{y}}) = 315$ Euro, significantly cheaper than $(\mathbf{x}^*, \mathbf{y}^*)$ ($\lambda(\mathbf{y}^*) = 435$ Euro).
Hence, the manager can considerably reduce the cost of the unsupervised break choices by selecting the shift assignment $\bar{\mathbf{y}}$ below *without increasing the cost of the agents*.

4-hour shifts		6-hour shifts	
Agent	Start	Agent	Start
1	09:00	1	08:30
2	09:00	2	10:00
3	09:30	3	13:00
4	09:30	4	13:30
5	16:00	5	13:30
		6	13:45
		7	14:30

Because we are considering unsupervised break choices, break starting times are not reported in the previous table. However, by construction, the shift assignment $\bar{\mathbf{y}}$ admits $\bar{\mathbf{x}}$ as an optimal solution to the corresponding SBS instance while any other feasible break assignment would induce a cost not larger than $\lambda(\bar{\mathbf{y}})$. Finally, Figure 3a shows the understaffing levels induced by unsupervised break choices with solution $\lambda(\mathbf{y}^*) = 415$ and Figure 3b shows the understaffing level corresponding to the best solution $\lambda(\bar{\mathbf{y}}) = 315$. The latter has clearly less impact on the system LoS.
Hence, the key idea for producing good solutions to DBSP-BLP consists in generating a set of leader's solutions of equal cost and selecting the cheapest w.r.t. the cost of follower reactions. In detail, after an *initialization* phase (§5.2), the MSS heuristic performs a *generation* step (§5.3), returning a set of candidate shift allocations, then an *evaluation* step (§5.4), where each shift allocation



(a) Understaffing levels in SBS-IP solution $(\mathbf{x}^*, \mathbf{y}^*)$ $(\lambda(\mathbf{y}^*) = 435)$



(b) Understaffing levels in SBS-IP solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ $(\lambda(\bar{\mathbf{y}}) = 315)$

Figure 3: Understaffing levels induced by unsupervised break choices in two different SBS solutions of equal cost

is evaluated by computing the unsupervised break choices cost. The generation/evaluation phases are repeated until a termination condition is met (§5.5). The overall approach is summarized in Algorithm 1 while implementation details are given in §6.1.

5.2 Initialization

As discussed in §2.1, the managers tend to consider the shift costs more important than under/overstaffing costs, that partially depends on the breaks. This is somehow understandable, as the shift costs must be paid no matter what, while the extra costs depend on events that may or may not occur or that may be less damaging than expected. For instance, if the agents decide to have rest breaks only in overstaffed slots, that is, without creating extra understaffing or overstaffing with respect to the situation with no breaks, rest breaks are not even perceived as disruptive of the work activity by the managers. Thus, managers can grant some autonomy to agents as long as shift costs do not exceed

too much the amount they would pay for shifts in the single decision maker case, where they assign both shifts and breaks to agents. Of course, this cost corresponds to the shifts cost $\sum_{i \in I} c_{j(i)} x_i^*$ of an optimal solution $\mathbf{x}^*$ of the SBS problem 2.1. Hence, the initialization of the MSS algorithm consists of solving SBS-IP to determine this ideal cost, that we denote by W^1.

5.3 The generation step

At each iteration h, the generation step of the MSS heuristic is used to produce a set of candidate shift allocations $Y^h = \{\mathbf{y}^{h1}, \ldots, \mathbf{y}^{hK}\}$, whose cost is at least W^h. This is obtained by selecting the best K solution of GEN^h-IP below, corresponding to SBS-IP plus the constraint (14) that imposes a bound on the cost of the agents.

$$
\begin{aligned}
\text{GEN}^h\text{-IP} \quad & \min \sum_{i \in I} c_{j(i)} x_i + \sum_{t \in T} (\phi_t u_t + \psi_t o_t) \\
& \sum_{i \in W(t)} x_i + u_t - o_t = b_t && t \in T \\
& u_t \leq U_t && t \in T \\
& o_t \leq O_t && t \in T \\
& \sum_{i \in I} c_{j(i)} x_i \geq W^h && (14) \\
& \mathbf{x} \in \mathbb{Z}_+^{|I|}, \mathbf{u}, \mathbf{o} \in \mathbb{Z}_+^{|T|}
\end{aligned}
$$

In principle, one can explore the solution space of GEN^h-IP by running a branch-and-bound search without stopping the algorithm when the optimality has been certified and collecting the solutions with the same optimal value. Then, the set Y^h contains unique solutions $\mathbf{y}^{hk}$ obtained as

$$
y_j^{hk} = \sum_{i \in I_j} x_i^{hk} \quad j \in J
$$

where $\mathbf{x}^{hk}$ belongs to the k-best solution found solving GEN^h-IP, and discarding possible multiple copies of vectors $\mathbf{y}$.

In the first iteration ($h = 1$), we look for shift assignments with cost of the agents at least equal to the one of the optimal SBS-IP solution. At iteration $h > 1$, we compute shift assignments accepting an increase of the agent costs to obtain a reduction of unsupervised break choices cost (i.e., decreasing the value $\lambda(\mathbf{y})$). Then, W^{h+1} is set to $W^{h+1} = \max\{W^h + \texttt{step}, \mathbf{c}^T \mathbf{y}^{hk}\}$ where `step` is a suitable positive number.

Note that, in principle, one could compute candidate shifts also by solving SP-IP with constraint (14) enforced, getting an easier model to solve than GEN^h-IP. However, this would produce many shifts that cannot be completed by suitable break assignments satisfying over/understaffing bound. On the contrary, GEN^h-IP generates a pool of fully feasible solutions y^{hk}.

5.4 The evaluation step

The evaluation step associates the unsupervised break choices cost λ^{hk} with each $y^{hk} \in Y^h$, by selecting the corresponding worst case realization $\mathbf{x}^{hk}$ in $S(\mathbf{y}^{hk})$. This is done by solving the ABA($\mathbf{y}$)-MILP, i.e., the *adversarial follower problem* of model DBSP-BL for $\mathbf{y}$ fixed to $\mathbf{y}^{hk}$, that corresponds to the IP reported below.

$$
\begin{aligned}
\lambda^{hk} = \max & \sum_{t \in T} (\psi_t u_t + \phi_t o_t) \\
& \sum_{i \in W(t)} x_i^{hk} + u_t - o_t = b_t && t \in T \\
& \sum_{i \in I_j} x_i^{hk} = y_j^{hk} && j \in J \\
& o_t \leq U_t \alpha_t && t \in T \\
& u_t \leq O_t (1 - \alpha_t) && t \in T \\
& \mathbf{x}^{hk} \in \mathbb{Z}_+^{|I|}, \mathbf{o}, \mathbf{u} \in \mathbb{R}_+^{|T|}, \boldsymbol{\alpha} \in \{0,1\}^{|T|}
\end{aligned}
$$

One can use different strategies to implement the evaluation step efficiently. The first consists in exploiting the modern computational infrastructures by solving concurrently the K instances of each evaluation step. The second concerns the termination of the branch-and-bound search when the value of the incumbent λ^{hk} exceeds the value $\lambda^{\max} = \max_{j<h}\{\lambda^{jk}\}$ found in the previous iterations. In the experiments we use the latter approach to demonstrate the usability of the algorithm in standard computational environments.

