Anoshkina, Yulia; Meisel, Frank

Article

Interday routing and scheduling of multi-skilled teams with consistency consideration and intraday rescheduling

EURO Journal on Transportation and Logistics (EJTL)

Provided in Cooperation with:
Association of European Operational Research Societies (EURO), Fribourg

Suggested Citation: Anoshkina, Yulia; Meisel, Frank (2020) : Interday routing and scheduling of multi-skilled teams with consistency consideration and intraday rescheduling, EURO Journal on Transportation and Logistics (EJTL), ISSN 2192-4384, Elsevier, Amsterdam, Vol. 9, Iss. 3, pp. 1-18, https://doi.org/10.1016/j.ejtl.2020.100012

EURO Journal on Transportation and Logistics 9 (2020) 100012

Contents lists available at ScienceDirect

EURO Journal on Transportation and Logistics

journal homepage: www.journals.elsevier.com/euro-journal-on-transportation-and-logistics

Interday routing and scheduling of multi-skilled teams with consistency consideration and intraday rescheduling

Yulia Anoshkina *, Frank Meisel

School of Economics and Business, Christian-Albrechts-University of Kiel, Germany

ARTICLE INFO

Keywords:
Multi-skill workforce scheduling
Interday planning
Intraday rescheduling
Consistency
Synchronization

ABSTRACT

We consider a combined manpower routing and scheduling problem for performing spatially distributed jobs that demand one or more skilled workers. In particular, we investigate how the staffing can be updated for new jobs that occur within the planning horizon. We address this by interday planning and intraday rescheduling strategies under team consistency consideration. We propose linked mathematical optimization models for the interday and intraday rescheduling. For solving large problem instances, we develop a fix-and-optimize heuristic. Our experiments analyze the effectiveness of the method and reveal the impact of integrating team consistency into manpower scheduling.

1. Introduction

An effective use of available personnel is one of the main instruments for companies to gain a market advantage, to increase customer satisfaction and to improve their core competence and business performance. For this reason, efficient workforce management is crucial for the prosperity of service-oriented companies. The importance of an effective workforce planning stems also from the fact that companies often provide complex services that require specialists with different qualifications and experience levels. In addition, these services often have to be provided at geographically dispersed customer locations. For an effective workforce management and scheduling as well as for the improvement of service quality, companies may thus create multi-skilled worker teams. Deciding on the order in which these teams perform their assigned jobs then directly determines the completion times of jobs and, thus, the service quality perceived by customers as well as the actual working time of each team. For this reason, skill-based team configuration as well as routing of teams and scheduling of jobs have to be considered not separately but in a complementary way. The relevance of this concern is reflected in the number of practical applications found in the maintenance sector, the telecommunication industry, the construction sector, airline catering or the home health care business, see e.g., Cordeau et al. (2010), Hurkens (2009), Kasirzadeh et al. (2017).

Another important challenge faced by companies is the adjustment of staffing and work plans to demand changes when new requests arrive or already scheduled requests have to be postponed or canceled while a previously determined work plan is executed. Due to these fluctuations, companies have to determine not only a work plan for the current planning period but also for the near-future.

A question that arises in this context of multi-period planning is to what extent the team composition should be changed from day to day. Usually, staff shortages may force companies to frequently change team compositions from period to period. However, empirical studies in the home health care sector demonstrate that a frequent change of coworkers (team members) at the operative level leads to miscommunications among the team members and, as a consequence, to an increasing number of failures, see e.g., Kalisch et al. (2008) and Russell et al. (2011). The need to frequently familiarize with new staff members undermines the staff working relationship, increases the stress level, decreases the ability to cope on the unit and results in a high level of frustration. These findings indicate the importance of integrating *team consistency* aspects into multi-period workforce planning.

In Anoshkina and Meisel (2019), we investigated the problem of multi-skilled workforce teaming and routing for a single period, where team consistency is not an issue. In this paper, we consider the problem from a multi-period perspective. Thereby, our primary interest lies in increasing employee satisfaction. We try to achieve this by striving for consistent team compositions during the considered planning horizon. Furthermore, we attempt to find an appropriate balance between the employee needs and the service quality of the work plan, which we measure by the number of performed jobs. Note that these objectives are conflicting because consistency of teams (i.e., avoiding changes in team

* Corresponding author. Christian-Albrechts-University of Kiel, School of Economics and Business, Olshausenstr. 40, 24098, Kiel, Germany.
E-mail addresses: yulia.anoshkina@bwl.uni-kiel.de (Y. Anoshkina), meisel@bwl.uni-kiel.de (F. Meisel).

https://doi.org/10.1016/j.ejtl.2020.100012
Received 10 January 2020; Received in revised form 5 June 2020; Accepted 5 June 2020

structures) might hinder performing the maximum number of jobs. Minimization of the total job completion times, which is a main goal in many earlier contributions, is considered as a subordinate objective only.

Our contribution is then threefold. First, we propose an *interday model* that composes teams to serve requests in an upcoming period where team consistency is one of the objectives next to service quality measures. Thereby, team consistency is modeled by measuring differences in team composition among two consecutive periods. We present two alternatives for measuring and modeling team consistency. Second, we formulate an *intraday model* for integrating new incoming jobs into a current operation plan taking into account the already made work progress of the teams. This model also supports team synchronization, where two or more teams jointly fulfill a job. Finally, we develop a *fix-and-optimize heuristic* for solving large problem instances. The heuristic comprises components for generating initial solutions, splitting and merging of teams, swapping of jobs and diversifying the search in a multi-start process.

The paper is organized as follows. In Section 2, we provide a literature review. We then present formal descriptions and mathematical formulations of the interday and the intraday problems together with an example in Section 3. In Section 4, we explain our solution method. Section 5 contains the computational study that analyses the potentials and limitations of the proposed models and the heuristic. Concluding remarks are presented in Section 6.

2. Literature

The combined problem of teaming and scheduling of multi-skilled workforce was first introduced by the optimization challenge ROADEF (2007). The introduced topic was based on a real-world problem encountered by France Telekom. The considered problem was to compose teams and to schedule jobs with respect to skills of workers and requirements of jobs for a single period. Furthermore, the jobs had different priorities and could be outsourced at some costs, which is bounded by a budget constraint. The objective of the problem is to minimize the weighted sum of the completion times of all processed jobs. This study gave rise to a number of competing solution procedures proposed by the participants of the contest. For instance, Estellon et al. (2009) propose a combination of a greedy algorithm with local search methods. Hurkens (2009) presents a two-phase approach where a MIP model is first applied for the identification of jobs to outsource and a matching model determines then the job-technician-assignment. This approach was later improved by Firat and Hurkens (2012) who add new matching mechanisms. Cordeau et al. (2010) propose a linear optimization model for this problem and develop a meta-heuristic that combines a construction procedure with an adaptive large neighbourhood search (ALNS) for an effective solution of real world instances. Khalfay et al. (2017) introduce further (greedy) heuristics for the model presented by Cordeau et al. (2010) that can solve larger instances. Similar to Hurkens (2009), Hashimoto et al. (2011) propose a decomposition approach using optimization models for the outsourcing decision. For the scheduling of the non-outsourced jobs, the authors apply a greedy randomized adaptive search procedure. Firat et al. (2016) combine teaming with the concept of a so-called stable workforce assignment in which employees show preferences for performing certain jobs. An assignment is considered stable if each technician performs one of his/her most preferred jobs. In contrast to other studies, every technician can get assigned at most one job. Since only a single job is assigned to each team in this problem, routing and scheduling decisions are out of scope of this study. Apart from ROADEF-related studies, Ho and Leung (2010) address a manpower scheduling problem from the airline catering industry that combines team construction and job assignment for a set of employees with different skills. In their study, the maximum size of each team is restricted to only two employees.

Other authors propose a number of rich extensions of these problems. For instance, Kovacs et al. (2012) integrate routing of technicians into the model of Cordeau et al. (2010) and present a corresponding optimization model for the minimization of total routing and outsourcing cost. For the solution of the resulting problem, they also apply an ALNS heuristic. A more complex extension is introduced by Zamorano and Stolletz (2016) who examine the problem from a multi-period perspective. They develop a branch-and-price algorithm with two alternative decomposition schemes: day decomposition and team decomposition. The results demonstrate that the proposed algorithm can solve instances with up to three teams with two technicians each. Finally, Anoshkina and Meisel (2019) investigate a combined teaming and routing problem that is solved in a linear decomposition framework under service-, cost- and fairness-objectives. The service quality is measured by the sum of job completion times. The cost objective refers to labor cost, which is approximated by the total employee working time. A fair workload distribution is reached by minimization of the longest working time among all teams. They show that a decomposition approach helps to find better solutions for larger problem instances. The results reveal that the chosen objective and the employee qualifications both have a strong impact on the solution quality.

Another stream of research focuses on additional aspects but leaves team building decisions out of scope. Motivated from an application of electronic equipment maintenance, Mathlouthi et al. (2018) combine scheduling and routing of individual technicians with inventory management for spare parts. Specifically, each technician starts its route with an initial inventory of spare parts and can later replenish it by visiting the depot. Chen et al. (2016) investigate home services and analyze the relationship between practical experience of employees and service times. The addressed problem is modeled as a Markov decision process. The study reveals that learning effects have a strong impact on the obtained solutions. Chen et al. (2017) extend this problem to the multi-period case. Furthermore, Cappanera et al. (2013) investigate asymmetric skill-based routing problems and propose various models and valid inequalities to derive tight integer linear programs that can be solved quickly. Finally, Van Eck et al. (2017) investigate the scheduling of multi-skilled workforce as a part of a business process where organizational and behavioral aspects come into the play. For a more detailed survey of workforce planning incorporating skills, we refer the interested reader to De Bruecker et al. (2015). A related survey with a discussion of applications and solution methods is provided by Paraskevopoulos et al. (2016).

Although a large number of different aspects has been considered in these works, there is a lack of general guidance about how a schedule can be adapted to future demand changes when new jobs appear during the planning horizon that have to be inserted into the baseline schedule. Studies in this area focus either on rerouting of single employees (Borenstein et al. (2010); Petrakis et al. (2012); Pillac et al. (2012)) or on staff scheduling without routing decisions (Siferd and Benton (1994); Patric et al. (2017); Maenhout and Vanhoucke (2018)). The multi-period approach of Zamorano and Stolletz (2016) is based on a long term planning and does not support updating of an existing schedule due to newly arriving jobs. Hence, there is a distinct lack of models and methods addressing rescheduling issues in combination with manpower teaming and routing. Furthermore, consistency requirements have gained an increasing attention in operations research, but typically as an extension of a classical vehicle routing problem (Gröer et al. (2009); Smilowitz et al. (2013); Feillet et al. (2014); Coelho et al. (2012); Kovacs et al. (2014)). Such studies try to increase customer satisfaction by assigning the same caregiver to one region (driver consistency), by delivering the products with a constant frequency (visit spacing consistency) or during the same time interval for the client (time consistency), or by guaranteeing a specified delivery quantity to each client (quantity consistency). However, no study analyses how the consistency requirement of the workforce teams can be integrated into multi-period workforce routing and scheduling. To bridge this gap, we investigate here how to solve the problem sequentially as a period-by-period interday planning and how to update the generated solutions in an intraday planning when new jobs

arrive, where team consistency is one of the considered objectives.

3. Mathematical optimization models

3.1. Notation and basic assumptions

We consider a planning horizon that is divided into a set D of periods and a workforce that consists of a set M of differently skilled employees, which are available throughout the whole planning horizon. In each period (e.g., a day), a set of teams T is composed out of workforce M to perform a given set of jobs that require different qualifications and experience levels. Each team consists of one or more employees. Each employee $m \in M$ can possess numerous skills from a skill set K at different competence levels L. The level of competence is related to personal features like experience, education or specialization of an employee in the corresponding skill domain. We describe the competences of employee m by the binary matrix Q_m. Each column $k \in K$ refers to a skill and each row $l \in L$ refers to a competence level. An element $Q_m(k,l) = q_{mkl}$ takes value 1 if the employee is qualified in skill $k \in K$ at level $l \in L$ and 0 otherwise. We assume that $q_{mkl'} \leq q_{mkl}$ is satisfied for all $l' > l$, i.e., if employee m possesses skill k at a higher level l' he/she can also perform jobs that require a lower experience level l for this particular skill. We give an example of such a matrix Q_m with $|K| = 3$ skills and $|L| = 3$ competence levels below. In this example, the considered employee m is proficient in skill $k = 1$ at the highest competence level $l = 3$ (and, thus, also at levels $l = 1$ and $l = 2$), in skill $k = 2$ only at level $l = 1$, and in skill $k = 3$ not at all. Based on the q_{mkl} values of employees that are grouped in a team, we compute the qualification level of the team by summing up the skills of the individual team members.

$$Q_m = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{matrix} l=1 \\ l=2 \\ l=3 \end{matrix}$$

We denote by J_d the set of jobs that are relevant for the scheduling of teams on period $d \in D$ of the planning horizon. Not all these jobs have to be performed but maximizing the number of performed jobs is one of the pursued objectives. The jobs can differ in the number of required employees, their skills and experience levels. Integer matrix R_j describes the qualification requirement of a job $j \in J_d$. Here, an element $R_j(k,l) = r_{jkl}$ gives the cumulated number of employees with qualification k and experience level l required for performing job j. The subsequent matrix shows an example of such qualification requirements. The part on the left presents the underlying qualification vectors of employees required in particular skill domains. According to the part on the right, job j requires three employees with skill $k = 1$, one being proficient at least at level $l = 3$ (which also covers $l = 2$ and $l = 1$), one further employee at level $l = 2$ (which also covers $l = 1$) and one further employee at level $l = 1$. These required skill levels are indicated by bold values in the left part whereas the implicitly covered skill levels are non-bold. Furthermore, one employee with proficiency level $l = 1$ is needed in domain $k = 2$ and two employees must be trained at least at level $l = 2$ for skill $k = 3$ (which also covers $l = 1$).