5.5 Termination of the algorithm

Depending on the application and on manager needs, several termination conditions can be defined. Typical ones concern the quality of the solution, a limit on the number of iterations and conditions on the cost of agents. Usually, the managers figure out a maximum cost that they are willing to pay for the shifts as a percentage increase β with respect to the ideal cost W^1 computed in the preprocessing stage, that is, $W^{\max} = (1+\beta)W^1$. This is actually the termination criterion used in Algorithm 1.

Algorithm 1: Multiple Solutions Search Heuristic

Data: A problem instance, integers K, W^1, $W^{\max}$, `maxIter`
Result: $\mathbf{y}$, number of agents per each workshift j in J
begin
  `bestObj` $\leftarrow +\infty$;
  `step`$\leftarrow (W^{\max} - W^1)/$`maxIter`;
  $\mathbf{y} \leftarrow \emptyset$;
  `h` $\leftarrow 1$;
  while $W^h < W^{\max}$ **do**
    Generate a pool $Y^h = \{y^{h1}, \ldots, y^{hK}\}$ of unique shift assignments with agents cost $\geq W^h$;
    for $\mathbf{y}^{hk} \in Y^h$ **do**
      Compute the cost of unsupervised break choices λ^{hk}, considering the realizations $\mathbf{x}^{hk} \in S(\mathbf{y}^{hk})$;
      if $\mathbf{c}^{\mathrm{T}}\mathbf{y}^{hk} + \lambda^{hk} <$ `bestObj` **then**
        `bestObj` $\leftarrow \mathbf{c}^{\mathrm{T}}\mathbf{y}^{hk} + \lambda^{hk}$;
        `agentsCost` $\leftarrow \mathbf{c}^{\mathrm{T}}\mathbf{y}^{hk}$
        $\mathbf{y} \leftarrow \mathbf{y}^{hk}$;
      end
    end
    $W^{h+1} \leftarrow \max\{W^h +$ `step`, `agentsCost`$\}$;
    `h` $\leftarrow$ `h` $+ 1$;
  end
end
return $\mathbf{y}$, `bestObj`;

6 Experimental findings

6.1 Implementation details

The initialization step, along with the two steps in the main loop of Algorithm 1 (generation and evaluation steps), require to solve three different ILP models (SBS-IP, GEN-IP and ABA-IP($\mathbf{y}$)). In the experiments we used the commercial ILP solver `Gurobi`. Details about the machine along with the non-default list of `Gurobi` parameters are reported in Table 1

Machine: AMD EPYC 7282 16-Core Processor, 32 thread clocked at 2.8 GHz, 512 GB RAM
ILP Solver: Gurobi Optimizer version 10.0.1
MSS parameters: $K = 3000$, $W^{\max} = 1.1 \times W^1$, `maxIter` $= 20$
Maximum CPU time allowed for pool generation: 120 sec
Gurobi non-default parameters: `PoolSearchMode`=2, `PoolGap`=0.01, `PoolSolutions`=K

Table 1: Machine characteristics and parameters setting

6.2 Test bed

We consider a set of 64 instances generated following the guidelines presented in [43]. First, as in the example of §5.1.1, we consider 16-hour operating day subdivided into 192 5-minute slots. Industry standards consider 15-minute slots for forecasting and evaluating staff schedules, but we use a finer temporal grain to better represent the behavior of agents. However, to conform to common practice, the staffing levels we generate have the same value for three consecutive slots. Then, as in [43], we generate staffing levels by considering four dimensions. The first dimension is the *ideal shape* of the staffing level, called *Multiple Peaks, Single Peak, Two Peaks* and *Three Peaks* shape. The second dimension is the average number of employees during operating hours (respectively 10, 20, 40, 80 employees) and, finally, the third dimension is the perturbation of the ideal staffing level shape, in the measure of $\pm 10\%, \pm 20\%, \pm 40\%$ and $\pm 80\%$. As explained in [43], the resulting 64 instances represent several common industrial environments. Figure 8 show the staffing levels for an average level of 40 agents, the four different ideal staffing shapes with the two extreme variations (perturbation 10% and 80%). Differently from [43], we do not consider small service environments (less than 10 employees). The set of possible workshifts and workpatterns (4 hours and 6 hours shifts with associated costs) are the same defined in §5.1.1.

6.3 Experiments

Two experiments are documented. The first experiment (6.3.1) evaluates the cost increase when employees are free to choose their breaks without any further restriction. In the second scenario employees can choose their breaks within a maximum allowed understaffing level. In both scenarios understaffing cost is set to 15 Euro per slot and overstaffing cost is set equal to zero.

6.3.1 Scenario 1: uncontrolled understaffing

In this scenario employees are allowed to select one of the work patterns associated with their shift without further constraints. This amounts to set both understaffing U_t and overstaffing O_t bounds to $\max_{t \in T}\{b_t\}$. This corresponds to a severe scenario for the manager, and the key issue here is to evaluate the price of protecting against discretionary breaks. Computational results are reported in Table 2 and 3 and summarized in Figure 5 for some instances with 80 agents.
Each table is referred to an ideal staffing shape and contains the following information: the columns "Avg. level" and "Var. (%)" report the average number of agents and the perturbation value. Under the header **SBS-IP** are reported the value of the optimal solution to SBS-IP ("Opt."), the corrresponding cost of the agents $c^{\mathrm{T}}\mathbf{y}^*$ ("Agents cost"), the CPU time (in seconds), and the overall unsupervised break choices cost $c^{\mathrm{T}}\mathbf{y}^* + \lambda(\mathbf{y}^*)$ ("Unsup. cost"). Under the header **Multiple Search Solution Heuristic** algorithm's statistics are presented.

Iteration #1 columns show the cost ("Unsup. cost") of the best realization found in the first iteration of the algorithm (the one with $W^1 = c^{\mathrm{T}}\mathbf{y}^*$), the CPU time needed to generate and evaluate the pool of solutions (whose number is in column "# sol."). Finally, under **Best solution** columns, one can find the value of the best solution $\bar{\mathbf{y}}$ to DBSP-BLP found by the MSS heuristic ("Unsup. cost"), the corresponding agents cost ("Agents cost"), the total CPU time and number of evaluated solutions.

						Multiple Search Solutions Heuristic						
			SBS-IP			Iteration #1			Best solution			
Avg. Lev.	Var. (%)	Opt.	Agents cost	Time (sec)	Unsup. cost	Unsup. cost	Time (sec)	#sol.	Unsup. cost	Agents cost	Time (sec)	#sol.
10	10	1990	1990	0.3	3295	3130	23.6	393	2805	2160	724.8	14190
	20	2080	2080	0.3	3445	3160	34.5	769	2845	2230	682.8	16628
	40	2310	2310	0.2	3540	3435	35.7	1147	3045	2490	597.6	17718
	80	2760	2760	0.2	4245	4095	23.5	898	3650	2990	614.9	22730
20	10	4125	4110	0.7	6660	6480	64.0	849	5745	4470	1059.9	18847
	20	4260	4260	0.2	7095	6750	34.7	772	5955	4620	1138.6	23570
	40	4660	4660	0.2	7450	7150	40.4	1110	6380	5060	1026.0	26929
	80	5610	5610	0.1	9390	8400	44.4	1552	7305	6090	815.3	27402
40	10	8340	8340	0.3	13680	13230	22.3	343	11695	9040	1385.4	24647
	20	8610	8610	0.3	13875	13410	41.5	663	11990	9350	1383.4	26161
	40	9410	9410	0.2	15305	14495	39.6	986	12465	10170	1068.4	26311
	80	11310	11310	0.3	18765	16695	36.0	1200	15575	12320	1101.5	35346
80	10	16770	16770	0.3	27525	26535	71.2	1116	23880	17850	1401.0	21432
	20	17310	17310	0.2	27840	27465	72.3	1493	24020	18860	1771.3	33037
	40	18910	18910	0.2	30070	29440	63.9	1577	26120	19850	740.7	17562
	80	22810	22810	0.1	36475	33355	47.2	1451	29595	24750	1232.2	31689

(a) Multiple peaks staffing shape

						Multiple Search Solutions Heuristic						
			SBS-IP			Iteration #1			Best solution			
Avg. Lev.	Var. (%)	Opt.	Agents cost	Time (sec)	Unsup. cost	Unsup. cost	Time (sec)	#sol.	Unsup. cost	Agents cost	Time (sec)	#sol.
10	10	2200	2200	1.2	3610	3430	14.3	330	3095	2360	413.9	9655
	20	2150	2150	1.6	3425	3290	16.0	277	3050	2300	569.2	12717
	40	2045	2030	0.7	3425	3350	29.2	319	2970	2220	654.8	11697
	80	1920	1920	1.1	3465	3375	18.8	22	2790	2100	779.1	9146
20	10	4510	4480	0.8	6955	6805	59.6	951	6305	4790	865.1	20214
	20	4395	4380	0.8	6840	6720	34.4	651	6210	4680	861.6	17668
	40	4190	4190	1.6	6830	6725	40.7	311	6015	4530	1163.1	17482
	80	3930	3930	5.6	6840	6825	26.1	8	5855	4280	1891.7	15816
40	10	9135	9000	0.3	14325	13920	47.7	899	12895	9730	1076.6	20523
	20	8915	8810	0.1	14300	13490	82.0	1537	12805	9610	1596.0	28316
	40	8500	8500	1.0	13525	13435	74.2	693	12020	9260	2168.0	28773
	80	7980	7980	8.1	14190	13695	240.2	290	11955	8610	3133.4	21021
80	10	18405	18300	0.3	29550	27390	85.6	1973	25940	19760	1376.7	27981
	20	17970	17970	1.1	28110	27300	64.4	1047	24920	19400	1737.7	30101
	40	17195	17150	1.3	27620	27035	140.0	1332	24550	18430	2148.6	28856
	80	16050	16050	2.8	28545	27825	146.5	29	24150	17490	4842.8	30807

(b) Single peak staffing shape

Table 2: Computational results for multiple peaks and single peak shapes in Scenario 6.3.1

						Multiple Search Solutions Heuristic						
			SBS-IP			Iteration #1			Best solution			
Avg. Lev.	Var. (%)	Opt.	Agents cost	Time (sec)	Unsup. cost	Unsup. cost	Time (sec)	#sol.	Unsup. cost	Agents cost	Time (sec)	#sol.
10	10	2170	2140	1.7	3430	3295	38.6	887	3060	2310	518.7	11468
	20	2115	2100	0.6	3390	3225	43.5	730	2975	2270	526.1	10247
	40	2055	2040	1.1	3315	3195	63.7	1012	2865	2190	657.8	11063
	80	2020	2020	1.6	3220	3175	20.5	165	2800	2140	503.8	6476
20	10	4465	4420	0.5	7105	6775	37.1	808	6320	4820	1045.6	19366
	20	4355	4340	0.6	6905	6635	54.3	795	6105	4680	943.6	18933
	40	4180	4180	0.4	6835	6715	47.7	486	5885	4520	1468.4	21099
	80	4100	4100	4.9	6380	6365	34.2	140	5745	4380	1253.3	11481
40	10	9085	9040	0.4	14125	13675	46.1	928	12580	9760	1242.7	27258
	20	8830	8830	0.5	13390	13225	54.8	906	12235	9490	1268.2	24022
	40	8470	8470	0.3	13945	13480	88.2	878	11855	9230	2029.8	29526
	80	8280	8280	3.5	13110	12960	23.1	68	11650	8980	2401.3	20524
80	10	18300	17970	0.2	28440	27360	77.9	1364	25880	19310	1504.6	26962
	20	17800	17530	0.3	28225	27130	107.4	1633	24810	18840	1848.2	30319
	40	17085	17070	0.3	28350	27465	106.2	992	24885	18600	2733.2	36110
	80	16630	16630	4.8	26110	26020	31.4	114	23625	17460	2026.9	11797

(a) Two peaks staffing shape

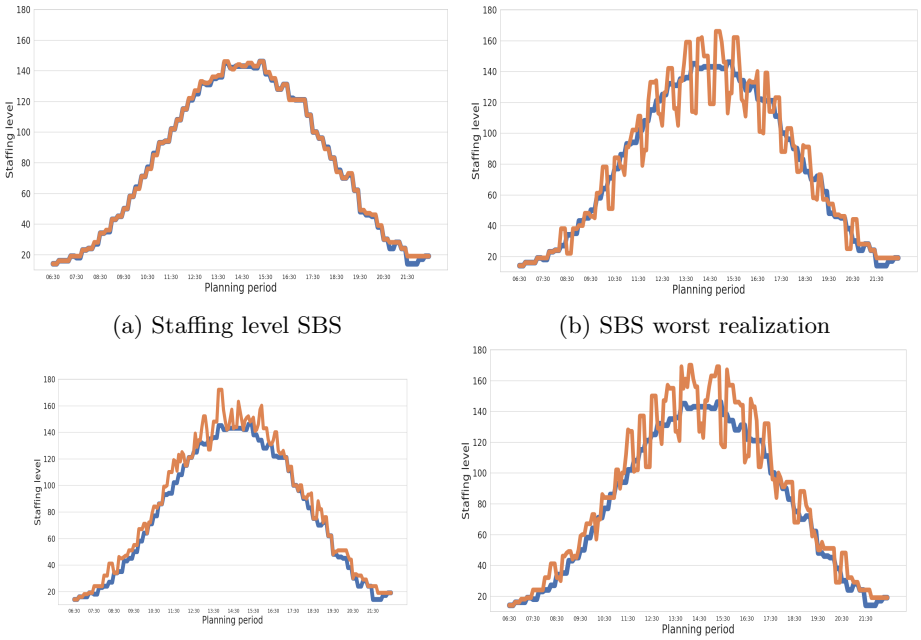
						Multiple Search Solutions Heuristic						
			SBS-IP			Iteration #1			Best solution			
Avg. Lev.	Var. (%)	Opt.	Agents cost	Time (sec)	Unsup. cost	Unsup. cost	Time (sec)	#sol.	Unsup. cost	Agents cost	Time (sec)	#sol.
10	10	2200	2170	0.5	3355	3340	20.3	506	2995	2350	494.4	12914
	20	2165	2150	0.9	3350	3275	19.1	431	2965	2320	538.8	14049
	40	2140	2140	0.1	3250	3145	44.2	815	2950	2320	658.0	18545
	80	2590	2590	0.1	3850	3745	14.7	266	3305	2810	603.3	15449
20	10	4525	4450	0.6	7060	6700	41.4	1032	6115	4780	692.8	16749
	20	4490	4430	0.2	6665	6440	53.1	1104	5985	4830	1084.7	25762
	40	4445	4430	0.2	6770	6470	60.9	1298	6090	4830	1003.7	24613
	80	5260	5260	0.3	7645	7195	27.4	536	6850	5680	807.6	18066
40	10	9230	9110	0.6	13910	13400	42.1	987	12485	9920	1394.5	29114
	20	9145	9040	0.2	13450	13360	55.9	1229	12135	9810	1249.7	25488
	40	8995	8980	0.2	13480	13015	50.8	948	12440	9560	914.8	20652
	80	10610	10610	0.2	15500	14735	21.5	195	13895	11300	929.0	18712
80	10	18630	18300	0.8	28935	27420	85.3	1324	25190	19940	1661.7	30221
	20	18415	18190	0.2	27025	26290	66.9	1533	24685	19810	1380.2	26624
	40	18140	18140	0.2	27275	26030	64.1	1439	24965	19670	1395.3	30033
	80	21280	21280	0.5	31405	30955	33.5	459	28200	23190	1172.2	23716