$$R_j = \begin{pmatrix} 1+1+\mathbf{1} & \mathbf{1} & 1+1 \\ 1+\mathbf{1}+0 & 0 & \mathbf{1}+\mathbf{1} \\ \mathbf{1}+0+0 & 0 & 0+0 \end{pmatrix} \begin{matrix} l=1 \\ l=2 \\ l=3 \end{matrix} = \begin{pmatrix} 3 & 1 & 2 \\ 2 & 0 & 2 \\ 1 & 0 & 0 \end{pmatrix} \begin{matrix} l=1 \\ l=2 \\ l=3 \end{matrix}$$

Each job j is further characterized by a processing time p_j. Processing job j has to start within a given time interval $[a_j, b_j]$ where a_j and b_j denote the earliest and the latest start times correspondingly. Furthermore, the jobs occur at different locations. We model the corresponding network for a period $d \in D$ as a connected graph $G_d = (J_d^0, E_d)$, where $J_d^0 = \{0\} \cup J_d$ is a node set that includes the depot 0 and the locations of those jobs that are given for day d. $E_d = \{(i,j) | i,j \in J_d^0\}$ is the corresponding set of edges. Each edge $e \in E_d$ is associated with a non-negative travel time g_{ij}. Finally, our modeling and solution approach is based on the following assumptions:

- A job $j \in J_d$ can be carried out by a team only if the aggregated team skills meet the job's qualification requirements r_{jkl} for all $k \in K$ and $l \in L$.
- Job processing times are constant and independent of the team composition. Interruption of processing a job is forbidden.
- Jobs that cannot be performed are rejected or outsourced.
- Jobs may arrive in the course of the planning horizon and even within a currently considered period.

3.2. Planning framework

Our multi-period scheduling concept bases primarily on the following ideas. At the beginning of period d, teams are composed and routes for the initially known requests J_d are determined. The teams then start executing their assigned jobs. Then, in the course of time, new jobs arrive. According to the urgency of a new job that arrives in period d, the job is classified as either normal priority or high priority. Normal priority jobs have time windows for future periods and, thus, are added to the corresponding sets $J_{d+1} \ldots J_{|D|}$ for subsequent periods. High-priority jobs $j \in J^h$ must be served within the current period d. For this reason, the scheduling of these jobs assumes short response time as well as the ability to communicate quickly with the teams for the assignment of new requests. In addition, it requires real-time knowledge of the positions of all teams in order to identify the new start nodes of their updated routes and their sets of still open requests. High-priority jobs that cannot be included in the current schedules of the teams are rejected or outsourced. As the different time frames for normal and high-priority jobs call for respective planning approaches, we propose two linked models for scheduling these jobs. A so-called *interday model* plans the operations for the subsequent period $d+1$ with respect to the teams composed for the current period d and the jobs known for the next period $d+1$. Planning is made on a daily basis, as, according to the concept described above, job information about future periods might be highly incomplete at the time of planning. It thus handles the transition of teams from period to period while trying to keep team compositions consistent if possible. An *intraday model* aims at inserting high-priority jobs J^h into the already determined work plan for period d that is currently executed by the team. For a better understanding of the model invoking process, Fig. 1 illustrates the planning process for four time periods. At the beginning of the planning horizon an initial schedule S_1 is constructed for day 1. High-priority jobs that appear within day 1 lead to an update of schedule S_1 which is done by the *intraday model*. At the end of day 1, the *interday model* then generates the schedule for the next period 2. The same processes are applied to generate and adapt schedules S_2, S_3 and S_4 in subsequent periods. We present the interday model in Section 3.3 and the intraday model in Section 3.4.

3.3. Interday model

The interday model creates a schedule for a day $d \in D$, taking into account the team composition from the previous day $d-1$. By applying this model sequentially for each day of the planning horizon, we create a sequence of schedules that are linked by the composition of teams. To

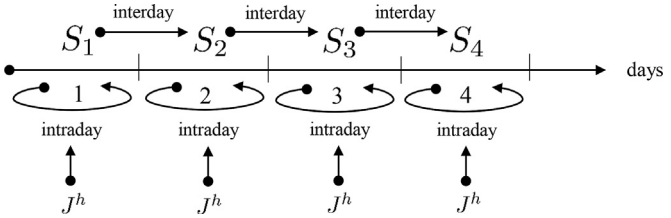

Fig. 1. Planning procedure.

handle the interdependencies between the days, we introduce the binary parameter x'_{mt} that indicates if employee m was assigned to team t on the previous day $d-1$ ($x'_{mt}=1$) or not ($x'_{mt}=0$). A corresponding binary decision variable x_{mt} is used for modeling the new team structure on the current day d, i.e., $x_{mt}=1$ if employee m is assigned to team t, 0 otherwise.

Linking periods of the planning horizon, we allow merging or splitting of teams at consecutive periods. Fig. 2 shows an example of a team composition at day $d=1$ where three employees are assigned to team 1 and two employees to teams 2 and 3 by decision variables $x_{11}=x_{21}=x_{31}=x_{42}=x_{52}=x_{63}=x_{73}=1$. These variables then become the input x'_{mt} for the subsequent decision making at day $d=2$. The schedule for day 2 restructures the teams generated on the first day to meet the qualification requirements of jobs in J_2. Thereby, the original team 1 is split into two teams while the original teams 2 and 3 are merged into a new single team 3, which is expressed by corresponding decision variables $x_{11}=x_{21}=x_{32}=x_{43}=x_{53}=x_{63}=x_{73}=1$ for day $d=2$. This means that three employees (3, 4, 5) change their team, which can be seen from comparing the x_{mt} decision variables of day $d=2$ with the x'_{mt} parameters derived from the decisions of the previous day. The number of employees that switch their team is used for assessing the team consistency in the interday optimization model. Clearly, since the indices of teams are somewhat substitutable, the same measure can also result from an alternative numbering of teams (e.g. if employee 3 would form a new team $t=3$ and employees 6 and 7 would be merged into team $t=2$, we would have other values of x_{mt} but the same number of total changes). This makes the consistency measure somewhat arbitrary. Furthermore, one could argue that employees 4, 5, 6 and 7 should have the same value of the consistency binary variable since they are facing the same outcome, which is that each of them is merged within one large team. From this employee-perspective, alternative consistency measures might be defined for this problem. One example of such an alternative is provided in Appendix A. Anyhow, as is shown there, both measures lead to almost identical results but the measure presented here is more attractive from a computational perspective.

We furthermore introduce the following decision variables. The routing of teams at day d is denoted by binary decision variable z_{tij}, which takes value 1 if team t performs job i directly before job j and 0 otherwise. The scheduling variable s_{tj} defines the start time of job j by team t. Similar, f_{tj} denotes the completion time of job j executed by team t. Note that team-specific job start times s_{tj} and completion times f_{tj} are not mandatory in the interday model but later required in the intraday model for synchronizing two or more teams that jointly perform a job j. Therefore, we employ s_{tj} and f_{tj} in the interday model for reasons of consistency. Furthermore, we conducted preliminary experiments where omitting index t for these variables did not improve model performance. Finally, in order to capture the team consistency in the objective function, we introduce an auxiliary binary variable X_m that takes value 1 if employee m switches the assigned team from day $d-1$ to day d. Using the introduced notation the interday model is formulated as follows:

$$\text{minimize}: \ \alpha \cdot \sum_{m\in M} X_m - \beta \cdot \sum_{t\in T}\sum_{i\in J_d^0}\sum_{j\in J_d} z_{tij} + \gamma \cdot \sum_{t\in T}\sum_{j\in J_d} f_{tj} \tag{1}$$

Subject to:

$$\sum_{t\in T} x_{mt} \le 1 \qquad \forall m \in M \tag{2}$$

$$\sum_{m\in M} x_{mt}\cdot q_{mkl} \ge r_{jkl}\cdot \sum_{i\in J_d^0} z_{tij} \qquad \forall j\in J_d, k\in K, l\in L, t\in T \tag{3}$$

$$\sum_{j\in J_d} z_{t0j} \le 1 \qquad \forall t\in T \tag{4}$$

$$\sum_{t\in T}\sum_{i\in J_d^0} z_{tij} \le 1 \qquad \forall j\in J_d \tag{5}$$

$$\sum_{i\in J_d^0} z_{tij} = \sum_{i\in J_d^0} z_{tji} \qquad \forall j\in J_d^0, t\in T \tag{6}$$

$$f_{ti} + g_{ij} \le s_{tj} + \mathscr{M}\cdot(1-z_{tij}) \qquad \forall i\in J_d^0, j\in J_d, t\in T \tag{7}$$

$$s_{tj} + p_j \le f_{tj} + \mathscr{M}\cdot\left(1-\sum_{i\in J_d^0} z_{tij}\right) \qquad \forall j\in J_d, t\in T \tag{8}$$

$$s_{tj} \ge a_j - \mathscr{M}\cdot\left(1-\sum_{i\in J_d^0} z_{tij}\right) \qquad \forall j\in J_d, t\in T \tag{9}$$

$$s_{tj} \le b_j + \mathscr{M}\cdot\left(1-\sum_{i\in J_d^0} z_{tij}\right) \qquad \forall j\in J_d, t\in T \tag{10}$$

$$x_{mt} - x'_{mt} \le X_m \qquad \forall m\in M, t\in T \tag{11}$$

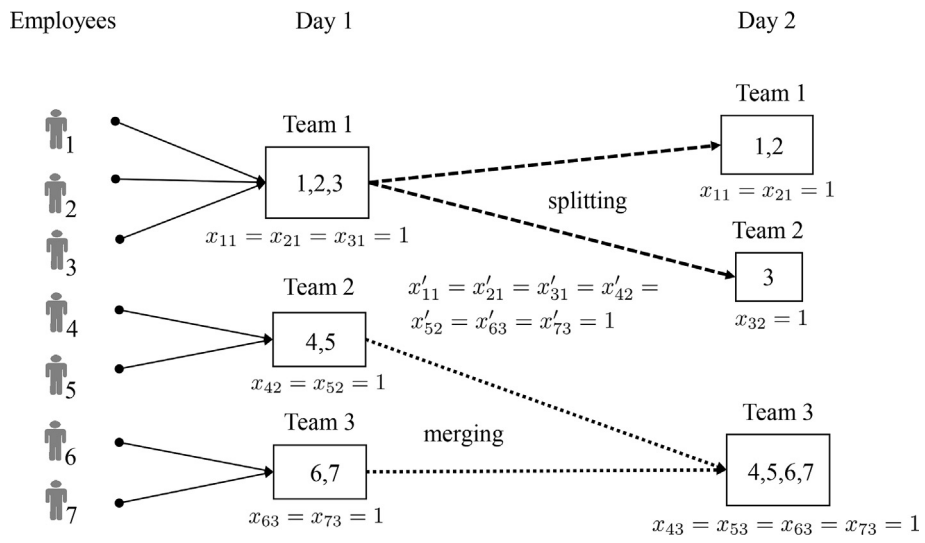

Fig. 2. Rescheduling of teams in interday planning.

$$x'_{mt} - x_{mt} \leq X_m \qquad \forall m \in M, t \in T \tag{12}$$

$$s_{tij}, f_{tij} \geq 0 \qquad \forall j \in J_d^0, t \in T \tag{13}$$

$$x_{mt}, X_m, z_{tij} \in \{0,1\} \qquad \forall i,j \in J_d^0, m \in M, t \in T \tag{14}$$

The main goal of the model is to preserve the stability of the team composition. However, focusing solely on the consistency aspect can have a negative impact on the service quality. For this reason, we seek for a compromise between team consistency and service quality. More precisely, the first component of the objective function (1) maximizes team consistency by minimizing the number of employees that change their assigned team from day $d-1$ to day d. The second component maximizes the number of performed jobs and the third component minimizes the total job completion time. Here, weights α, β and γ are used for expressing different priorities of the three objectives. Note that we measure service quality primarily by the number of performed jobs. Minimization of the total job completion time is considered merely as a subordinate objective because completion times might be determined strongly by (tight) time windows. Anyhow, if time windows are wide, minimizing job completion times can be a service issue, which is why we added it as a minor objective. From this, we assume that $\alpha > \beta > \gamma$. The proposed objective function captures two issues that are practically relevant for companies: employee satisfaction and service quality. With this model, we are able to analyze the tradeoff of these two relevant yet conflicting objectives.

Constraints (2) guarantee that each employee is assigned to at most one team. Constraints (3) ensure that created teams are sufficiently qualified for performing their assigned jobs. Constraints (4) demand that each team departs from the depot at most once. Constraints (5) state that each job is visited by at most one team. Constraints (6) ensure that each team visiting a node j also leaves this node. Constraints (7) determine the start time of job j with respect to the finishing time of the preceding job i and the corresponding traveling time. Here as well as in further constraints, $\mathscr{M}$ denotes a sufficiently large positive value. Constraints (8) define the time at which job j is completed by team t. Together, (7) and (8) also avoid subtours in the solution. Constraints (9)–(10) reflect the time windows for the starting time of job j. Constraints (11)–(12) set the auxiliary variables X_m. Constraints (13)–(14) specify the domains of decision variables.