(b) Three peaks staffing shape

Table 3: Computational results for two peaks and three peak shapes in Scenario 6.3.1

Results in Tables 2 and 3 show that SBP-IP optimal solutions rarely contains under/overstaffing costs as the optimal cost mostly coincides with the agents cost. As in this scenario overstaffing has zero cost, this implies that SBS-IP solutions practically do not present understaffing. Under this respect, this is the most favourable scenario for SBS-IP, as one expects to have large room to accommodate breaks. Nevertheless, SBS-IP solutions are not robust to discre-

tionary breaks as the worst realizations of unsupervised break choices for such solutions present large understaffing costs (i.e., induces LoS degradation) for all of the tested instances. However, this cost can be significantly mitigated by applying MSS and this is evident since the very first iteration of the algorithm. Precisely, in Iteration #1, agents cost is the same of SBS-IP and planning the work shifts according to the best realization found by exploring the cost-equivalent solutions significantly reduces the cost of unsupervised break choices. Then, the best solution found by MSS reduces the latter by about 50% in all instances.



(a) Staffing level SBS

(b) SBS worst realization

(c) Staffing level MSS

(d) MSS worst realization

Figure 4: Staffing level of SBS and MSS solutions (single peak shape, 80 agents, 80% pert.) and the corresponding worst realizations in Scenario 6.3.1

In Figure 4 the solutions for the single peak instance of 80 agents with 80% perturbation are compared. The first row shows the staffing level determined by the SBS-IP optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ (left chart) and the corresponding staffing level for the worst realization of unsupervised breaks $\lambda(\mathbf{y}^*) = 28545$ (right chart). The second row contains the staffing level determined by the solution $\bar{\mathbf{y}}$ found by the MSS algorithm. The left chart shows the staffing level of the best possible supervised break assignment in $\bar{\mathbf{y}}$, while the right chart shows the staffing level of the worst realization of unsupervised breaks (i.e, the one of cost $\lambda(\bar{\mathbf{y}}) = 24150$). Clearly, the MSS solution $\bar{\mathbf{y}}$ exhibits some overstaffing when breaks are assigned by the manager w.r.t. SBS-IP, but it is clearly more robust than the SBS-IP solution when unsupervised breaks are allowed, resulting in a

smaller overall cost (agents cost plus unsupervised break choices cost).
Figure 5, which refers to the largest instances considered (80 agents on average) and extreme range perturbations (10% and 80%), shows the trends of cost reduction. The top line in graphs represents the incumbent value of the MSS heuristic w.r.t. the increase of the agents cost (bottom line). The bottom labels on the x-axis are the number of MSS iterations, while labels on the top are the percentage agents cost increase w.r.t. the SBS-IP agents cost. Therefore, the gap between lower and upper lines represents the cost of the worst realization which decreases as the MSS iterations progress. At each iteration the manager can estimate the cost to pay to reduce understaffing due to unsupervised break choices and can select the best tradeoff.
All instances in this scenario show similar trends and the overall CPU time is reasonably low as only in a few cases it exceeds one hour (while the first iteration is executed on average under a minute). In conclusion, in this scenario, MSS provides good solutions for a (non-trivial) cost/benefit analysis of alternative staff schedules and it is also computationally efficient.