3.4. Intraday model

The task of the intraday model is to insert newly appearing high-priority jobs J^h into the baseline schedule generated for a current period. Fig. 3 illustrates an example of a baseline schedule with one new incoming job and two insertion strategies. In this example, there are three skill domains. For reason of simplicity, we consider only one qualification level. The vector attached to each job in Fig. 3a describes the job's skill requirements. The vector attached to a team describes the cumulated skills of those employees that form this team. The first time interval attached to each job in Fig. 3a indicates the scheduled service time for performing the corresponding job in the baseline schedule while the values in square brackets represent the given time window for the job start time. The consecutive arrows indicate the route for each team. For instance, team 1 performs three jobs 3, 8 and 9 in this order in the baseline schedule.

As the intraday replanning of jobs $j \in J^h$ occurs during the execution of the baseline schedule, this schedule has to be updated with minimal deviation such that the number of inserted jobs from set J^h is maximized. Thereby, the following consistency requirements are considered:

- Team consistency: Teams created by the interday model are kept and transferred to the intraday model as parameter x'_{mt}.
- Schedule consistency: All scheduled jobs $j \in J_d$ must be served by the already assigned teams and cannot be canceled or reassigned to other teams.
- Time consistency: The start time of already scheduled but not yet performed jobs $j \in J_d$ can be changed, but it must still comply with the time window $[a_j, b_j]$.
- High-priority jobs $j \in J^h$ must be scheduled after the point in time τ when they arrive and before the end of the working day e_{max}. Thus, the time window for these jobs is $[a_j, b_j] = [\tau, e_{max}]$. Here, τ either indicates the arrival time of a single high-priority job or the point in time when the planning is to be conducted for a set of jobs that arrived up to that time, depending on the planning policy of the company.

In Fig. 3b, a single new high-priority job $J^h = \{10\}$ arrives at time $\tau =$ 1:10. This job's skill requirements are $r_{10,k,1} = (2,1,1)$. Based on the above assumptions, we propose the following rescheduling strategies:

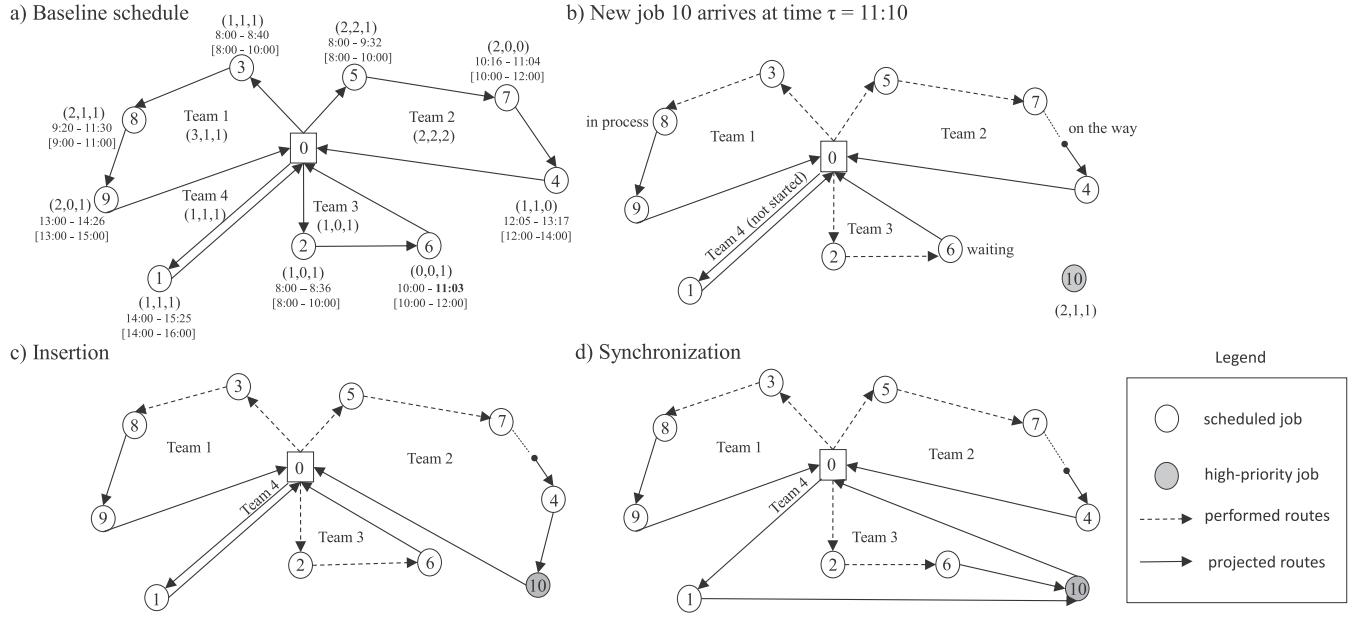

Fig. 3. Intraday planning.

a) *Insertion*: Job $j \in J^h$ is inserted into the existing route of a sufficiently qualified team t. In the example in Fig. 3c, the new job is inserted into the route of team 2 right after job 4.
b) *Synchronization*: Two or more teams are synchronized for jointly performing job j if the qualification of one team is insufficient or if j cannot be inserted in the existing tour of any single qualified team (e.g., due to time windows of already scheduled jobs). In the example in Fig. 3d, the teams 3 and 4 both visit the location of job 10 for jointly performing this job. Since the teams may arrive at the job location at different times, we denote by δ_{max} a maximal allowed temporal distance.

Obviously, at the moment when a new high-priority job arrives, the current positions of all teams have to be identified. Thereby, one of the following four situations is observed for each team:

- the team is performing a job (see team 1 in Fig. 3b),
- the team is on the way to its next job (see team 2 in Fig. 3b),
- the team has completed its last assigned job and is waiting for a new job (see team 3 in Fig. 3b),
- the team has not started from the depot yet (see team 4 in Fig. 3b).

The current location of a team $t \in T$ represents the starting point (virtual depot) of its remaining route, which is denoted as D_t. In Fig. 3b, we have $D_1 = 8, D_2 = 4, D_3 = 6$ and $D_4 = 0$. Furthermore, we denote by f_t^τ the earliest point in time at which team t becomes available at its location D_t. For our example, $f_1^\tau = f_{1,8}$ is the completion time of the currently processed job 8 according to the baseline schedule, $f_2^\tau = f_{2,4}$ is the planned completion time of job 4 towards which team 2 is currently moving, and $f_3^\tau = \tau$ as well as $f_4^\tau = \tau$ refer to the current point in time as these teams are ready immediately.

We further define by J_t the set of jobs that are relevant for team t in the intraday replanning. This set includes those jobs that are already assigned to team t in the baseline schedule but that are not yet started at time τ. It furthermore includes all new high-priority jobs J^h, the virtual depot D_t, and the original depot 0 which is the ending location of all routes. For the example above $J_1 = \{0, 8, 9, 10\}$, $J_2 = \{0, 4, 10\}$, $J_3 = \{0, 6, 10\}$ and $J_4 = \{0, 1, 10\}$.

The corresponding planning decisions are modeled with the same routing variables (z_{tij}) and scheduling variables (f_{tj}, s_{tj}) introduced in Subsection 3.3. Note that the earlier decision variable x_{mt} becomes now the parameter x'_{mt} because the team compositions are not changed in the intraday replanning. Furthermore, we save the start times s_{tj} that are assigned to the jobs in the baseline schedule as parameters s'_{tj}. We also introduce a continuous variable S_{tj} which denotes the absolute deviation in the start time of job j by team t after rescheduling. We specify by h_j a binary variable which takes value 1 if high-priority job $j \in J^h$ is processed by any team and 0 otherwise. Using the introduced notation, the intraday model for including a set of high-priority jobs J^h into a partly executed baseline schedule is formulated as follows.

$$\text{maximize: } \alpha \cdot \sum_{j \in J^h} h_j - \beta \cdot \sum_{t \in T} \sum_{i \in J_t} \sum_{j \in J^h} z_{tij} - \theta \cdot \sum_{t \in T} \sum_{j \in J_t \setminus \{J^h\}} S_{tj} - \gamma \cdot \sum_{t \in T} \sum_{j \in J_t} f_{tj} \tag{15}$$

Subject to:

$$\sum_{t \in T} \left(\sum_{m \in M} x'_{mt} \cdot q_{mkl} \cdot \sum_{i \in J_t} z_{tij} \right) \geq r_{jkl} \cdot h_j \qquad \forall j \in J^h, k \in K, l \in L \tag{16}$$

$$\sum_{j \in J_t} z_{tD_t j} = 1 \qquad \forall t \in T \tag{17}$$

$$\sum_{i \in J_t} z_{tiD_t} = 0 \qquad \forall t \in T, D_t \neq 0 \tag{18}$$

$$\sum_{i \in J_t} z_{ti0} = 1 \qquad \forall t \in T \tag{19}$$

$$\sum_{i \in J_t} z_{tij} = 1 \qquad \forall t \in T, j \in J_t \setminus \{D_t, 0\} \setminus J^h \tag{20}$$

$$\sum_{i \in J_t} z_{tij} \leq 1 \qquad \forall j \in J^h, t \in T \tag{21}$$

$$\sum_{i \in J_t} z_{tij} = \sum_{i \in J_t} z_{tji} \qquad \forall t \in T, j \in J_t \setminus \{D_t, 0\} \tag{22}$$

$$f_{tD_t} = f_t^\tau \qquad \forall t \in T \tag{23}$$

$$f_{ti} + g_{ij} \leq s_{tj} + \mathscr{M} \cdot (1 - z_{tij}) \qquad \forall t \in T, i \in J_t, j \in J_t \setminus \{0\} \tag{24}$$

$$s_{tj} + p_j \leq f_{tj} + \mathscr{M} \cdot \left(1 - \sum_{i \in J_t} z_{tij}\right) \qquad \forall t \in T, j \in J_t \setminus \{0\} \tag{25}$$

$$s_{tj} - s_{t'j} \leq \delta_{max} + \mathscr{M} \cdot \left(2 - \sum_{i \in J_t} z_{tij} - \sum_{i \in J_{t'}} z_{t'ij}\right) \qquad \forall j \in J^h, t, t' \in T, t \neq t' \tag{26}$$

$$s_{t'j} - s_{tj} \leq \delta_{max} + \mathscr{M} \cdot \left(2 - \sum_{i \in J_t} z_{tij} - \sum_{i \in J_{t'}} z_{t'ij}\right) \qquad \forall j \in J^h, t, t' \in T, t \neq t' \tag{27}$$

$$s_{tj} \geq a_j - \mathscr{M} \cdot \left(1 - \sum_{i \in J_t} z_{tij}\right) \qquad \forall t \in T, j \in J_t \setminus \{D_t, 0\} \tag{28}$$

$$s_{tj} \leq b_j + \mathscr{M} \cdot \left(1 - \sum_{i \in J_t} z_{tij}\right) \qquad \forall t \in T, j \in J_t \setminus \{D_t, 0\} \tag{29}$$

$$s_{tj} \geq f_t^\tau + g_{D_t, j} - \mathscr{M} \cdot \left(1 - \sum_{i \in J_t} z_{tij}\right) \qquad \forall t \in T, j \in J_t \setminus \{D_t, 0\} \tag{30}$$

$$s_{tj} - s'_{tj} \leq S_{tj} \qquad \forall t \in T, j \in J_t \setminus J^h \tag{31}$$

$$s'_{tj} - s_{tj} \leq S_{tj} \qquad \forall t \in T, j \in J_t \setminus J^h \tag{32}$$

$$s_{tj}, f_{tj}, S_{tj} \geq 0 \qquad \forall t \in T, j \in J_t \tag{33}$$

$$h_j, z_{tij} \in \{0, 1\} \qquad \forall t \in T, i, j \in J_t \tag{34}$$

Objective function (15) maximizes the service quality and minimizes the deviations from the initial planning. More precisely, the first term of the objective maximizes the number of scheduled high-priority jobs. The second term minimizes the number of synchronization processes for high-priority jobs. The third term minimizes the absolute change of job starting times compared with the baseline schedule. The fourth term minimizes the total job completion time of all scheduled jobs. Here, weights α, β, θ and γ are used for expressing different priorities of the four objectives. Constraints (16) demand that high-priority jobs are performed by teams with appropriate skills where multiple teams might be involved in serving a job. Constraints (17) guarantee that each team continues the route from its current position (virtual depot). If the current position D_t of the team does not correspond to the original depot (node 0), Constraints (18) forbid to return to the virtual depot D_t and, also, to insert jobs before the virtual depot into the route. Constraints (19) stipulate that all teams return to the depot 0 at the end of their tour.

Constraints (20) ensure that each team performs all its already assigned jobs. Constraints (21) state that each high-priority job can be visited by a team at most once. Constraints (22) balance the flow ensuring that each team visiting a node j also leaves this node. Constraints (23)–(25) define the start and completion times of jobs. Note that, similar to the interday model, Constraints (24)–(25) allow to avoid subtours in the solution. Constraints (26)–(27) bound the difference between the arrival times of synchronized teams. As the number of teams to synchronize for a job j is not limited, the comparison is performed for every pair of teams. Constraints (28)–(29) state that all jobs, if scheduled, have to be performed within the predefined time windows. Constraints (30) establish a lower bound on the start times of jobs. Here, a not yet processed job j cannot be started before the assigned team t is released at its virtual depot D_t at time f_t^s and at least $g_{D_t,j}$ time units have elapsed. The latter is a lower bound on the traveling time in case that the team travels directly from D_t to j. Constraints (31)–(32) define the values S_{tj}. Constraints (33)–(34) specify domains of decision variables.