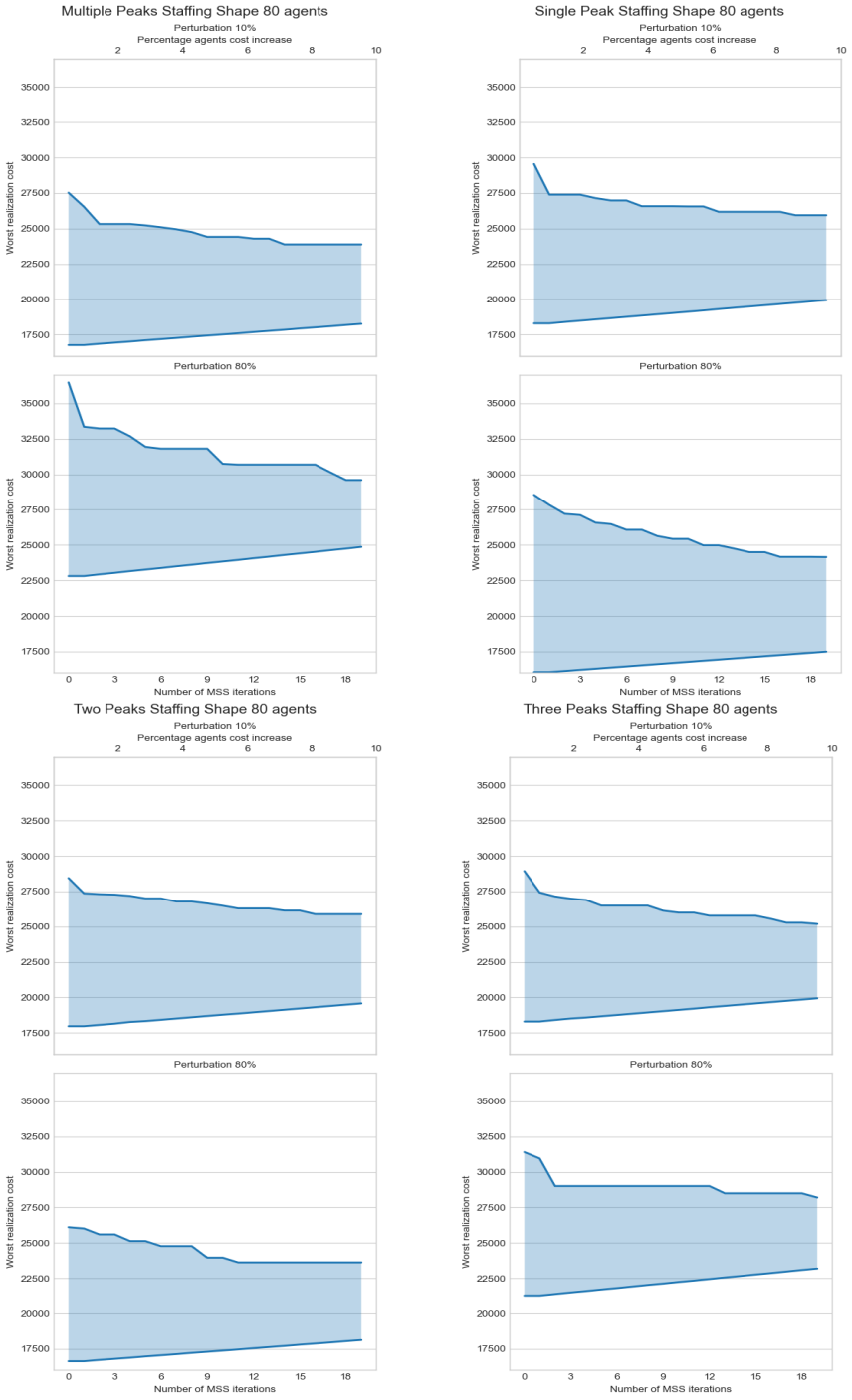


Figure 5: Unsupervised break cost trends (80 agents, pert. 10% and 80%)

6.3.2 Scenario 2: controlled understaffing

This scenario differs from scenario 6.3.1 since a maximum allowed understaffing U_t is imposed. Precisely, U_t is set equal to 10% of the average staffing level, for all $t \in \mathcal{T}$.

						Multiple Search Solutions Heuristic						
			SBS-IP			Iteration #1			Best solution			
Avg. Lev.	Var. (%)	Opt.	Agents cost	Time (sec)	Unsup. cost	Unsup. cost	Time (sec)	#sol.	Unsup. cost	Agents cost	Time (sec)	#sol.
10	10	1990	1990	0.3	2770	2665	15.6	249	2495	2120	468.3	8787
	20	2080	2080	0.3	2725	2680	43.2	1095	2520	2130	321.4	7457
	40	2310	2310	0.6	2850	2745	33.0	987	2695	2380	236.3	6930
	80	2760	2760	0.2	3135	3105	27.2	993	3105	2760	27.5	993
20	10	4125	4110	1.3	5655	5505	32.1	580	5160	4470	1342.5	20216
	20	4260	4260	0.2	5805	5580	61.0	1088	5205	4470	781.2	13980
	40	4660	4660	0.2	6070	5590	47.5	1254	5510	4730	209.6	5569
	80	5610	5610	0.1	6495	6435	42.8	1257	6315	5670	119.8	3559
40	10	8340	8340	0.3	11565	11250	43.2	533	10510	8800	1061.8	15123
	20	8610	8610	0.3	11700	11310	88.1	792	10620	9090	1031.3	16957
	40	9410	9410	0.1	12215	11450	60.2	1317	11300	10070	1133.1	26912
	80	11310	11310	0.2	13245	12780	43.2	1228	12780	11310	43.3	1228
80	10	16770	16770	0.3	22935	22500	127.4	896	21090	17940	1930.3	26174
	20	17310	17310	0.3	24210	23190	99.1	1445	21450	18600	1690.2	28016
	40	18910	18910	0.2	24430	23215	52.1	1019	22730	19100	207.3	4305
	80	22810	22810	0.2	26380	25705	36.9	1148	25365	23160	238.4	6754

(a) Multiple peaks staffing shape

						Multiple Search Solutions Heuristic						
			SBS-IP			Iteration #1			Best solution			
Avg. Lev.	Var. (%)	Opt.	Agents cost	Time (sec)	Unsup. cost	Unsup. cost	Time (sec)	#sol.	Unsup. cost	Agents cost	Time (sec)	#sol.
10	10	2200	2200	1.0	2770	2740	11.4	235	2605	2380	503.7	11803
	20	2150	2150	0.5	2780	2735	14.5	220	2615	2210	204.5	4338
	40	2045	2030	0.4	2840	2810	22.6	288	2585	2180	510.8	8542
	80	1920	1920	2.7	2985	2895	18.2	23	2575	2110	1037.9	11451
20	10	4510	4510	0.5	5800	5515	43.5	790	5305	4690	520.8	11111
	20	4395	4380	0.9	5775	5550	51.2	1152	5365	4720	926.8	19566
	40	4190	4190	1.9	5930	5840	31.3	160	5360	4550	1470.8	21920
	80	3930	3930	2.5	6015	6000	133.5	8	5400	4200	3175.8	9671
40	10	9135	9060	0.3	11475	11250	73.9	1618	10755	9420	639.3	12416
	20	8915	8840	0.2	11480	11210	67.0	1360	10900	9550	1543.2	28300
	40	8500	8500	1.1	11755	11635	207.5	637	10835	9260	2906.3	32075
	80	7980	7980	3.2	12375	12150	470.5	207	10880	8690	6756.3	23037
80	10	18405	18210	0.5	23520	22425	70.5	1256	21565	19030	867.8	15107
	20	17970	17880	1.1	23820	22650	108.4	1597	21935	19220	1816.3	29122
	40	17195	17180	1.6	23630	23315	426.2	1539	21745	17950	2302.6	17475
	80	16050	16050	15.0	24570	24570	419.0	144	21970	17410	9350.9	29648

(b) Single peak staffing shape

Table 4: Computational results for multiple peaks and single peak shapes in Scenario 6.3.2

						Multiple Search Solutions Heuristic						
			SBS-IP			Iteration #1			Best solution			
Avg. Lev.	Var. (%)	Opt.	Agents cost	Time (sec)	Unsup. cost	Unsup. cost	Time (sec)	#sol.	Unsup. cost	Agents cost	Time (sec)	#sol.
10	10	2170	2170	4.7	2755	2665	24.0	400	2580	2280	473.9	8447
	20	2115	2100	0.6	2850	2700	38.4	608	2560	2290	775.0	12914
	40	2055	2040	0.6	2760	2715	124.9	1149	2520	2220	1350.9	15063
	80	2020	2020	1.5	2860	2815	30.6	263	2520	2160	876.2	10124
20	10	4465	4450	0.8	5815	5530	54.0	1192	5335	4630	425.8	7722
	20	4354	4340	0.6	5690	5555	43.3	710	5350	4510	564.9	9076
	40	4180	4180	0.8	5710	5695	204.0	925	5275	4540	2348.1	23496
	80	4100	4100	4.4	5750	5660	48.4	244	5175	4440	2315.3	18207
40	10	9085	9040	0.8	11575	11050	88.5	888	10750	9220	443.4	7402
	20	8830	8830	0.7	11470	11095	87.1	1275	10785	9630	1783.7	30021
	40	8470	8470	0.6	11710	11425	267.7	882	10705	9100	2848.1	24748
	80	8280	8280	3.0	11580	11460	39.9	43	10705	8860	3418.5	15800
80	10	18300	18240	0.5	23385	22065	77.3	1411	21955	18700	554.0	10642
	20	17800	17740	0.2	23410	22135	92.4	1476	21655	19060	1853.9	29856
	40	17085	17070	0.8	23805	23250	236.6	1095	21785	18350	3326.4	30856
	80	16630	16630	3.2	23290	23125	23.1	34	21455	17960	5284.0	21656