4. Fix-and-optimize heuristic

While the intraday model can be solved relatively fast with a standard MIP solver for reasonably sized instances, the interday model can be solved quickly only for small problem instances. This is also because the interday problem includes decisions on team formation into an underlying uncapacitated VRP, which is known to be NP-hard, see Balas (1989), Kovacs et al. (2012). As a heuristic solution approach for the interday problem, we introduce a fix-and-optimize algorithm that combines mathematical programming with heuristic search. This method provides an iterative solution approach that significantly reduces the computational effort.

The interday planning process comprises three interdependent decisions: team building, job assignment, and routing. The difficulty stems mostly from the large number of routing variables z_{tij}. Our heuristic therefore strives for reducing the computational effort by transferring routing decisions to a subordinate level and by reducing the number of these variables. For this purpose, we propose a combination of team and job decomposition techniques meaning that routing has to be optimized for each created (fixed) team iteratively where the job set is fixed for every candidate team. The heuristic comprises of four phases: generation of an initial solution (Subsection 4.1), improvement by merging and splitting of teams (Subsection 4.2), improvement by swapping of jobs (Subsection 4.3) and randomized disturbance for diversifying the search (Subsection 4.4).

4.1. Initial solution

The purpose of this algorithm is to generate an initial baseline schedule for a period d. The primary idea here is to use the given workforce teams from the previous period $d-1$ (for reasons of team consistency) and to process as many jobs as possible from those jobs J_d that have to be performed in the current planning period d. For initializing the employee-team assignment for the current period we set $x_{mt} = x'_{mt}$, where x'_{mt} refers to the employee-team-assignment of the previous period. These teams form the set T. For the first period, where no previous employee team-assignment is available, teams T require an alternative initialization like, for example, adding all employees to one large team or forming equally large teams. The procedure for generating the initial schedule for the teams T is outlined in Algorithm 1 in Appendix B. It starts by initializing sets $J_t = \{0\}$ of jobs that are allocated to team t (where only the depot 0 is initially allocated). We next determine for each team $t \in T$ the set $J_t^{qual} \subseteq J_d$ of jobs for which this team is sufficiently qualified. We do this by comparing skill vectors of team members (q_{mkl}) with qualification requirements of each job j (r_{jkl}). Job j enters set J_t^{qual} if $\sum_{m \in M} q_{mkl} \cdot x_{mt} \geq r_{jkl}$. Note that sets J_t^{qual} are not necessarily disjunct. The algorithm then sorts the jobs in J_t^{qual} in ascending order of a *team-job overqualification factor* μ_{tj} which is computed as follows:

$$\mu_{tj} = \sum_{k \in K} \sum_{l \in L} \left(\sum_{m \in M} q_{mkl} \cdot x_{mt} - r_{jkl} \right) \quad \forall t \in T, j \in J_t^{qual} \tag{35}$$

This step aims at reducing overqualification of job-to-team-assignments in the subsequent steps of the procedure. The next steps consider the teams one by one in a loop, see lines 4–10 in Algorithm 1. In each iteration, we first pick the team $t \in T$ that is qualified for the smallest number of jobs among all teams (see line 5), where ties are broken arbitrarily. For this team, we iteratively identify jobs to perform and solve the corresponding routing problem. Therefore, we create a temporary subset J^{temp} that contains up to λ^{max} jobs. More precisely, this set is composed of all jobs from J_t and further $\lambda^{max} - |J_t|$ jobs from J_t^{qual}, see line 7. We set the bound λ^{max} to guarantee that a single team does not get excessively many jobs where the overall solution quality (e.g., the total job completion times) might deteriorate from. Furthermore, this bound reduces the computational time for each routing subproblem and, thus, the total computational effort. Finally, recall that jobs in set J_t^{qual} are sorted according to their similarity to the qualifications of the team. From this, considering in each iteration only λ^{max} jobs, we give priority to the more appropriate jobs to be assigned to this team. We then remove all currently examined jobs from J_t^{qual}, see line 8. Afterwards, we solve a routing model for team t and jobs J^{temp} to identify which of those jobs the team can actually process (see line 9). Although this test could be conducted using simple insertion heuristics or the like, we solve an optimization problem here to guarantee that the maximum possible number of jobs is inserted. The corresponding routing model is formulated as follows:

$$\text{maximize}: \quad \beta \cdot \sum_{i \in J^{temp}} \sum_{j \in J^{temp}} z_{tij} - \gamma \cdot \sum_{j \in J^{temp}} f_{tj} \tag{36}$$

$$\text{subject to}: \quad (4), (6) - (10) \text{ and}$$

$$\sum_{i \in J^{temp}} z_{tij} \leq 1 \qquad \forall j \in J^{temp} \backslash J_t \tag{37}$$

$$\sum_{i \in J^{temp}} z_{tij} = 1 \qquad \forall j \in J_t \tag{38}$$

$$s_{tj}, f_{tj} \geq 0 \qquad \forall j \in J^{temp} \tag{39}$$

$$z_{tij} \in \{0,1\} \qquad \forall i, j \in J^{temp} \tag{40}$$

Objective (36) maximizes the number of performed jobs as the main goal and the job completion times as a subordinate goal ($\beta > \gamma$). Constraints (37) demand that each newly considered job can be visited by the considered team at most once. Constraints (38) guarantee that all jobs J_t that were already assigned to team t in previous iterations are still contained in the route. Constraints (4) and (6)–(10), which are taken from the original interday model, are modified by replacing J_d and J_d^0 with J_t (which also includes depot 0).

Having solved this routing problem, we update J_t such that this set contains all jobs that are actually processed by team t in the obtained route, see line 10. The described procedure is repeated as long as team t processes less than λ^{max} jobs and there are further uninspected jobs in J_t^{qual}, see line 6. Finally, the procedure removes those jobs that are processed by team t from the $J_{t'}^{qual}$ sets of all other teams $t' \neq t$ (see line 12) and it removes team t from the list T of teams that need further inspection (see line 13). Afterwards, the procedure continues with the next team until all teams are processed.

4.2. Splitting and merging of teams

If, in the current solution, not all jobs are processed, the heuristic proceeds with splitting and merging of teams and allocating jobs to the new teams (see Algorithm 2 in Appendix B). We denote by J^{un} the set of jobs that are not processed in the current solution. This set is given as input to Algorithm 2. The procedure starts by identifying free teams T^F to which no jobs have been allocated so far ($|J_t| = 0$), see line 1. In this step, we also create additional free teams by identifying redundant employees in the already used teams T. Therefore, we check for each employee $m \in M$ in each team $t \in T$ with $|J_t| > 0$ if m can be removed from t without turning the job assignment infeasible w.r.t. qualification requirements. Each redundant employee is removed from t and added as a new single-person team to set T^F. In this way, we invoke a feasible *splitting* of existing teams. As free teams are not qualified for performing unprocessed jobs themselves, they have to be merged into larger teams to meet the qualification requirements of jobs. Therefore, the unprocessed jobs $j \in J^{un}$ are sorted according to descending *difficulty factor* ϑ_j (see line 2), which is defined as follows:

$$\vartheta_j = \sum_{k \in K} \sum_{l \in L} r_{jkl} \qquad \forall j \in J^{un} \tag{41}$$

The algorithm then considers jobs $j \in J^{un}$ one after the other. It first checks if the aggregated qualifications of the entirety of free teams T^F allows serving the considered job j, see line 4. In this case, it merges free teams until all skills of job j are covered (see lines 6–10). These teams form the merged team t^M. Thereby, in order to provide an adequate team assignment, we select the next team $t \in T^F$ to take up into t^M according to the minimum *skill gap factor*, see line 7. The *skill gap factor* computes how many of the job qualification requirements are still unmet if team t would be included into t^M. It is computed as follows:

$$\mathscr{E}_{jt} = \sum_{k \in K} \sum_{l \in L} \max \left\{ r_{jkl} - \sum_{m \in M} q_{mkl} \cdot x_{mt^M} - \sum_{m \in M} q_{mkl} \cdot x_{mt}, 0 \right\} \quad \forall j \in J^{un}, t \in T \tag{42}$$

After this process, the algorithm has composed a team t^M that is qualified for processing job j. The algorithm then attempts to assign further unprocessed jobs to this team. Therefore, it first identifies the jobs $J^{qual}_{t^M} \subseteq J^{un}$ for which t^M is qualified (see line 11). Afterwards, it assigns chunks of these jobs to team t^M and solves the temporary routing problems (see lines 12–17) like in the process described for Algorithm 1. Subsequently, it removes all jobs that are processed by team t^M from set J^{un} (see line 18). The process continues until all unprocessed jobs from J^{un} are examined.

4.3. Swapping of jobs

We next try to improve the generated solution by reducing job completion times. The team consistency objective is respected here by still keeping the initial team composition. The procedure is outlined in Algorithm 3 in Appendix B. It first initializes the set of teams T^S that have at least one job assigned in the current solution (see line 1). The algorithm then considers pairs of these teams (t, t') iteratively. It attempts to swap jobs between teams t and t' in order to reduce the total job completion times. For this purpose, the algorithm first identifies a set $J^S_{t',t}$ of jobs that are currently assigned to team t' but that could be served by team t according to job qualification requirements as well as a set $J^S_{t,t'}$ of jobs that could be moved from team t to team t'. If both sets are non-empty, we attempt for each job pair $(j^1, j^2) \in J^S_{t',t} \times J^S_{t,t'}$ to swap the currently assigned jobs. This is done by first attempting to insert job j^1 in the route of team t using the routing model (36)–(40), see line 6. If j^1 is served in the resulting route of team t, the algorithm tries to insert job j^2 in the route of team t' (see line 8). If this is successful too and the resulting solution has a lower total job completion time, the obtained solution is saved and the job sets J_t and $J_{t'}$ are updated accordingly (see lines 10–11).

4.4. Randomized disturbance

Finally, we propose a multi-start procedure (Algorithm 4 in Appendix B) that diversifies the search for good solutions by altering the team compositions. This procedure relaxes the team consistency requirement in order to improve the two other objectives (maximizing the number of processed jobs and minimizing the total job completion time). In order to maximize the number of processed jobs, we conduct a fixed number of I iterations. Each iteration incorporates three steps: 1. modification of the team composition, 2. generation of initial solution and 3. solution improvement by splitting and merging of teams. As a starting point, we take in each iteration the initial employee-team assignment x'_{mt} from the previous day. For modifying this team structure, we remove randomly selected employees from teams and allocate them to randomly selected other teams. The extent of this modification is controlled by the number N of employees that are interchanged, which is a further input parameter of the algorithm. Afterwards, Algorithm 4 employs the previously described Algorithms 1 and 2 to obtain a solution for this new team structure. Objective function (1) is used to keep track of the quality improvement of the best known solution. Finally, we attempt to reduce the total job completion time for the best found solution by calling Algorithm 3. As this algorithm can only improve the total job completion time without deteriorating the other (more important) objectives, we invoke swapping of jobs (Algorithm 3) only once at the end of the procedure.

5. Computational study

The computation experiments aim at testing the performance of the models and methods described in Sections 3 and 4 and at exploring the effect of the team consistency requirement on the scheduling decisions. All tests have been run on an Intel(R) Core (TM) i7-7700 3.60 GHz with 32 GB of RAM. We used CPLEX 12.8 for solving the MIP models with a runtime limit of 3600 s for the interday planning and of 200 s for the intraday replanning. The fix-and-optimize heuristic was implemented in Java 1.8.0. We next describe the used test instances, which is followed by the presentation of results for each planning approach.

5.1. Generation of test instances

For the experimental evaluation of the interday scheduling scheme, we consider a planning horizon of 5 days. We generate 15 instances differing in the number of jobs per day (ranging from 8 to 500) and the number of employees (ranging from 4 to 100). Instances with up to 20 jobs form the so-called small instances, and with up to 80 jobs are the medium instances. These instances are used for analyzing the solvability of the interday model by the CPLEX solver. Instances containing more than 80 jobs form the large instances that are used for further testing the potentials of the heuristic. All instances are generated with $|K| = 3$ skill domains and $|L| = 3$ competence levels. The qualification matrix Q_m of an employee m is generated as follows. The employee can be qualified in each skill $k = 1, 2, 3$ at level $l = 1$ with independent probability of 0.5 and have the same skill at a higher level with a probability of 0.5^l. From this, it is possible that each employee can be proficient in several skills at different competence levels. Furthermore, the generation process guarantees that each employee owes at lea st one skill (i.e., $\forall m \in M : \exists k \in K \Rightarrow q_{mk1} = 1$). The jobs in sets J_d are generated for each planning period as follows. The job requirement matrix R_j is constructed in such a way that each element at level 1 is sampled uniformly from the set $\{0, 1, 2\}$ meaning that job j either requires no employee for skill k ($r_{jkl} = 0$) or one employee ($r_{jkl} = 1$) or two employees ($r_{jkl} = 2$).

The values at levels $l = 2$ and $l = 3$ are 0, $r_{jkl-1} - 1$ or r_{jkl-1} with a probability of 0.33. It is ensured that each job requires at least one skill. Time windows of jobs are generated using uniform distributions with $a_j \sim U[0, 300]$ and $b_j = a_j + 180$, where 0 denotes the beginning of the planning horizon. All values are expressed in minutes and represent an 8-h workday with a total of 480 min. We defined processing times as $p_j \sim U[30, 60]$ minutes. The jobs and the depot are randomly located within an area of size 30 × 30 from which travel times g_{ij} are computed by the corresponding euclidean distances. For the team synchronization process, we bound the maximal temporal distance by $\delta_{max} = 30$.

5.2. Results for the interday planning

We first test the extent to which the proposed interday model can be solved to optimality. Putting emphasis on consistency, we use the following parameters for evaluating the objective function: $\alpha = 100$, $\beta = 1$ and $\gamma = 0.0001$. We refer to this configuration as *consistency setting* as the parameters represent a dominating preference for team consistency followed by the subordinate service objectives of maximizing the number of processed jobs and minimizing the total job completion time. The weights are chosen such that they clearly separate the objectives and establish a hierarchy among them. This means that changing teams comes at prohibitively high cost and, thus, teams will stay the same. In this setting, teaming decisions actually play no role but we use this setting as a benchmark for the more flexible *service setting*. This service setting prioritizes the maximization of the number of performed jobs through parameters $\alpha = 1$, $\beta = 100$ and $\gamma = 0.0001$, which can come along with changes of teams. As a starting point for both settings, we generate a solution for day $d = 1$ by setting the objective coefficients to $\alpha = 0$, $\beta = 100$ and $\gamma = 0.0001$, i.e., by completely ignoring team consistency as there are no teams given from the previous day for the first planning period. For all subsequent days, we solve the interday model taking into account the team composition of the preceding day. For example, the employee-team-assignment obtained for day $d = 2$ is transferred as an input to the data set of day $d = 3$ etc.

Table 1 reports aggregated computational results of CPLEX for the small and medium sized instances and each objective setting. The values reported in a row of this table are averages for the interday solutions of days 2–5 in an instance. The first three columns of the table show instance properties: problem size, number of variables, and number of constraints in the interday model. The next seven columns display results for the consistency setting where the reported values are *averages* of the 4 interday solutions obtained for each instance. They show the difference in team composition (column *X*), the absolute number of performed jobs (column *Z*), the relative share of performed jobs (column *Z*%), the total job completion time (column *F*), the total processing time of performed jobs (column *P*), the total travel time of all teams (column *G*), the number of active teams that process at least one job in the solution (column *T*), the consumed runtime in seconds (column *CPU*) and the optimality gap after the runtime limit of 3600 s (column GAP_{3600}) and after an extended runtime limit of 7200 s (column GAP_{7200}). The optimality gap expresses the deviation of the objective function achieved by the interday model to the lower bound value B reported by CPLEX as GAP= (Objective -LB)/LB. As the reported CPU times and GAPs are averages over 4 interday solutions, we might observe GAPs > 0 in combination with an average CPU time below the runtime limit, if only a subset of the interday models belonging to a test instance was solved to optimality. The results for the service setting are presented at the right of the table.

Based on Table 1, we see that the consistency setting avoids team reconfigurations ($X = 0.0$) as is expected for this configuration. Furthermore, we see that the number of performed jobs (*Z*) and the total job completion time (*F*) increase with larger instance size. We observe that about 70% of jobs are served for instances up to size 40 × 20. Note that even for those instances that were solved to optimality ($GAP_{3600} = 0\%$) there are unprocessed jobs, which is because either the number of

Table 1
Interday Model CPLEX.

Instance	Problem size		Consistency setting									Service setting								
$\lvert J_d\rvert \times \lvert M\rvert$	Variables	Constraints	*X*	*Z*	*Z*%	*F*	*P*	*G*	*T*	*CPU*	GAP_{3600}	GAP_{7200}	*X*	*Z*	*Z*%	*F*	*P*	*G*	*T*	*CPU*	GAP_{3600}	GAP_{7200}
8 × 4	764	416	0.0	4.5	56%	928	194	128	1.2	0.03	0%	0%	0.2	5.0	63%	1122	213	124	1.0	0.09	0%	0%
8 × 8	1584	864	0.0	5.0	63%	976	213	161	2.2	0.06	0%	0%	0.2	5.5	69%	1094	241	141	2.0	0.13	0%	0%
10 × 5	1285	745	0.0	7.2	72%	1778	323	163	1.2	1.15	0%	0%	0.0	7.2	72%	1778	323	163	1.2	2.44	0%	0%
10 × 10	2660	1540	0.0	7.2	72%	1442	323	185	2.2	0.97	0%	0%	0.0	7.2	72%	1442	323	185	2.2	0.82	0%	0%
15 × 7	3193	2072	0.0	10.5	70%	2235	462	241	2.5	901.28	1%	1%	0.8	11.2	75%	2783	495	219	2.0	1177.18	2%	2%
15 × 15	7065	4560	0.0	12.0	80%	2349	531	322	4.2	4.24	0%	0%	0.5	13.8	92%	2656	617	356	4.5	26.90	0%	0%
20 × 10	7070	4940	0.0	14.5	73%	3499	637	294	3.0	2035.69	2%	2%	1.0	15.2	76%	3712	678	315	3.0	2712.90	0%	0%
30 × 15	20355	15585	0.0	25.8	86%	5852	1119	531	5.2	3600.00	1%	1%	1.8	25.8	86%	5978	1120	549	5.0	3600.00	8%	3%
40 × 20	44140	35680	0.0	26.0	65%	5776	1110	624	8.5	3600.00	14%	14%	2.0	31.8	80%	8209	1369	669	5.2	3600.00	14%	14%
80 × 30	227630	202620	0.0	39.5	50%	9987	1715	880	8.8	3600.00	50%	39%	0.2	26.5	33%	6434	1159	624	8.5	3600.00	67%	59%

workers, the qualifications, or the given time windows prevent serving all jobs. For the largest instances considered here (80×30), the service level is even lower because this instance is far from being solved to optimality. As expected, the solution times increase drastically as the problem size increases. This is explained by the strong growth of the model size with increasing problem size. Hence, finding an optimum solution is possible only for about half of the considered instances. Considering GAP_{3600} and GAP_{7200}, there is no significant improvement of the solution quality, which means that further extending the runtime limit for CPLEX does not help solving the problem. This indicates that the time needed for solving large-scale problems would be unacceptably high.

The service setting allows to perform a few more jobs for most instances but also requires to transfer some employees between teams ($X > 0$). As this setting exploits the freedom of reconfiguring teams, the actual number of active teams in a solution (T) is often smaller in the service setting compared to the consistency setting. Looking at columns F, we see that the consistency and service setting can deliver identical solutions with same total completion time, see instances 10×5 and 10×10. However, for most instances solved to optimality, we observe higher total completion times in the service setting, which is because of the above mentioned tendency to serve more jobs under this prioritization of goals. The service setting is solved slightly slower than the consistency setting and, again, it cannot be solved for medium sized instances to optimality either. We even observe that the number of jobs performed for instance 80×30 is lower for the service setting than for the consistency setting despite the higher flexibility for forming teams.

We next evaluate the fix-and-optimize heuristic. To achieve a consistent comparison with CPLEX, the heuristic is initialized using the same teams as those determined by CPLEX for the first period of the planning horizon. For all subsequent periods, the heuristic generates teams using the construction heuristic from Section 4. For the large instances that were not considered for CPLEX, we initialize teams for the first day by constructing $0.5 \cdot |M|$ teams to which employees are uniformly distributed. Solutions of the subsequent days are then again determined one after the other using the fix-and-optimize heuristic. The parameters that control the heuristic are set as described in Appendix C. Results obtained by the heuristic are reported in Tables 2 and 3. In order to reveal the benefit of the improvement phases, the tables show the results obtained after each stage of the heuristic. Since the consistency setting forbids modifying the team compositions, we omit reporting X in Table 2 and we restrict the heuristic to the construction of initial solutions and the improvement by swapping jobs. For the service setting the heuristic conducts all its phases, see Table 3.

Table 2 reveals that the heuristic solutions serve the same number of jobs for instances with size up to 30×15 as the exact solutions produced by CPLEX. For instance 40×20 the heuristic performs slightly less jobs whereas for instance 80×30 it significantly outperforms the non-optimal CPLEX solution. Furthermore, the heuristic requires a significantly lower computational effort. Almost all problem instances are solved within a fraction of the time that is required by CPLEX. Even for the largest instance of size 500×100 the CPU time for each phase of the heuristic is just half a minute. As we enforce team consistency in this setting, the only improvement possible for the heuristic is to reduce completion times through swapping of jobs. This leads to a reduction of job completion times for 9 out of 15 instances. However, the extent of these reductions is relatively low, which is explained by the time windows given for the jobs.

Regarding the service setting, Table 3 clearly demonstrates that all improvement strategies contribute to better solution quality. Thereby, this effect is getting stronger with the increase of the problem size. Splitting and merging of teams allows to serve up to 12 further jobs compared to the initial solutions. Swapping of jobs contributes to the reduction of the total completion time for most of the instances. Eventually, randomized disturbance achieves to process up to 15 additional jobs for an instance. For the large sized instances, we attain an overall improvement of 7–23 additional jobs being served per instance. Looking at the largest instance tackled by CPLEX (80×30), we observe that the heuristic serves more than two times as many jobs as the non-optimal CPLEX solution but it requires merely 4 min of computational time. Looking at the largest instance of the instance set (500×100), we observe that the service setting allows the heuristic to serve about 10% more jobs compared to the consistency setting, which, however, requires about half an hour of computational time. Anyhow, even this solution time lies considerably below the preset runtime limit. Hence, the proposed heuristic appears as a powerful method for solving the considered optimization problem also for instances of large size.

5.3. Results for the intraday replanning

In this subsection, we evaluate the performance of the intraday model. We use as baseline schedules the solutions generated by the interday fix-and-optimize heuristic under the consistency setting for days $d = 2$ to $d = 5$. Thereby, qualification requirements of high-priority jobs are such that some of them require synchronization of several teams. As the main goal here is to schedule as many high-priority jobs as possible, we adopt the following parameters for evaluating the objective function: $\alpha = 100$, $\beta = 10$, $\theta = 1$, $\gamma = 0.001$. Finally, we set the arrival time of high-priority jobs to $\tau = 100$ minutes.

Table 4 shows the results obtained for different numbers of newly arriving high-priority jobs $|J^h| = 1$, 3 or 5. The values reported in each block represent averages of the four intraday solutions of days $d = 2$ to $d = 5$. The table reports for each instance and each number of jobs $|J^h|$ the number of actually inserted high-priority jobs (column H) and the number of team visits for performing these jobs (column Z^h). Here, if $H = Z^h$, each

Table 2
Interday Fix-and-Optimize Heuristic, consistency setting.

Instance	Initial solution				Swapping						
$\|J_d\| \times \|M\|$	Z	F	T	CPU	Z	Z%	F	P	G	T	CPU
8×4	4.5	928	1.2	0.05	4.5	56%	928	194	128	1.2	0.00
8×8	5.0	1045	2.0	0.04	5.0	63%	1045	213	151	2.0	0.00
10×5	7.2	1778	1.2	0.07	7.2	72%	1778	323	163	1.2	0.00
10×10	7.2	1574	2.2	0.16	7.2	72%	1574	323	199	2.2	0.08
15×7	10.5	2437	2.5	0.30	10.5	70%	2404	462	258	2.5	0.95
15×15	12.0	2546	4.5	0.07	12.0	80%	2546	531	342	4.5	0.00
20×10	14.5	3712	3.0	0.69	14.5	73%	3636	637	319	3.0	3.56
30×15	25.8	6396	5.2	1.27	25.8	86%	6250	1115	529	5.2	3.47
40×20	25.5	5960	8.0	0.90	25.5	64%	5960	1119	607	8.0	0.00
80×30	59.2	15192	9.0	3.35	59.2	74%	14957	2606	1134	9.0	1.19
100×40	53.8	13324	12.5	3.08	53.8	54%	13044	2350	1132	12.5	5.92
200×50	114.0	30050	16.8	9.80	114.0	57%	29836	5006	2090	16.8	7.71
300×80	187.0	49065	28.8	13.17	187.0	62%	48468	8283	3525	28.8	10.22
400×80	188.8	50297	26.0	20.91	188.8	47%	49064	8291	3308	26.0	32.38
500×100	257.8	69076	34.8	31.52	257.8	52%	67955	11302	4534	34.8	29.47

Table 3
Interday Fix-and-Optimize Heuristic, service setting.