(a) Two peaks staffing shape

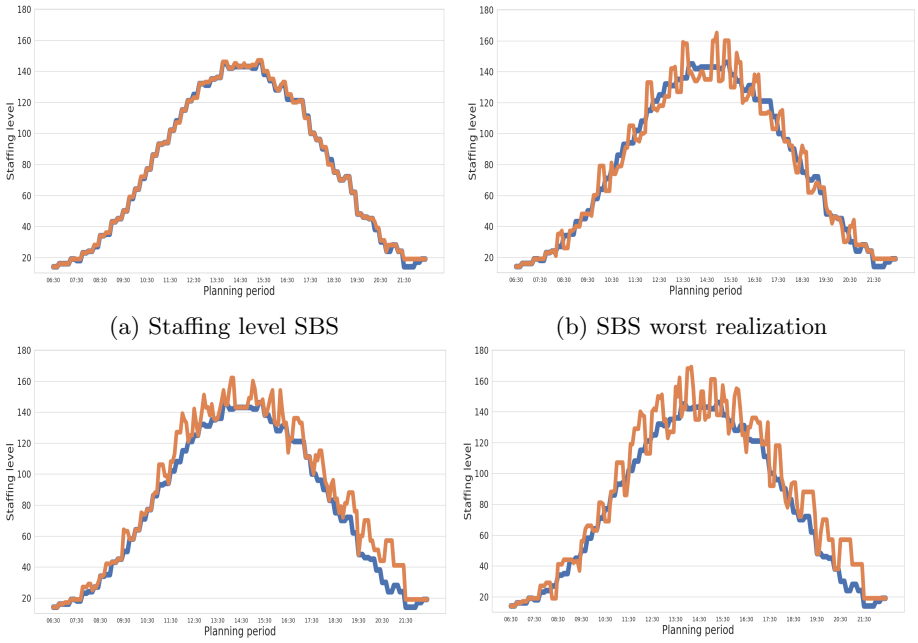
						Multiple Search Solutions Heuristic						
			SBS-IP			Iteration #1			Best solution			
Avg. Lev.	Var. (%)	Opt.	Agents cost	Time (sec)	Unsup. cost	Unsup. cost	Time (sec)	#sol.	Unsup. cost	Agents cost	Time (sec)	#sol.
10	10	2200	2170	0.3	2845	2755	20.1	351	2590	2380	828.8	16708
	20	2165	2150	1.2	2780	2690	38.2	739	2550	2220	261.9	4499
	40	2140	2140	0.2	2755	2695	14.1	285	2570	2270	616.1	13382
	80	2590	2590	0.2	3325	3160	21.3	419	3095	2660	241.3	4523
20	10	4525	4480	0.3	5755	5440	41.1	750	5310	4740	1776.8	28390
	20	4490	4430	0.2	5570	5375	87.2	1360	5255	4610	1159.5	17661
	40	4445	4430	0.3	5735	5525	85.5	876	5340	4560	962.6	12919
	80	5260	5260	0.4	6700	6535	22.6	288	6230	5600	2172.6	33171
40	10	9230	9110	0.2	11510	11015	64.6	1226	10755	9240	231.7	3526
	20	9145	9040	0.2	11335	11155	118.4	1610	10770	9810	1501.9	24980
	40	8995	8980	0.5	11350	11155	120.8	1263	10895	9380	1120.1	14998
	80	10610	10610	0.2	13310	13055	26.4	404	12685	11410	1204.2	20542
80	10	18635	18410	0.2	23315	22340	92.2	1498	21630	18690	380.9	5812
	20	18415	18220	0.2	22540	22105	69.7	744	21570	19410	1421.2	20568
	40	18140	18140	0.4	22880	22385	139.3	1369	21830	19040	1265.7	17709
	80	21280	21280	0.3	27025	26950	29.0	330	25835	21920	615.8	7134

(b) Three peaks staffing shape

Table 5: Computational results for two peaks and three peaks shapes in Scenario 6.3.2

From Tables 4 and 5 one can observe that the optimal solutions to SBS-IP coincide with the solutions of the previous scenario (with the exception of one instance). This was somewhat expected, because understaffing was not present in the SBS-IP solutions of the previous scenario and, therefore, the bound U_t is not binding. However, the restriction on the maximum allowed understaffing U_t becomes active in evaluating $\lambda(\mathbf{y})$, reducing the cost associated with the worst

realization. This can be observed also by comparing profiles in Figure 4b and 6b where the latter shows considerably less understaffing.



(a) Staffing level SBS

(b) SBS worst realization

(c) Staffing level MSS

(d) MSS worst realization

Figure 6: Staffing level of SBS and MSS solutions (single peak shape, 80 agents, 80% pert.) and the corresponding worst realizations in Scenario 6.3.2

From the employees' perspective, the bound on maximum understaffing restricts complete discretion in break choices but still provides more freedom than the supervised assignments of rest times.
The MSS algorithm performs similarly to scenario 6.3.1, being able to reduce considerably the worst case cost after a few iterations (see also Figure 7) while keeping the computational burden to an acceptable level, as in Scenario 6.3.1.