Instance	Initial solution					Splitting and Merging					Swapping					Randomized disturbance					
$\|J_d\| \times \|M\|$	Z	F	T	CPU	X	Z	F	T	CPU	X	Z	F	T	CPU	X	Z	Z%	F	P	G	T	CPU
8 × 4	4.0	824	1.2	0.34	0.0	4.0	824	1.2	0.16	0.0	4.0	824	1.2	0.00	1.0	4.8	60%	962	205	133	1.2	11.85
8 × 8	5.2	1076	2.0	0.33	0.0	5.2	1076	2.0	0.16	0.0	5.2	1076	2.0	0.00	0.8	5.5	69%	1154	241	153	2.0	15.95
10 × 5	7.2	1778	1.2	0.30	0.0	7.2	1778	1.2	0.17	0.0	7.2	1778	1.2	0.00	0.0	7.2	72%	1778	323	163	1.2	18.16
10 × 10	7.2	1574	2.2	0.50	0.0	7.2	1574	2.2	0.16	0.0	7.2	1574	2.2	0.14	0.0	7.2	72%	1574	323	199	2.2	19.61
15 × 7	9.5	2359	2.0	0.70	0.2	10.8	2585	2.5	0.34	0.2	10.8	2549	2.5	1.02	2.2	11.2	75%	2732	493	308	2.8	33.62
15 × 15	13.2	2977	4.5	0.80	0.0	13.2	2977	4.5	0.16	0.0	13.2	2977	4.5	0.01	1.0	13.5	90%	2981	604	345	4.5	28.06
20 × 10	12.0	3059	2.8	0.90	2.8	13.2	3284	3.5	0.47	2.8	13.2	3208	3.5	3.83	4.0	15.0	75%	3681	661	359	4.0	50.13
30 × 15	25.2	6182	5.0	1.88	0.0	25.2	6182	5.0	0.16	0.0	25.2	5968	5.0	7.21	4.2	27.5	92%	6750	1217	551	5.0	75.49
40 × 20	30.5	7500	6.8	2.59	0.0	30.5	7500	6.8	0.16	0.0	30.5	7424	6.8	2.25	3.0	34.5	86%	8628	1499	679	6.2	77.16
80 × 30	59.2	15434	9.8	7.33	2.5	60.5	15767	10.2	0.26	2.5	60.5	15589	10.2	23.76	6.2	64.0	80%	16502	2844	1242	10.5	246.13
100 × 40	66.5	16777	12.5	7.46	5.0	79.0	20037	13.5	1.23	5.0	79.0	19848	13.5	109.04	6.8	85.8	86%	22125	3802	1692	13.5	422.65
200 × 50	120.5	32146	17.2	18.13	4.2	123.5	32924	17.8	0.83	4.2	123.5	32233	17.8	30.02	8.2	127.8	64%	33582	5597	2332	18.2	603.30
300 × 80	185.5	48824	28.5	24.79	6.2	196.8	51749	30.0	1.81	6.2	196.8	50855	30.0	104.74	9.5	204.0	68%	53115	9054	3869	30.2	1145.56
400 × 80	204.8	55139	28.2	35.36	7.2	216.8	58465	29.5	2.42	7.2	216.8	56750	29.5	202.86	11.2	227.2	57%	59244	9958	3930	30.5	1544.05
500 × 100	261.0	70860	36.0	51.09	10.0	273.8	74355	37.0	3.21	10.0	273.8	72376	37.0	124.92	11.8	278.8	56%	73441	12217	4965	38.5	1763.85

served job is visited by one team whereas for $H < Z^h$ team synchronization is part of the solution. Furthermore, this table shows the total deviation in start times of scheduled jobs (column *S*), the total job completion time (column *F*), the needed runtime (column *CPU*) and the optimality gap (column *GAP*).

As expected, we observe that the required computational time grows noticeably with increasing size of J^h. However, nearly all instances can be solved to optimality if only $|J^h| = 1$ or $|J^h| = 3$ jobs have to be inserted. Note that the intraday model schedules not necessarily all high-priority jobs but a feasible solution to the model can always be found here. For the setting with $|J^h| = 1$, the new incoming job can be served in almost all instances while a consistent insertion of all three jobs in setting $|J^h| = 3$ is possible for only 8 out of 15 instances. An example of where these jobs are inserted into the routes is provided in Appendix D. Further, we see that the computation time is at most 83 s, which is considerably below the runtime limit of 200 s. This indicates that the intraday model can cope successfully with up to 3 jobs arriving at the same time. Looking at columns *S*, we see that start times of already scheduled jobs are not changed ($S = 0$). This can be explained by the time windows for the jobs, which create time gaps in work schedules that are used for inserting new jobs. Eventually, if $|J^h| = 5$ jobs become available at a time, the heuristic manages to insert about 4–5 of them for most of the instances. Here, not all instances are solved to optimality ($GAP > 0$) within the runtime limit of 200 s. For the largest instances (400×80, 500×100) not even a feasible solution is obtained within the runtime limit of 200 s for at least one day of the time span $d = 2$ to $d = 5$. Note that *GAP* values of smaller instances are sometimes even higher than those of larger instances, compare 80×30 and 100×40 or 200×50 and 300×80. This indicates that the *GAPs* do not only depend on the problem size, which is because large parts of the routing and job-assignment variables are fixed now. Instead, problem difficulty is also determined by the particular skill vectors involved in an instance, the synchronization operations performed in a solution, and the opportunities for inserting high-priority jobs in the existing routes, which explain the somewhat erratic *GAPs* observed for medium and large instances in Table 4.

We now briefly analyze how the solution quality responds to differing values of job arrival time τ. For this purpose, we changed τ from its original value 100 to values 200, 300 and 400. Fig. 4 shows for three selected instances and $J^h = 5$ high-priority jobs the number of inserted jobs Z^h for the different settings of τ. For instance 20×10, we observe, as expected, that Z^h diminishes for high values of τ because inserting jobs might fail if they arrive too late. Surprisingly, for instance 80×30, we observe an increase of Z^h for increasing values of τ. The explanation for this counterintuitive result is as follows. While low values of τ indicate early job arrivals and, thus, offer potential for high-quality solutions, the resulting optimization problem is more difficult to solve than for higher values of τ that leave only few insertion possibilities. This effects that the solution under $\tau = 100$ is merely suboptimal (see *GAP* in Table 4) and even worse than the optimal solution achieved under the more restrictive $\tau \geq 300$. Hence, it can be easier for a MIP solver to cope with late arriving jobs where the solutions obtained within the runtime limit might even be better compared to solutions for earlier arriving jobs. For the third instance 200×50, we observe just another behavior as all jobs can be served even if they arrive relatively late, which shows that late arrivals not necessarily constitute a problem for the intraday rescheduling.

In order to reduce the computational effort of inserting multiple jobs at the same time, we conduct a further experiment where we insert the jobs J^h one by one. The number of iterations corresponds to the number of incoming jobs $|J^h|$. In each iteration, we run the model only for one high-priority job. If this job is taken up in the solution, we update the sets J_t for those teams that are involved in performing this job. The results of this experiment are summarized in Table 5. Comparing Tables 4 and 5 for $|J^h| = 1$, we observe identical results (as expected). For $|J^h| = 3$, iterative insertion of jobs yields the same solution quality in terms of the number of

Table 4
Intraday Model CPLEX.

Instance	$\|J^h\| = 1$						$\|J^h\| = 3$						$\|J^h\| = 5$					
$\|J\| \times \|M\|$	H	Z^h	S	F	CPU	GAP	H	Z^h	S	F	CPU	GAP	H	Z^h	S	F	CPU	GAP
8 × 4	0.5	0.5	0.0	1085	0.03	0%	2.2	2.2	0.0	1678	0.02	0%	3.8	3.8	0.0	2355	0.56	0%
8 × 8	1.0	1.5	0.0	1544	0.01	0%	2.5	3.0	0.0	1992	0.10	0%	4.0	4.8	0.0	2528	11.72	0%
10 × 5	0.5	0.5	0.0	1956	0.02	0%	1.5	1.5	0.0	2435	0.43	0%	1.8	1.8	7.2	2533	31.31	0%
10 × 10	1.0	1.5	0.0	1979	0.01	0%	2.8	3.2	0.0	2445	0.07	0%	4.8	5.5	0.0	3271	2.04	0%
15 × 7	1.0	1.5	0.0	2876	0.02	0%	2.8	3.2	0.0	3482	0.13	0%	4.2	4.8	0.0	4123	70.28	4%
15 × 15	1.0	1.5	0.0	2702	0.01	0%	3.0	3.5	0.0	3168	0.04	0%	5.0	5.8	0.0	3824	1.04	0%
20 × 10	0.8	1.0	0.0	3821	0.03	0%	2.2	2.5	0.0	4513	2.27	0%	3.2	3.5	0.0	4864	142.30	15%
30 × 15	1.0	1.5	0.0	6787	0.05	0%	3.0	3.8	0.0	7722	2.57	0%	5.0	6.2	0.0	8672	98.47	0%
40 × 20	1.0	2.0	0.0	6356	0.05	0%	3.0	5.0	0.0	7217	1.72	0%	5.0	8.8	0.0	8365	110.28	1%
80 × 30	1.0	1.5	0.0	15464	0.17	0%	3.0	3.8	0.0	16463	4.77	0%	4.5	6.2	0.0	17704	185.45	13%
100 × 40	1.0	1.8	0.0	13318	0.10	0%	3.0	5.2	0.0	14271	5.95	0%	5.0	9.8	0.0	15873	200.00	4%
200 × 50	1.0	1.5	0.0	29825	0.30	0%	3.0	3.5	0.0	30834	16.87	0%	3.2	4.0	10.0	30999	200.00	40%
300 × 80	1.0	1.2	0.0	48427	0.54	0%	3.0	3.5	0.0	49387	22.37	0%	4.2	5.8	25.2	50449	200.00	26%
400 × 80	1.0	1.2	0.0	49498	0.77	0%	2.5	3.0	0.0	50376	82.59	21%	–	–	–	–	–	–
500 × 100	1.0	1.2	0.0	66855	1.47	0%	3.0	3.5	0.0	67921	68.34	0%	–	–	–	–	–	–

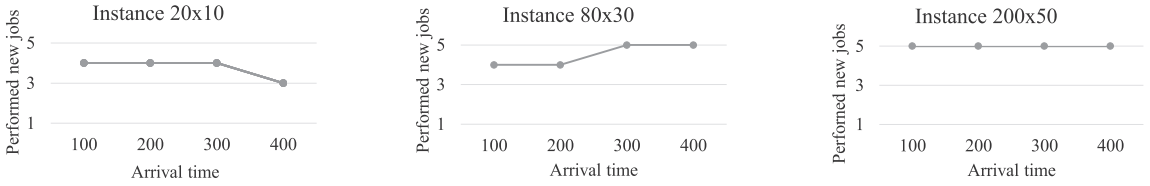

Fig. 4. Variation of arrival time.

Table 5
Intraday Model CPLEX (iteratively).

Instance	$\|J^h\| = 1$					$\|J^h\| = 3$					$\|J^h\| = 5$				
$\|J\| \times \|M\|$	H	Z^h	S	F	CPU	H	Z^h	S	F	CPU	H	Z^h	S	F	CPU
8 × 4	0.5	0.5	0.0	1085	0.03	2.2	2.2	0.0	1678	0.04	3.8	3.8	0.0	2355	0.37
8 × 8	1.0	1.5	0.0	1544	0.00	2.5	3.0	0.0	1994	0.06	4.0	4.8	0.0	2549	0.48
10 × 5	0.5	0.5	0.0	1956	0.02	1.5	1.5	0.0	2436	0.13	1.8	1.8	7.2	2539	0.58
10 × 10	1.0	1.5	0.0	1979	0.01	2.8	3.2	0.0	2447	0.04	4.8	5.5	0.0	3271	0.15
15 × 7	1.0	1.5	0.0	2876	0.01	2.8	3.2	0.0	3482	0.07	4.2	4.8	0.0	4127	1.04
15 × 15	1.0	1.5	0.0	2702	0.01	3.0	3.5	0.0	3234	0.04	5.0	5.8	0.0	3893	0.10
20 × 10	0.8	1.0	0.0	3821	0.03	2.2	2.5	0.0	4514	0.15	3.0	3.2	0.0	4840	1.58
30 × 15	1.0	1.5	0.0	6787	0.06	2.8	3.5	0.0	7590	0.32	4.8	6.0	0.0	8625	1.56
40 × 20	1.0	2.0	0.0	6356	0.04	2.8	4.2	0.0	6961	0.21	4.8	8.2	0.0	8060	0.54
80 × 30	1.0	1.5	0.0	15464	0.13	3.0	3.8	0.0	16472	0.93	4.2	6.0	0.0	17557	11.55
100 × 40	1.0	1.8	0.0	13318	0.09	3.0	5.2	0.0	14271	0.61	5.0	9.2	0.0	15676	2.82
200 × 50	1.0	1.5	0.0	29825	0.76	3.0	3.8	0.0	30934	3.84	4.8	7.2	0.0	32575	15.89
300 × 80	1.0	1.2	0.0	48427	0.44	3.0	3.5	0.0	49401	2.38	5.0	7.0	0.0	50900	13.36
400 × 80	1.0	1.2	0.0	49498	0.74	2.8	4.0	16.0	50859	8.94	4.2	6.8	16.0	52200	37.77
500 × 100	1.0	1.2	0.0	66855	0.88	3.0	3.5	0.0	67926	6.44	5.0	5.8	0.0	69068	28.47

performed jobs for most instances but requires much lower computational times of at most 9 s. For $|J^h| = 5$, all instances are solved feasibly now and within just a few seconds. The computational time does not exceed 1 min even for the larger instances. For the larger instances that could not be solved to optimality in the original intraday model, the iterative approach inserts additional jobs (200 × 50, 300 × 80, 400 × 80 and 500 × 100), avoids some of the synchronization processes (100 × 40) and reduces job completion times (100 × 40). For instances (20 × 10, 30 × 15, 40 ×20 and 80 × 30), the iterative insertion integrates one job less only for one data set (day $d = 2$ or day $d = 5$) compared to the original intraday model. To summarize, the obtained results demonstrate that the intraday model can be successfully used for updating even large size schedules. The iterative approach is furthermore suitable for integrating a large number of high-priority jobs within very short computational time.

6. Conclusions

In this paper, we have investigated the interday composition, routing and scheduling of multi-skilled workforce teams with a consistency requirement and an intraday rescheduling opportunity for serving jobs that become available on short notice. We have presented two interrelated optimization models. As large interday problems cannot be solved exactly, we have proposed and evaluated a fix-and-optimize heuristic that embeds a routing optimization model. Computational experiments show that the heuristic yields an effective solution approach. Test instances, for which exact solutions are generated by the CPLEX solver in up to 2 h, have been solved to similar quality within a few seconds only. Tests on larger instances confirm that the algorithm produces good solutions in a consistent and reliable way. Furthermore, the experiments indicate that the consistency requirement can be successfully integrated

into the planning. Finally, we demonstrate that the intraday rescheduling can be solved as an integer linear programming problem if only few jobs are to be inserted. If multiple jobs are to be inserted, a sequential insertion procedure can be applied. Its short computation times guarantee an almost immediate reaction, which makes the method suitable for practical application.

In spite of the achieved progress, future research may be conducted to assess team overqualification by cost, or to integrate within-day team splitting and synchronization into the interday planning. Furthermore, it could be interesting to consider hiring temporary workers as an alternative to the outsourcing of jobs. Also, while our interday planning and intraday rescheduling are suitable for settings with (highly) incomplete information about future jobs, it might be worth to develop a rolling horizon methodology for settings with (almost) complete information about future jobs. Such a methodology could solve the problem for several periods ahead and implement decisions for the current period only while resolving the multi-period problem in the next period with updated information and so on. In this way, team consistency might be improved by exploiting all available information.

Appendix A. Team Consistency

We describe here an alternative approach for measuring team consistency, which is based on Hamming distance. We introduce a new binary decision variable $y_{mm'}$ that takes value 1 if employees m and m' work together in the same team on the current day, 0 otherwise. The corresponding values of these variables from the previous day are denoted by parameters $y'_{mm'}$. Thus, if $y'_{mm'} = 1$ and $y_{mm'} = 1$, employees m and m' work together at both days no matter whether they stay in the same team of jointly switch to another team. If $y'_{mm'} = 0$ and $y_{mm'} = 0$, employees m and m' are not working together neither at the previous day nor at the current day. In both cases, no change takes place from the perspective of the pair m, m'. However, if $y'_{mm'} = 1$ and $y_{mm'} = 0$, or if $y'_{mm'} = 0$ and $y_{mm'} = 1$, team composition changed for pair m, m' as they no longer work in the same team or as they are newly assigned to a same team. The idea of the subsequent model is to establish team consistency by minimizing the number of such changing employee pairings. For this, we specify an auxiliary binary variable $Y_{mm'}$ that takes value 1 if $y'_{mm'} \neq y_{mm'}$, 0 otherwise.

As an example, consider the teaming of employees 1, 2 and 3 in Fig. 2. We have the following values for $y_{1,2} = y_{1,3} = y_{2,3} = 1$ for day $d = 1$ and, thus, $y'_{1,2} = y'_{1,3} = y'_{2,3} = 1$ for day $d = 2$. Furthermore, the pairing of employees at day $d = 2$ is reflected by $y_{1,2} = 1$ and $y_{1,3} = y_{2,3} = 0$, which leads to $Y_{1,2} = 0$ and $Y_{1,3} = Y_{2,3} = 1$. This can be continued similarly for employees in other teams and across teams.

The alternative consistency formulation can be incorporated into the interday model using the following terms:

$$\text{minimize}: \quad \sum_{m \in M} \sum_{m' \in M} Y_{mm'} \tag{43}$$

$$x_{mt} + x_{m't} - 1 \leq y_{mm'} \qquad \forall m, m' \in M, t \in T \tag{44}$$

$$2 - x_{mt} - \sum_{t' \in T \setminus \{t\}} x_{m't'} \geq y_{mm'} \qquad \forall m, m' \in M, t \in T \tag{45}$$

$$\sum_{t \in T} x_{mt} \geq y_{mm'} \qquad \forall m, m' \in M \tag{46}$$

$$y_{mm'} - y'_{mm'} \leq Y_{mm'} \qquad \forall m, m' \in M \tag{47}$$

$$y'_{mm'} - y_{mm'} \leq Y_{mm'} \qquad \forall m, m' \in M \tag{48}$$

$$x_{mt}, y_{mm'}, Y_{mm'} \in \{0, 1\} \qquad \forall m, m' \in M, t \in T \tag{49}$$

Objective (42) maximizes team consistency by minimizing changes in employee pairing. This objective can be combined with the other objectives as done in (1). Constraints (43) enforce $y_{mm'} = 1$ if employees m and m' are in the same team. Constraints (44) set $y_{mm'} = 0$ if employees m and m' are assigned to different teams. If employee m is not assigned to any team, Constraints (45) set $y_{mm'} = 0$ for all m'. Constraints (46)–(47) enforce $Y_{mm'} = 1$ if $y'_{mm'} \neq y_{mm'}$.

Corresponding CPLEX results for the interday model with the alternative consistency measure are reported in Table 6. Compared with the results in Table 1, it can be seen that both consistency measures deliver identical solutions regarding the number of performed jobs and the total job completion time for all those instances that are solved to optimality. This finding holds for both, the consistency setting and the service setting. In the consistency setting, this is because team consistency is of utmost importance, i.e., total changes in teams are always $X = 0$ and $Y = 0$ in Table 1 and 6 Team Consistency, respectively. In the service setting, maximizing the number of performed jobs is much more important than minimizing team consistency, which is why the used consistency measure plays no significant role. However, as the alternative measure requires adding new decision variables and constraints, solvability of the problem deteriorates, which leads to larger gaps and runtimes for the largest instances considered here. Hence, the alternative formulation of team consistency does not seem to provide an advantage.

Table 6
Interday Model CPLEX (alternative consistency formulation).

Instance	Problem size		Consistency setting						Service setting					
$\|J_d\| \times \|M\|$	Variables	Constraints	Y	Z	F	T	CPU	GAP_{3600}	Y	Z	F	T	CPU	GAP_{3600}
8 × 4	876	444	0.0	4.5	928	1.2	0.08	0%	4.5	5.0	1122	1.0	0.23	0%
8 × 8	2544	984	0.0	5.0	976	2.2	0.22	0%	2.0	5.5	1094	2.0	1.93	0%
10 × 5	1510	790	0.0	7.2	1778	1.2	10.26	0%	0.0	7.2	1778	1.2	17.73	0%
10 × 10	4560	1730	0.0	7.2	1442	2.2	63.91	0%	0.0	7.2	1442	2.2	52.39	0%
15 × 7	3830	2163	0.0	10.5	2236	2.5	960.97	1%	3.8	11.2	2877	2.0	2421.95	2%
15 × 15	13590	4995	0.0	12.0	2349	4.5	1803.84	1%	4.0	13.8	2659	4.5	3600.00	1%
20 × 10	8970	5130	0.0	14.5	3504	3.0	2724.25	2%	1.5	14.5	3525	3.0	3600.00	4%
30 × 15	26880	16020	0.0	25.8	5991	5.0	3600.00	1%	0.5	25.8	5997	5.2	3600.00	8%
40 × 20	59740	36460	0.0	25.2	5720	7.8	3600.00	20%	6.0	22.8	5867	5.8	3600.00	39%
80 × 30	280730	204390	0.0	11.2	2580	3.5	3600.00	86%	0.0	11.8	2700	4.2	3600.00	85%

Table 7
Comparison of Different α and β Weights in the Objective.

	Instance	Service setting (original measure)						Service setting (alternative measure)					
α, β	$\|J_d\| \times \|M\|$	X	Z	F	T	CPU	GAP_{3600}	Y	Z	F	T	CPU	GAP_{3600}
	8 × 4	0.0	4.5	928	1.2	0.04	0%	0.0	4.5	928	1.2	0.09	0%
$\alpha = 1$	8 × 8	0.0	5.0	976	2.2	0.09	0%	0.0	5.0	976	2.2	0.69	0%
$\beta = 1$	15 × 7	0.0	10.5	2235	2.5	901.44	2%	0.0	10.5	2236	2.5	1032.68	0%
	15 × 15	0.2	13.2	2682	4.0	26.06	0%	0.0	12.0	2349	4.5	3600.00	5%
	8 × 4	0.2	5.0	1122	1.0	0.08	0%	0.0	4.5	928	1.2	0.12	0%
$\alpha = 1$	8 × 8	0.2	5.5	1094	2.0	0.12	0%	0.0	5.0	976	2.2	0.62	0%
$\beta = 2$	15 × 7	0.5	11.0	2449	2.5	947.51	2%	0.0	10.5	2236	2.5	1133.05	0%
	15 × 15	0.5	13.8	2656	4.5	29.85	0%	0.0	12.0	2349	4.5	3600.00	8%
	8 × 4	0.0	4.5	928	1.2	0.04	0%	0.0	4.5	928	1.2	0.11	0%
$\alpha = 2$	8 × 8	0.0	5.0	976	2.2	0.14	0%	0.0	5.0	976	2.2	0.26	0%
$\beta = 1$	15 × 7	0.0	10.5	2235	2.5	457.14	0%	0.0	10.5	2236	2.5	1015.08	0%
	15 × 15	0.2	13.2	2682	4.0	17.97	0%	0.0	12.0	2349	4.5	3600.00	1%

Furthermore, we analyze how the solutions react to variations of weights α and β in the objective function. Table 7 shows results of four instances under both consistency measures and varied objective function weights. The selected instances have solutions with team changes ($X > 0$ and $Y > 0$) in the service setting of both consistency measures, see Table 1 and 6 Weights α and β are chosen such that the substitution ratio of 'team changes' against 'additionally performed jobs' is 1:1, 1:2 and 2:1. For the original consistency measure from Section 3.3, we observe that relatively small changes of objective weights can effect an outcome in the solutions. Looking at column Z, we see that a marginal increase of β results in an increase of the number of performed jobs for all four instances. Moreover, these results are almost identical to the results obtained for the service setting with the much higher weight $\beta = 100$ in Table 1. In contrast, increasing α keeps the solutions stable with almost no changes in the teams, which indicates that also team consistency can be controlled through relatively small variations of weights.

For the alternative consistency measure from Appendix A, the results remain constant for all three combinations of weights. Thereby, we observe no changes in the team structure and relatively low numbers of performed jobs. These results coincide completely with the results achieved under the consistency setting, see Table 1. This can be explained by the fact that relative small changes in the team composition lead to a quite large number of changes of $y_{mm'}$ variables. Hence, to generate the same results, the alternative consistency measure requires a considerably higher difference between the two objective coefficients. In other words, it seems that the original measure is somewhat easier to control if one seeks for a tradeoff of team consistency and number of performed jobs.

Appendix B. Pseudocodes for Fix-and-Optimize Heuristic

Algorithm 1: Interday Planning: Initial Solution

Input: set of teams T, job set J_d

$J_t \leftarrow \{0\}\ \forall t \in T$
generate sets $J_t^{qual}\ \forall t \in T$
sort jobs $j \in J_t^{qual}$ in ascending order of $\mu_{tj}\ \forall t \in T$
while $T \neq \varnothing$ **do**
  $t \leftarrow arg\ min_{t' \in T}\{|J_{t'}^{qual}|\}$ ▷ select team with fewest jobs in set J_t^{qual}
  while $|J_t| < \lambda^{max}$ and $J_t^{qual} \neq \varnothing$ **do**
    $J^{temp} \leftarrow J_t \cup (\lambda^{max} - |J_t|)$ first jobs from set J_t^{qual} ▷ build set of temporary jobs
    $J_t^{qual} \leftarrow J_t^{qual} \backslash J^{temp}$ ▷ remove new considered jobs from J_t^{qual}
    solve routing problem (36)-(40) for team t
    update set J_t according to jobs processed in obtained route
  end while
  $J_{t'}^{qual} \leftarrow J_{t'}^{qual} \backslash J_t\ \forall t' \in T, t' \neq t$ ▷ remove assigned jobs from J_t^{qual} of all not yet examined teams
  $T \leftarrow T \backslash \{t\}$ ▷ remove considered team t from the list T
end while

Algorithm 2: Interday Planning: Splitting and Merging of Teams

Input: solution of Algorithm 1, set of still unprocessed jobs J^{un}

create free team list T^F
sort jobs $j \in J^{un}$ in descending order of difficulty factor ϑ_j
for $j \in J^{un}$ **do**
  if aggregated teams in T^F are qualified for job j **then**
    $t^M \leftarrow \varnothing$
    while t^M is not qualified for job j **do** ▷ find teams to merge
      $t \leftarrow arg\ min_{t' \in T^F}\{\mathcal{E}_{jt'}\}$ ▷ choose team t with minimum skill gap factor
      assign all employees from team t to team t^M ▷ extend merged team
      $T^F \leftarrow T^F \backslash \{t\}$ ▷ remove the considered team from free team list
    end while
    create $J_{t^M}^{qual}$ for team t^M ▷ identify the set of jobs for which the merged team is qualified
    while $|J_{t^M}| < \lambda^{max}$ **and** $J_{t^M}^{qual} \neq \varnothing$ **do**
      $J^{temp} \leftarrow J_{t^M} \cup (\lambda^{max} - |J_{t^M}|)$ first jobs from set $J_{t^M}^{qual}$ ▷ build set of temporary jobs
      $J_{t^M}^{qual} \leftarrow J_{t^M}^{qual} \backslash J^{temp}$ ▷ remove new considered jobs from $J_{t^M}^{qual}$
      solve routing problem (36)-(40) for team t^M
      update set J_{t^M} according to obtained route
    end while
    $J^{un} \leftarrow J^{un} \backslash J_{t^M}$ ▷ remove assigned jobs from J^{un}
  end if
end for

Algorithm 3: Interday Planning: Swapping of Jobs

Input: solution of Algorithm 2

initialize set of teams for job swapping $T^S \leftarrow \{t | t \in T, J_t \neq \varnothing\}$
for $(t, t') \in T^S \times T^S$ with $t < t'$ **do**
  create sets $J^S_{t',t}$ **and** $J^S_{t,t'}$ ▷ job sets that can be moved between teams t and t'
  if $J^S_{t',t} \neq \varnothing$ **and** $J^S_{t,t'} \neq \varnothing$ **then**
    for $(j^1, j^2) \in J^S_{t',t} \times J^S_{t,t'}$ **do**
      solve routing problem (36)-(40) for team t with job set $J^{temp} \leftarrow J_t \backslash \{j^2\} \cup \{j^1\}$
      if j^1 is served in the obtained route **then**
        solve routing problem (36)-(40) for team t' with job set $J^{temp'} \leftarrow J_{t'} \backslash \{j^1\} \cup \{j^2\}$
        if j^2 is served in the obtained route **and** total job completion time is reduced **then**
          save new schedules for teams t and t'
          $J_t \leftarrow J^{temp}$, $J_{t'} \leftarrow J^{temp'}$
        end if
      else
        continue with next j^1
      end if
    end for
  end if
end for

Algorithm 4: Interday Planning: Randomized Disturbance

Input: employee assignment on previous day x'_{mt}, number of iterations I, number of employees N to interchange

$obj^{best} \leftarrow \infty$
for $i = 1$ **to** I **do**
  exchange N employee assignments in x'_{mt}
  generate an initial solution (Algorithm 1)
  improve by splitting and merging of teams (Algorithm 2)
  calculate objective value obj of current solution
  if $obj < obj^{best}$ **then**
    save new solution
    $obj^{best} \leftarrow obj$
  end if
end for
improve by swapping of jobs (Algorithm 3)

Appendix C. Selection of Parameters for Fix-and-Optimize Heuristic

In order to determine the best trade-off between efficiency and solution quality, we carry out preliminary analyses for setting the parameters of the heuristic. First, we vary the maximal number of jobs λ^{max} transferred to the routing subproblem as well as the time limit for solving this subproblem. Fig. 5 demonstrates the impact of different parameter settings on the number of performed jobs for three selected instances. On the one hand, larger λ^{max} values enable allocating more jobs in each subproblem and improving the solution quality. On the other hand, if the time limit is set too low, the subproblem cannot be solved exactly, which leads to a deterioration of the solution quality. An increased time limit can be reasonable up to a saturation point beyond which there is no further improvement obtained. Furthermore, the time limit for the subproblem has a major influence on the total CPU time needed for solving an instance, see Fig. 6. The figure shows that the total CPU time grows noticeably and more than doubles with the considered increase of the subproblem time limit. From the results in Figs. 5 and 6, we find that a good compromise can be reached by setting $\lambda^{max} = 8$ and a runtime limit of 0.5 s, which we apply in all further experiments.

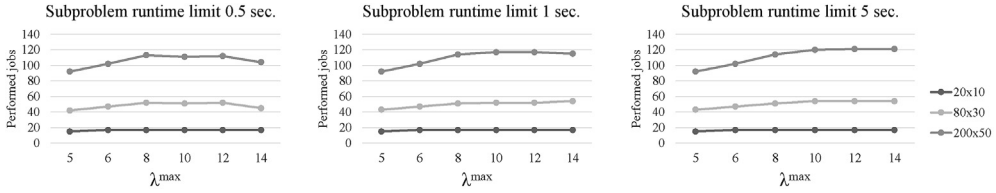

Fig. 5. Variation of λ^{max} and subproblem runtime limit

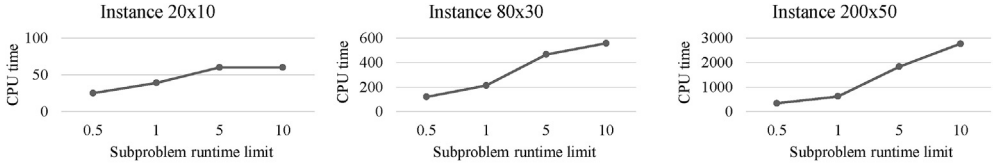

Fig. 6. Impact of subproblem time limit on total CPU time

We also analyze how the solution quality responds to the number of interchanged employees N in the randomized disturbance phase of the heuristic. Note that this parameter is irrelevant for the consistency setting where team consistency is of utmost importance. For this reason, we only conduct this experiment for the service setting. Fig. 7 shows for three instances the number of performed jobs in the final solutions if the heuristic uses Algorithm 4 with $N = 2, 3 \ldots 10$ employees to be exchanged. Unfortunately, the fluctuations shown in the figure do not provide guidance on how to set N. The parameter appears to be irrelevant for the small sized instance, whereas for the medium and the large sized instance an increase of N does not necessarily contribute to the improvement of solution quality. As we aim at maximizing the number of assigned jobs while minimizing modifications of the team structure, we choose a relatively low value of $N = 3$. We also conducted experiments for setting parameter I but omit them here for reasons of brevity. This parameter is set to $I = 30$.

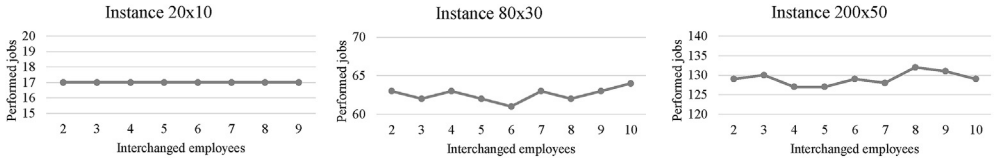

Fig. 7. Variation of number of interchanged employees

Appendix D. Insertion of High-Priority Jobs

Table 8 illustrates where, in the routes, new jobs are inserted at the example of the intraday solutions for instance 8 × 8. The first column of the table reports the initial routes and the start times in the baseline schedule with two teams performing 4 out of 8 jobs. The remaining three columns report the corresponding solutions for settings $|J^h| = 1$, $|J^h| = 3$ and $|J^h| = 5$ where 1, 3 and 5 high-priority jobs are available. The high-priority jobs are indexed 9 to 13. They can be all inserted in these solutions. We observe that most high-priority jobs are inserted at the end of the baseline-routes. An exception is job 10, which finds a suitable time gap within the route of team 1, where the subsequently served job 6 keeps its original start time due to its time window. Note that job 13 requires synchronization of two teams. Thereby, the difference in start times $348 - 318 = 30$ respects the maximal temporal distance δ_{max}.

Table 8
Insertion of High-Priority Jobs for Instance 8 × 8.

	Baseline schedule		$\|J^h\| = 1$		$\|J^h\| = 3$		$\|J^h\| = 5$	
	Team 1	Team 2	Team 1	Team 2	Team 1	Team 2	Team 1	Team 2
route	0-1-8-6-0	0-5-0	0-1-8-6-0	0-5-9-0	0-1-8-10-6-0	0-5-11-9-0	0-1-8-10-6-13-0	0-5-9-11-12-13-0
start times	0-94-161-267	0–55	0-94-161-267	0-55-118	0-94-161-217-267	0-55-114-165	0-94-161-217-267-348	0-55-118-164-205-318

References

Anoshkina, Y., Meisel, F., 2019. Technician teaming and routing with service-, cost- and fairness-objectives. Comput. Ind. Eng. 135, 868–880.

Balas, E., 1989. The prize collecting traveling salesman problem. Networks 19, 621–636.

Borenstein, Y., Shah, N., Tsang, E., Dorne, R., Alsheddy, A., 2010. On the partitioning of dynamic workforce scheduling problem. J. Sched. 13, 411–425.

Cappanera, P., Gouveia, L., Scutellà, M.G., 2013. Models and valid inequalities to asymmetric skill-based routing problems. EURO J. Transport. Logistic. 2, 29–55.

Chen, X., Thomas, B.W., Hewitt, M., 2016. The technician routing problem with experience-based service times. Omega 61, 49–61.

Chen, X., Thomas, B.W., Hewitt, M., 2017. Multi-period technician scheduling with experience-based service times and stochastic customers. Comput. Oper. Res. 82, 1–17.

Coelho, L.C., Cordeau, J.-F., Laporte, G., 2012. Consistency in multi-vehicle inventory routing. Transport. Res. C Emerg. Technol. 24, 270–287.

Cordeau, J.-F., Laporte, G., Pasin, F., Ropke, S., 2010. Scheduling technicians and tasks in a telecommunication company. Eur. J. Oper. Res. 243, 1–16.

De Bruecker, P., Van den Bergh, J., Beliën, J., Demeulemeester, E., 2015. Workforce planning incorporating skills: state of the art. J. Sched. 13, 393–409.

Estellon, B., Gardi, F., Nouioua, K., 2009. High-performance local search for task scheduling with human resource allocation. Lect. Notes Comput. Sci. 5752, 1–15.

Feillet, D., Garaix, T., Lehuédé, F., Péton, O., Quadri, D., 2014. A new consistent vehicle routing problem for the transportation of people with disabilities. Networks 63, 211–224.
Firat, M., Briskorn, D., Laugier, A., 2016. A branch-and-bound algorithm for stable workforce assignment with hierarchical skills. Eur. J. Oper. Res. 251, 676–685.
Firat, M., Hurkens, C.A.J., 2012. An improved MIP-based approach for a multi-skill workforce scheduling problem. J. Sched. 15, 363–380.
Gröer, C., Golden, B., Wasil, E., 2009. The consistent vehicle routing problem. Manuf. Serv. Oper. Manag. 11, 630–643.
Hashimoto, H., Boussier, S., Vasquez, M., Wilbaut, C., 2011. A GRASP-based approach for technicians and interventions scheduling for telecommunications. Ann. Oper. Res. 183, 143–161.
Ho, S.C., Leung, J.M.Y., 2010. Solving a manpower scheduling problem for airline catering using metaheuristics. Eur. J. Oper. Res. 202, 903–921.
Hurkens, C.A.J., 2009. Incorporating the strength of MIP modeling in schedule construction. Oper. Res. 43, 409–420.
Kalisch, B.J., Begeny, S., Anderson, C., 2008. The effect of consistent nursing shifts on teamwork and continuity of care. J. Nursing. Admin. Ops. Res. 38, 132–137.
Kasirzadeh, A., Saddoune, M., Soumis, F., 2017. Airline crew scheduling: models, algorithms, and data sets. EURO J. Transport. Logistic. 6, 111–137.
Khalfay, A., Crispin, A., Crockett, K., 2017. Applying the intelligent decision heuristic to solve large scale technician and task scheduling problems. Int. Con. Intell. Decision. Technol. 71–81.
Kovacs, A.A., Parragh, S.N., Doerner, K.F., Hartl, R., 2012. Adaptive large neighborhood search for service technician routing and scheduling problems. J. Sched. 15, 579–600.
Kovacs, A.A., Golden, B.L., Hartl, R., Parragh, S.N., 2014. Vehicle routing problems in which consistency considerations are important: a survey. Networks 64, 192–213.
Maenhout, B., Vanhoucke, M., 2018. A perturbation matheuristic for the integrated personnel shift and task re-scheduling problem. Eur. J. Oper. Res. 269, 806–823.
Mathlouthi, I., Gendreau, M., Potvin, J.-Y., 2018. Mixed integer programming for a multi-attribute technicians routing and scheduling problem. Info. Sys. Ops. Res. 56, 33–49.
Paraskevopoulos, D.C., Laporte, G., Repoussis, P.P., Tarantilis, C.D., 2016. Resource constrained routing and scheduling: review and research prospect. Eur. J. Oper. Res. 263, 737–754.
Patric, J., Puterman, M.L., Queyranne, M., 2017. Dynamic multipriority scheduling for a diagnostic resource. Oper. Res. 56, 1507–1525.
Petrakis, I., Hass, C., Bichler, M., 2012. On the impact of real-time information on field service scheduling. Decis. Support Syst. 53, 282–293.
Pillac, V., Gueret, C., Medaglia, 2012. On the dynamic technician routing and scheduling problem. In: ODYSSEUS 2012 - 5th International Workshop on Freight Transportation and Logistics, May 2012, Mikonos, Greece 194, pp. 1–11, 16.
ROADEF Challenge, 2007. Technicians and Interventions Scheduling for Telecommunications. http://www.roadef.org/challenge/2007/en/index.php (visited on 01.06.2019).
Russell, D., Rosati, R.J., Rosenfeld, P., Marren, J.M., 2011. Continuity in home health care: is consistency in nursing personnel associated with better patient outcomes? J. Healthc. Qual. 33, 33–39.
Siferd, S.P., Benton, W.C., 1994. A decision modes for shift scheduling of nurses. Eur. J. Oper. Res. 74, 519–527.
Smilowitz, K., Nowak, M., Jiang, T., 2013. Workforce management in periodic delivery operations. Transport. Sci. 47, 214–230.
Van Eck, M.L., Firat, M., Nuijten, W.P.M., Sidorova, N., van der Aalst, W.M.P., 2017. Human performance-aware scheduling and routing of a multi-skilled workforce. Complex. Sys. Infor. Model. Quart. 12, 1–21.
Zamorano, E., Stolletz, R., 2016. Branch-and-price approaches for the multiperiod technician routing and scheduling problem. Eur. J. Oper. Res. 1, 1–14.