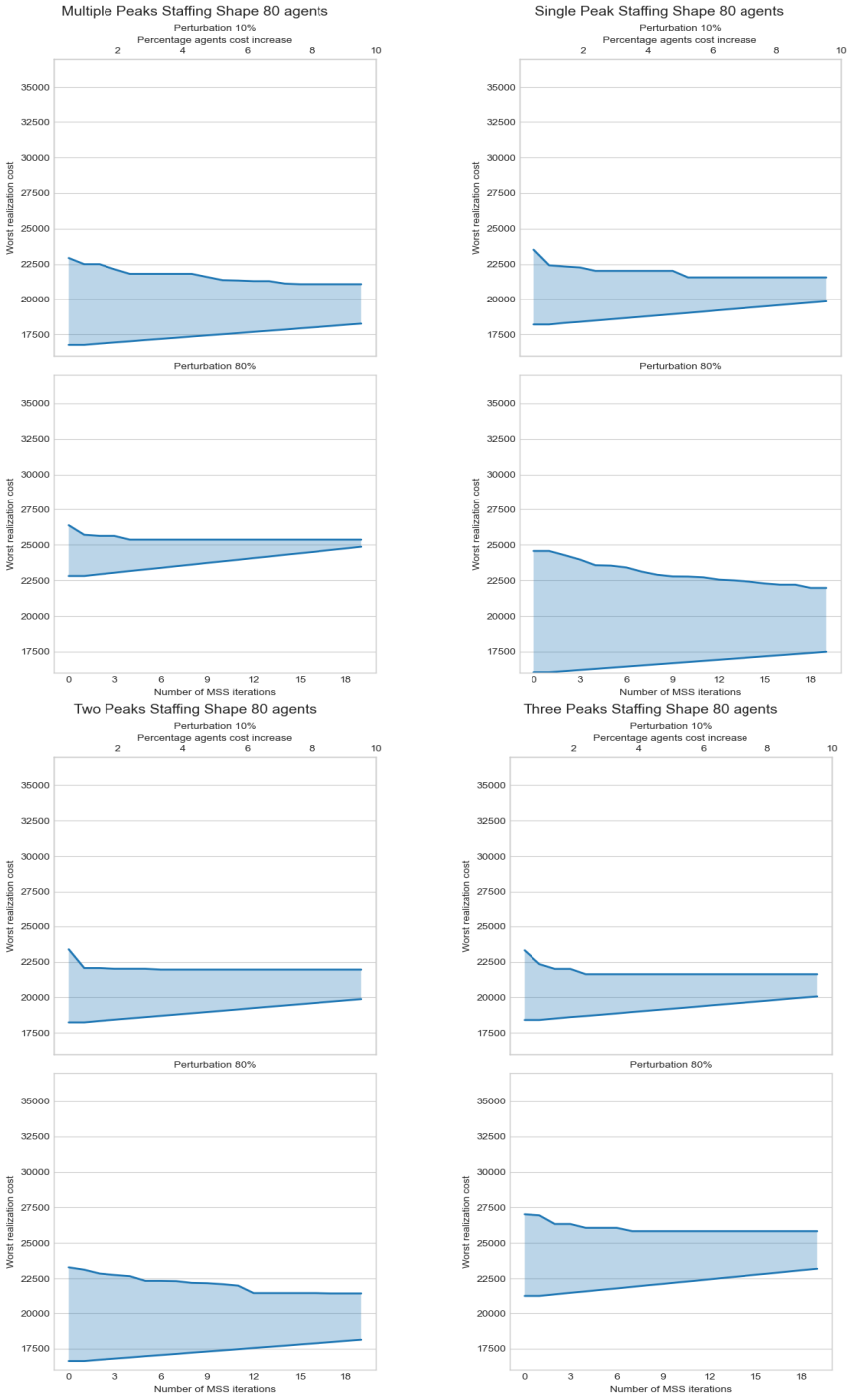


Figure 7: Unsupervised break cost with bounded understaffing level (80 agents, pert. 10% and 80%)

7 Conclusions

We have shown that robust optimization and bilevel programming provide a modelling framework able to bridge the gap between scheduling workbreaks in advance and (re-)adjusting them in real-time. Furthermore, it also allows to enrich the model by letting employees free to decide when to take breaks, a need that is becoming increasingly important in all industrial settings, where the pressure on workers is now very high. Our results show that the proposed model achieves an excellent level of conservatism, as the price of robustness with respect to employees discretionary turns out to be quite reasonable in real-world scenarios. Furthermore, by controlling the maximum level of understaffing, our tool is compliant with a widespread practice and allows for a fine chasing of the desired trade-off between cost, LoS and employees discretionary. The proposed heuristic has no criticality with respect to an industrial implementation and can represent a powerful support for workforce managers.
Finally, this framework not only has an operational application, but it can be used at the strategic level to design workforce management regulatory frameworks.

References

[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice hall, 1993.

[2] Z. Aksin, M. Armony, and V. Mehrotra. The modern call center: A multi-disciplinary perspective on operations management research. *Production and Operations Management*, 16(6):665–688, 2007.

[3] T. Aykin. Optimal shift scheduling with multiple break windows. *Management Science*, 42(4):591–602, 1996.

[4] T. Aykin. A comparative evaluation of modeling approaches to the labor shift scheduling problem. *European Journal of Operational Research*, 125(2):381–397, 2000.

[5] S. E. Bechtold and L. W. Jacobs. Implicit modeling of flexible break assignments in optimal shift scheduling. *Management Science*, 36(11):1339–1351, 1990.

[6] A. Beer, J. Gärtner, N. Musliu, W. Schafhauser, and W. Slany. Scheduling breaks in shift plans for call centers. In *Proc. of the 7th International Conference on the Practice and Theory of Automated Timetabling, Université de Montréal*, 2008.

[7] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton Series in Applied Mathematics, 2009.

[8] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004.

[9] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[10] G. Blasche, A. Arlinghaus, and R. Crevenna. The impact of rest breaks on subjective fatigue in physicians of the general hospital of vienna. *Wien Klin Wochenschr*, 134(3-4):156–161, 2022.

[11] G. Blasche, S. Pasalic, V.M. Bauböck, D.Haluza, and R. Schoberberger. Effects of rest-break intention on rest-break frequency and work-related fatigue. *Human Factors*, 59(2):289–298, 2017.

[12] J. Cabrita and C. Cerf. Rest breaks from work: Overview of regulations, research and practice. `https://eurofound.link/ef19018`. Accessed: 24 Sep 2023.

[13] A. Chavaillaz, A. Schwaninger, S. Michel, and J. Sauer. Work design for airport security officers: Effects of rest break schedules and adaptable automation. *Applied Ergonomics*, 79:66–75, 2019.

[14] G. B. Dantzig. Letter to the Editor — A comment on Edie's "Traffic Delays at Toll Booths". *Journal of the Operations Research Society of America*, 2(3):339–341, 1954.

[15] M. Defraeye and I. Van Nieuwenhuyse. Staffing and scheduling under nonstationary demand for service: A literature review. *Omega*, 58:4–25, 2016.

[16] J. Van den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester, and L. De Boeck. Personnel scheduling: A literature review. *European Journal of Operational Research*, 226(3):367–385, 2013.

[17] F. Kiermaier Ferdinand, M. Frey, and J.F. Bard. The flexible break assignment problem for large tour scheduling problems with an application to airport ground handlers. *Journal of Scheduling*, 23(2):177–209, 2020.

[18] M. Fischetti, I. Ljubic, M. Monaci, and M. Sinnl. A new general-purpose algorithm for mixed-integer bilevel linear programs. *Operations Research*, 65:1615–1637, 2017.

[19] J. Fliege and L.N L.N. Vicente. Multicriteria approach to bilevel optimization. *Journal of Optimization Theory and Applications*, 131:209–225, 2006.

[20] L . Di Gaspero, J. Gärtner, N. Musliuand A. Schaerf, W. Schafhauser, and W. Slany. Automated shift design and break scheduling. In A. Ş. Uyar, E. Özcan, and N. Urquhart, editors, *Automated Scheduling and Planning - From Theory to Practice*, volume 505 of *Studies in Computational Intelligence*, pages 109–127. Springer-Verlag, 2013.

[21] R.A. Henning, P. Jaques, G.V. Kissel, A.B. Sullivan, and S.M. Alters-Webb. Frequent short rest breaks from computer work: effects on productivity and well-being at two field sites. *Ergonomics*, 40(1):78–91, 1997. PMID: 8995049.

[22] Y. Hur, J. F. Bard, M. Frey, and F. Kiermaier. A stochastic optimization approach to shift scheduling with breaks adjustments. *Computers and Operations Research*, 107:127–139, 2019.

[23] R. Jeroslov. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical Programming*, 32:146–164, 1985.

[24] T. Kleinert, M. Labbé, I. Ljubic, and M. Schmidt. A survey on mixed-integer programming techniques in bilevel optimization. `http://www.optimization-online.org/DB_FILE/2021/01/8187.pdf`, Accessed February 2021.

[25] G. Koole and S. Li. A practice-oriented overview of call center workforce planning, 2023.

[26] F. Legillon, A. Liefooghe, and E.-G. Talbi. Cobra: A cooperative coevolutionary algorithm for bi-level optimization. In *IEEE CEC 2012 Congress on Evolutionary Computation*, 2012. DOI: 10.1109/CEC.2012.6256620.

[27] S. Liao, G. Koole, C. van Delft, and O. Jouini. Staffing a call center with uncertain non-stationary arrival rate and flexibility. *OR Spectrum*, 34(3):691–721, 2012.

[28] S. Liao, C. van Delft, and J.-P. Vial. Distributionally robust workforce scheduling in call centres with uncertain arrival rates. *Optimization Methods and Software*, 28(3):501–522, 2013.

[29] L. Lozano and J. Cole Smith. A value-function-based exact approach for the bilevel mixed-integer programming problem. *Operations Research*, 65:768–786, 2017.

[30] S. Mattia. The robust network loading problem with dynamic routing. *Computational Optimization and Applications*, 54(3):619–643, 2013.

[31] S. Mattia. Reformulations and complexity of the clique interdiction problem by graph mapping. *Discrete Applied Mathematics*, 2021. in press, DOI 10.1016/j.dam.2021.06.008.

[32] S. Mattia. The follower optimality cuts for mixed integer linear bilevel programming problems. *Soft Computing*, 27:11529–11550, 2023.

[33] S. Mattia and M. Poss. A comparison of different routing schemes for the robust network loading problem: polyhedral results and computation. *Computational Optimization and Applications*, 69:753–800, 2018.

[34] S. Mattia, F. Rossi, M. Servilio, and S. Smriglio. Staffing and scheduling flexible call centers by two-stage robust optimization. *Omega*, 72:25–37, 2017.

[35] M. Rekik, J. F. Cordeau, and F. Soumis. Implicit shift scheduling with multiple breaks and work stretch duration restrictions. *Journal of Scheduling*, 13(1):49–75, 2010.

[36] M.A. Schmidt. Valuing flexibility: A model of discretionary rest breaks. `https://ma-schmidt.com/papers/breaks_JMP_schmidt.pdf`, 2019. Accessed 10 Oct 2023.

[37] M. Segal. The operator-scheduling problem: A network-flow approach. *Operations Research*, 22(4):808–823, 1974.

[38] J.C. Smith and Y. Song. A survey of network interdiction models and algorithms. *European Journal of Operations Research*, 283:797–811, 2020.

[39] El-G. Talbi, editor. *Metaheuristics for Bi-level Optimization*. Studies in Computational Intelligence. Springer Berlin, Heidelberg, 2013.

[40] Y. Tang, J.-P.P. Richard, and J. Cole Smith. A class of algorithms for mixed-integer bilevel min-max optimization. *Journal of Global Optimization*, 66:225–262, 2016.

[41] G. M. Thompson. Improved implicit optimal modeling of the labor shift scheduling problem. *Management Science*, 41(4):595–607, 1995.

[42] G. M. Thompson. Controlling workforce schedules in real time. In *Labor Scheduling, Part 4*. Cornell University, 1999.

[43] G. M. Thompson and M. E. Pullman. Scheduling workforce relief breaks in advance versus in real-time. *European Journal of Operational Research*, 181:139–155, 2007.

[44] J.P. Trougakos, I. Hideg, B.H. Cheng, and D.J. Beal. Lunch breaks unpacked: The role of autonomy as a moderator of recovery during lunch. *Academy of Management Journal*, 57(2):405–421, 2014.

[45] U. Wen and Y. Yang. Algorithms for solving the mixed integer two-level linear programming problem. *Computers & Operations Research*, 17:133–142, 1990.

[46] M. Widl and N. Musliu. The break scheduling problem: complexity results and practical algorithms. *Memetic Comp.*, 6:97—-112, 2014.

[47] S. Xu and N.G. Hall. Fatigue, personnel scheduling and operations: Review and research opportunities. *European Journal of Operational Research*, 295(3):807–822, 2021.

A Supplementary material

A.1 Example of staffing levels

]

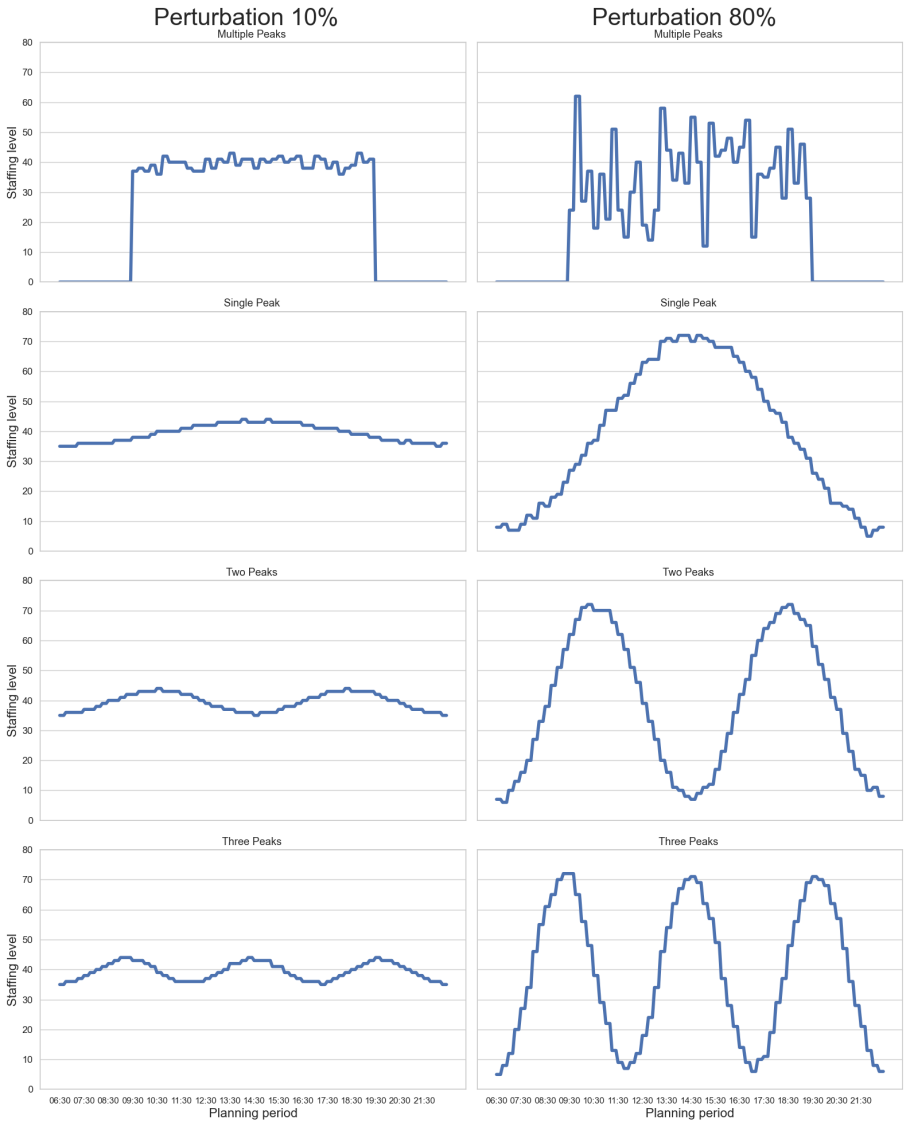


Figure 8: Staffing levels for 40 agents (on average)

Conflicts of Interest Statement

Manuscript title: Robust shift scheduling with discretionary rest breaks

The authors whose names are listed immediately below certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

Author names: Sara Mattia, Fabrizio Rossi, Stefano Smriglio

L'Aquila, 6/12/2023

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: