Integration of Workforce Teaming and Routing

Inaugural-Dissertation

zur Erlangung des akademischen Grades eines Doktors

der Wirtschafts- und Sozialwissenschaften

der Wirtschafts- und Sozialwissenschaftlichen Fakultät

der Christian-Albrechts-Universität zu Kiel

vorgelegt von

M. Sc. Yulia Anoshkina

aus Kaluga

Kiel, 07.12.2020

Erstbegutachtung:	Prof. Dr. Frank Meisel
Zweitbegutachtung:	Prof. Dr. Catherine Cleophas
Tag der mündlichen Prüfung:	22.03.2021

Contents

List of Figures

List of Tables

1 Synopsis

1.1 Introduction and Research Gap

An intelligent personnel planning is a key prerequisite to achieve greater productivity and competitive strength in many industries. It enables to bridge reliably future personnel gaps, to reduce overstaffing and understaffing as well as to increase employee's efficiency and satisfaction. Therefore, a well-planned workforce scheduling has become a top priority for many companies, see e.g. Ernst et al. (2004), Pinedo et al. (2015). Its importance continues to grow especially in service industries where maintaining growth in a highly competitive market requires that companies use their expensive and limited personnel resources effectively. However, to capture practical realities is not a trivial matter. Work rules and legal regulations as well as current market developments impose consistently additional constraints making staffing and scheduling more and more complex. This motivates the development of sophisticated scheduling methods. For instance, one of the big challenges are job's complexity and large number of involved employees. In view of this complexity, jobs can often be carried out only by a group of specialists. The situation is complicated by the fact that specialists can possess skills in multiple domains. Skills can in turn be characterized by several hierarchical levels according to expert knowledge, experience, social competencies or other performance attributes. The fashion in which a team is configured is a key factor that influences the outcome of the work. Therefore, implementing multi-skill teams requires companies to rethink their organization on multiple levels. In the past decade, research on team effectiveness has been spurred as team work has become central in organizations of different types. For instance, Shelton et al. (2010) analyzed the group potency that was defined as the success with which groups accomplish their tasks. One of the main findings of this study is that perceived organizational support has a positive influence on the team working process. The latter can be promoted, for instance, by increasing employee empowerment or team building activities. Leggat (2007) analyzed critical teamwork competencies for health service managers. Mello and Ruckes (2006) investigated the factors that are crucial for a balanced team composition. The authors emphasized that heterogeneous groups demonstrate the capability to reach better decisions due to a wider range of experiences and opinions and can outperform homogeneous ones in highly uncertain situations. Furthermore, numerous psychological and empirical studies show that well-balanced teams are likely to experience more interpersonal compatibility and agreements about their tasks and team processes, see e.g. Cassera et al. (2009), Kalisch et al. (2008), Ilgen et al. (2005), Mathieu et al. (2008). In the context of scheduling, the question of the team “optimal„ composition refers not only to the distribution of knowledge or personal attributes across members of a team but also to the nature of the tasks that are to be processed. Due to limited personal resources, teams often cannot be split within a planning period but can get assigned multiple jobs of different complexity to process them consecutively. From this perspective, another analytic challenge that has to be addressed is an efficient cross-functional task distribution across jobs as well as of jobs among the teams. To this end, a strict service orientation may also require that services are provided at customer locations. Hence, routing of teams is another essential feature of workforce teaming and job allocation.

This dissertation offers innovative analytic and methodological approaches to a workforce scheduling problem that is defined as a combination of three subproblems each of which represents a complex and highly constrained optimization problem itself: teaming of multi-skilled employees, assignment of jobs to the created teams and team routing across customer locations. The problem is referred to in the following as RSPMST (routing and scheduling problem of multi-skilled teams). The problem is of practical relevance in a wide range of organizations. For instance, scheduling of multi-skilled technicians in the telecommunication industry can be regarded as a representative optimization topic, see Cordeau et al. (2010). Further domains of such companies include maintenance, construction, airline or health care sector, see e.g. Kovacs et al. (2012), Firat and Hurkens (2012), Ho and Leung (2010), Dohn et al. (2009), Castillo-Salazar et al. (2014). Despite its high practical relevance, the discussed problem represents a relatively new and unexplored research field. Although, there have been first attempts to formalize basic planning concepts (see e.g. Cordeau et al. (2010), Estellon et al. (2009), Hurkens (2009)), the incorporation of real life aspects in designing workforce scheduling systems is still limited and challenging. Including elements such as employee-oriented scheduling, dynamic environments, data uncertainty or skill-based estimation of processing times could offer additional insights into the problem. Therefore, following research questions are investigated in the essays that belong to this thesis:

1. Under what objectives should RSPMST be optimized? Can the complexity of the respective problem be reduced by decomposition techniques?

2. How can companies manage demand and capacity in a dynamic environment while maintaining employee engagement?

3. How can schedule reliability be guaranteed in the presence of data uncertainty?

4. How does cross-functional task distribution influence job processing times?

The above mentioned aspects describe the scope of this thesis, which aims to develop new planning concepts and couple them with advanced solution methods. From an optimization point of view, this work is concerned with linear programming techniques and metaheuristics that allow to obtain good quality solution for large-scale problems in comparatively less computational time compared to exact optimization methods.

The remainder of this chapter is structured as follows. Section 1.2 provides a brief literature review. Section 1.3 outlines the structure of this thesis and highlights the individual contributions. Section 1.4 presents extended abstracts for the involved essays that explain the approached research gap, the developed models and solution methods and contain a brief description of obtained results individually for each essay. Finally, potential areas for future research are discussed in Section 1.5.

1.2 Literature Overview

As personnel scheduling represents a multidisciplinary research area, there is a vast amount of literature where the problem is considered from different perspectives. Therefore, this review will be limited to the most relevant cross-border studies devoted to multi-skill scheduling as well as to routing and scheduling of technicians.

Since the beginning of the 1990s, a series of studies were conducted to reveal the economical and psychological benefits and challenges of multi-skilling, see e.g. Cordery (1989), Bergman (1994), McCune (1994). A large stream of research designed optimization models and conducted computational experiments. For instance, Campbell (1999) introduced and evaluated one of the first non-linear optimization models for allocating cross-trained employees where employees qualifications were defined by parameters ranging between zero and one. The main goal was to maximize the utility associated with the assignment of employees to different departments. The study demonstrated that already a small amount of cross-training can offer substantial benefits in terms of cross-utilization. Li and Li (2000) considered a problem from another context. Specifically, the authors analyzed scheduling flexibility to possible demand fluctuations and introduced a planning model where multi-skilling was designed by three parameters: employee's ability to substitute for different skill categories, productivity coefficients for each particular staff category and relative efficiency for a substitution of staff. It was shown that integrating the flexibility aspect into the personnel scheduling provides a capacity cushion. However, this does not necessarily result in a significant cost reduction.

A further line of research established the concept of multi-dimension skill representation in which employees are classified into different categories based on the level of knowledge, experience or training reached in each skill domain. This modeling approach found application in a number of studies devoted to a tour scheduling problem that involves allocating of shifts to the individual day work schedules of employees as well as assignment of activities to shifts. To incorporate the concept into a linear optimization framework, some of these studies (Eiselt and Marianov (2008), Cuevas et al. (2016), Gérard et al. (2016)) parametrized the set of feasible activities for each employee while other approaches (Al-Yakoob and Sherali (2007), Eitzen and Panton (2004)) conversely derived the set of qualified employees for each skill category. In a recent contribution (Altner et al. (2019)), the problem of assigning multi-skilled employees was extended by training decisions that have to be taken under demand uncertainty. The study demonstrated that proposed stochastic programming approach is superior to a deterministic one and can significantly reduce personal costs.

A similar modeling principal found its application in production environments where a set of workers has to be allocated to a set of production lines and machines. For instance, Park (1991) described the skill degree of workers by a two-dimensional efficiency matrix. Piya and Al-Hinai (2014) assigned workers to different hierarchy levels depending on their capability to operate different machines and the speed to perform certain operations. Shahnazari-Shahrezaei et al. (2013) investigated a multi-skilled manpower scheduling problem where employees were grouped in two specializations and specialization were discretized into three skill levels. Thereby, each employee could perform operations at his real or any lower skill level. A special case represents scheduling of temporary workers proposed by Techawiboonwong et al. (2006) where only two types of workstations demanding skilled and unskilled workers are considered. Further applications can be found e.g. in Seckiner et al. (2007), Narashiman (2000), Pastor and Corominas (2010), Ozguven and Sungur (2013). Parallels can also be drawn to project scheduling where a set of employees is required to execute project activities based on skills required for the tasks involved in the project, see e.g. Bellenguez-Morineau (2008), Kazemipoor et al. (2013), Ahmadpour and Ghezavatil (2019).

Another issue that deserved more attention and appeared highly promising was scheduling of jobs for multi-skilled teams that was first presented as a topic of ROADEF optimization challenge organized in 2007, see ROADEF (2007). An introductory study in this field was given in Estellon et al. (2009), Hurkens (2009), Cordeau et al. (2010) and Hashimoto et al. (2011). The authors presented alternative basic concepts for forming teams and assigning of jobs based on multi-dimensional qualification requirement matrices. The contribution of these works was the introduction of teaming decisions into operations management. One of the most inspiring works in the related domain represents the study of Kovacs et al. (2012) that first incorporated the need to route the teams from customer location to customer location. In the scheduling literature, routing itself does not represent a novel planning problem on its own. However, most research in the routing domain refers usually to scenarios where only one person is required to serve a job. Examples can be found in the home health care sector (Kergosien et al. (2009), Bertels and Fahle (2006), Everbon et al. (2006)), the telecommunication industry (Tsang and Voudouris (1997), Xu and Chiu (2001)), port manpower planning (Lim et al. (2004)), and the repair and the maintenance sector (Cortés et al. (2014), Pillac et al. (2013)). Teaming decisions are not part of these studies.

This literature review reveals that the research on incorporating routing and scheduling into the composition of multi-skilled teams is very limited so far. Therefore, RSPMST represents a relatively unexplored field that offers still a broad spectrum of research topics. To bridge this gap, the thesis contributes to the development of models and methods that provide a deeper insight into the problem.

1.3 Overview of Contributions

This thesis aims at researching challenges in the area of composing and scheduling of multi-skilled groups of employees as is faced by service-oriented companies. It provides useful models and solution techniques that address several so-far uninvestigated aspects of the problem. The submitted work is a cumulative dissertation that contains a collection of four essays. Each essay will cover one or more relevant practical aspects for successful workforce management and involves planning concepts, the formulation of optimization models, the design of algorithms as well as computational analyses of the involved decisions. More precisely, Essay 1 analyses the involved problem under three different optimization objectives: improvement of service quality, reduction of labor cost and fairness of workload distribution. In doing so, it contributes to the answering of research question 1 from Section 1.1. The service quality is associated with the total job completion times. The labor cost are represented by the total employee working time. A scheduling fairness is achieved by minimization of the longest working time among all teams. The methodological contribution of Essay 1 is to provide a sequential solution approach for RSPMST that uses a bi-level decomposition of the overall problem and to compare it with a monolithic optimization model. Furthermore, a sensitivity analysis is conducted to evaluate the model performance under each objective and different skill settings.

Essay 2 approaches research question 2 by investigating how companies can manage demand and capacity in a dynamic environment. In particular, it analyzes how to implement a more employee-oriented strategy by ensuring stability of team compositions in a multi-period planning. The main motivation behind this is to find a compromise between cost reduction and an increase of employee loyalty and satisfaction. Moreover, the research provides a planning framework to adapt the schedule to demand changes when new requests arrive within the planning horizon. As both concepts are interrelated, the methodology includes two linked linear optimization programs developed for a short- and long-term planning. The first program is solved through a fix-and-optimize heuristic which is designed to tackle large-scale problems.

Essay 3 elaborates research question 3 by addressing the issue of data uncertainty that is represented as variations in job qualification requirements. The variation might refer to required skills as well as to required experiences. To capture this type of uncertainty, a robust optimization methodology is employed to anticipate skill deviations rather than to merely react to such changes. Based on the concept of budget uncertainty, skill realizations are defined through interval uncertainty sets. In the context of RSPMST, uncertainty can be specified job-wise or be bounded overally. Therefore, two robust counterparts of the considered problem are introduced correspondingly. To evaluate the models performance under different types of uncertainty, experiments involve extensive simulation studies and compare the quality of robust and deterministic solutions by varying the uncertainty level.

For reason of simplicity, traditional scheduling approaches usually consider processing time of tasks as given and constant such that they do not vary with the number and competences of assigned employees. Essay 4 pursues, however, the idea that processing times are linked to team size and efficiency of each single team member. In this way, Essay 4 provides an answer to research question 4. This research particularly identifies and models the conditions under which job operations are performed simultaneously or sequentially. It begins by establishing the concept of „sequential“ use of skill. A parametrization technique is then applied to embed the sequential skill setting into a linear optimization framework. A comparison of large problem instances is based on an Adaptive Neighborhood Search method that combines the main characteristic of a classical neighborhood framework with a sequential search heuristic.

Table 1.1: Overview of manuscripts

Essay	Title	Authors	Journal	VHB JQ3
1	Technician Teaming and Routing with Service-, Cost- and Fairness-Objectives	Y. Anoshkina and F. Meisel	Computers & Industrial Engineering (2019), 135, 868-880.	B
2	Interday Routing and Scheduling of Multi-Skilled Teams with Consistency Consideration and Intraday Rescheduling	Y. Anoshkina and F. Meisel	EURO Journal on Transportation and Logistics (2020), 9, 1-18.	B
3	Robust Optimization Approaches for Routing and Scheduling of Multi-Skilled Teams under Uncertain Job Skill Requirements	Y. Anoshkina, M. Goerigk, F. Meisel	Transportation Science (2021), submitted.	(A)
4	Routing and Scheduling of Multi-Skilled Teams with Simultaneous and Sequential Use of Skill	Y. Anoshkina	Annals of Operations Research (2021), submitted.	(B)

To give a brief overview of the structure of this work, Table 1.1 outlines the parts of the research project in chronological order by publication or submission date. As can be seen from Table 1.1, the Essays 1 and 2 are already published. The Essays 3 and 4 have been submitted and are currently under review. The research was conducted in collaboration with other researchers. Authors confirm their contribution as follows. As the first author, I substantially contributed to all mentioned essays in terms of conceptualizing the research, developing the optimization models and algorithms, conducting the experiments, acquiring and analyzing results as well as writing the manuscripts. Professor Frank Meisel supervised the project, provided critical feedback, helped on shaping the research and write the manuscripts. In the first three essays, he is listed as second author. Essay 3 was written in collaboration with professor Marc Goerigk from University of Siegen, Germany, who provided the methodological input for the design of the robust optimization concept. The breakdown of each author's contribution is provided at the end of the thesis. The authors declare that there are no conflicts of interest regarding the publications listed above. The research was not funded by any specific project grant. The thesis has been written independently and has not been submitted at any other university for the conferral of a PhD degree, neither has the thesis been previously published in full.

1.4 Extended Abstracts

1.4.1 Essay 1

Technician Teaming and Routing with Service-, Cost- and Fairness-Objectives

Yulia Anoshkina, Frank Meisel

Research gap: A central objective of RSPMST studies is to find a schedule that minimizes routing costs. In this essay, we try to align employees perspective with company goals through a variety of alternative objectives. In order to maintain and to increase the organizational performance, we are striving for improving the service quality by minimizing total job completion times. To involve the employee's representation as an objective, we are looking for fairness of workload distribution. Therefore, we attempt to minimize the longest working time among the created teams. We also combine these objectives with a more classical reduction of labor cost expressed in terms of total team working time. To this end, we analyze the problem under different combinations of these objectives and solution methods. More precisely, we compare a monolithic model formulation with a bi-level decomposition approach.

Solution methods: The problem is first solved by using a standard MIP solver. As such a standard approach encounters great difficulty in finding solutions within reasonable computational times, we propose an alternative formulation that decomposes the problem into two subproblems of a smaller dimension that are then solved sequentially. The two-stage approach involves team building as the first stage decision while job assignment and routing decisions are forwarded to the subordinate stage. We support the original concept by involving alternative objectives within the teaming subproblem in the following different ways. Specifically, we reduce labor cost by minimizing the number of assigned employees and we try to reach flexibility of job distribution by maximizing the number of possible job-team assignments. To further combine the benefits of both approaches, multi-objective optimization is applied by weighting both criteria in the objective function. Eventually, low and high job qualification requirements as well as skill settings with differing experience levels are simulated to estimate the effect on solution quality and computation effort.

Results: The sequential solution approach that involves a decomposition technique for the overall problem represents a powerful heuristic regarding solution quality as well as computational effort. Furthermore, it also offers a useful linearization of the cost objective. However, to achieve reasonable results, an appropriate surrogate objective has to be applied for the involved subproblems. In general, the weighted bi-level model that involves a combination of both surrogate objectives provides the best results if improvement of service level or fairness is prioritized. In contrast, the best results for cost reduction are attained if minimization of the number of assigned employees is a main objective. Sensitivity analyzes demonstrate that all proposed variants of the decomposed model can successfully handle the different skill settings and are superior to the monolithic optimization model.

Conclusions: The proposed optimization framework is an effective method to solve the problem and implement different operations management objectives. By selecting a particular setting, decision makers can implement organizational as well as employee-oriented goals. Furthermore, the qualification of the employees has a strong impact on the quality of the obtained solutions. This emphasizes the importance of considering employee qualifications within RSPMST.

1.4.2 Essay 2

Interday Routing and Scheduling of Multi-Skilled Teams with Consistency Consideration and Intraday Rescheduling

Yulia Anoshkina, Frank Meisel

Research gap: In recent years there has been an increased interest in the development of multi-period optimization models to capture interdependences of consecutive planning periods. Despite the broad practical relevance, it appears that multi-period oriented algorithms are limited in scope as they have been developed primarily for rerouting of single employees or for staff scheduling without routing decisions. Therefore, Essay 2 addresses the RSPMST from a multi-period perspective to close this gap. The major premise behind multi-period scheduling presented here is that stable team compositions are essential for highlevel teamwork whereas flexible regrouping of employees supports efficient service operations. This consideration is guided by empirical studies that suggest that team stability creates cohesion among team members, intensifies mutual understanding, enables an efficient communication and facilitates fast decision-making, see e.g. Kalisch et al. (2008). However, in practice, the team configuration is usually based on an hierarchical competence level system or on employees' availability. To change the focus from a resource-based to a more employee-oriented view, we propose a concept of interday planning where the schedules are created on a daily basis but the team structure of the previous period is presented as far as possible. Another essential condition for a successful practical application is the real-time capability to change plans on-the-fly when new jobs arrive. The personnel planning is usually conducted in a dynamic environment that requires a quick respond to changing conditions. The main issue arising in this context is schedule adaptation to demand changes. Therefore, an intraday rescheduling approach is proposed to update an existing schedule to newly arriving jobs. Both concepts are interrelated as the outcome of the interday planning is forwarded as input for intraday rescheduling.

Solution methods: Two interrelated exact and heuristic solution methods are proposed here. First, a linear optimization model is developed to generate a schedule for the current planning period. Furthermore, two alternative ways to formulate team consistency are included. More precisely, the first approach summarizes the total number of employees that switch their teams. The second approach is based on Hamming distance. Here, we analyze if employees work together at consecutive periods or not. Thereby, it makes no difference whether employees stay in the same team or jointly switch to another one. The model is also considered as a part of a fix-and-optimize heuristic framework that can solve instances of a large size. Starting from an initial solution, the algorithm tries to improve the solution by subsequently splitting and merging of teams, swapping jobs between the teams and altering the team structure. The generated schedule is taken up by the intraday model that dynamically inserts new arriving requests. This model can integrate jobs either simultaneously or it can be applied in an iterative manner where the schedule is updated sequentially, i.e. the requests are inserted one-by-one.

Results: The computational results show that team consistency can be successfully integrated into a multi-period planning but at the cost of a slightly lower service level. The compromise can, however, be controlled by the weighting factors applied in the objective function. Both proposed consistency measures can be applied in a flexible way. The problem complexity increases drastically with an increasing number of jobs and employees considered. The proposed fix-and-optimize heuristic provides a good method to also solve problems of large size. Each heuristic phase contributes to the improvement of solution quality by either a further increase of the number of performed jobs or by a reduction of the total job completion time. Furthermore, the intraday model allows to update schedules within very short computational time.

Conclusions: Interday planning is an efficient method to preserve team structures as required and, thus, has a lot of potential to directly improve employee engagement. Intraday rescheduling allows to react to unexpected daily changes and to keep the service level as high as possible. With its help, decision makers can approve urgent jobs and postpone less important jobs in an online manner and guarantee a quick response to user requests.

1.4.3 Essay 3

Robust Optimization Approaches for Routing and Scheduling of Multi-Skilled Teams under Uncertain Job Skill Requirements

Yulia Anoshkina, Marc Goerigk, Frank Meisel

Research gap: The majority of concepts for short- and medium-term planning adopt a view of planning that is based on pure deterministic methods where all parameters are assumed to be known with certainty. However, these methods may not work in a dynamic environment where information can be incomplete or is subject to considerable fluctuations during the planning horizon. In general, randomness in data and parameters is a common feature of many routing optimization problems where uncertainty is usually associated with traveling times or demand. In the context of RSPMST, we consider the uncertainty in job qualification requirements that can arise due to many sources including working environment or incomplete or incorrectly job data submitted by customers. We consider deviations from two perspectives: in terms of required skills and in terms of required experience. Thereby, the first one is always coupled with a variation in the number of required employees while the latter is not necessarily.

Solution methods: We propose two linear modeling optimization frameworks that generate solutions that are robust to possible data variations. Robustness is achieved if a solution remains feasible for all anticipated variations of skill requirement. The level of robustness is defined as the number of jobs that can be still performed after uncertainty is realized. Both our approaches represent a MILP formulation based on the concept of the so-called uncertainty budget proposed by Bertsimas and Sim (2004) where the uncertainty is described through a set that contains all possible realizations of respective parameters. The first approach defines such sets for an aggregated qualification requirement of each single job while the second approach hedges global uncertainty, i.e. skill deviations are modeled as a network and finding an optimal solution is equivalent to finding a longest path in the created graph. Numerical experiments are performed on instance sets originally generated for the deterministic problem version in Essay 1. Simulation-based sensitivity analyses are conducted to compare the performance of both approaches under different levels of data variations.

Results: The obtained results show that a robust solution is associated with a lower service level which is measured as in the other essays by the number of processed jobs. However, the robust approaches outperform significantly the deterministic model in both, the absolute and relative share of jobs that can still be performed if data variations come into the play. In general, the proposed methods can successfully manage the uncertainty and allow to generate a more stable schedule without the original solution becoming infeasible in the case of some small deviations. Thereby, the aggregation method is slightly faster due to a lower number of decision variables and constraints. The robustness factor allows to control the amount of risk the decision maker is willing to accept.

Conclusions: Data uncertainty can incur higher cost for a risk seeking decision maker. The robust approach allows to find a compromise between the risk aversion and the achieved service level. Moreover, decision makers adopting this approach can determine their operations management strategy in such a way, that unforeseen conditions will be less likely to invalidate the base schedule and that small perturbations cannot lead to far-reaching interruptions in working processes.

1.4.4 Essay 4

Routing and Scheduling of Multi-Skilled Teams with Simultaneous and Sequential Use of Skill

Yulia Anoshkina

Research gap: A critical feature of the traditional scheduling approach is that the job processing time is usually considered as constant value or at least as independent on personal qualifications. All studies addressed RSPMST so far are based on this assumption. This is explained by the fact that models incorporating the evolution of employee efficiency over the planning horizon usually lead to non-linearity in the objective function or in constraints and, thus, are less suitable for mathematical programming formulations. Depending on the team size, job operations can be performed simultaneously or sequentially. From this, the deviation in time that the team needs to execute the work can be significant. This can in turn incur higher cost as some jobs cannot be completed in time or cannot be processed at all. In order to ensure schedule reliability under such conditions, Essay 4 provides a concept of sequential use of skill that describes job processing time as a function of skill contributions of each team member.

Solution methods: The main challenge for modeling sequential use of skill lies in the fact that the relationship between employee efficiency and job processing time is not linear and can be hardly described by means of a linear optimization framework. To overcome non-linearity, we conduct a data preprocessing and introduce a method for constructing parametrized efficiency units of labor for each employee-job pair. Further, we develop a concept of an effective team size that allows to define lower and upper bounds if job operations are performed sequentially vs. simultaneously. On this basis, a linear competence-based performance model is derived. As additional parameters and constraints significantly increase the problem's complexity, a Large Adaptive Neighborhood Search is proposed to solve large-scale RSPMST instances. The algorithm involves minimum, average and maximum team configuration schemes and 15 pairs of destroy and repair operators. In addition, a new restart procedure based on a sequential search heuristic is implemented in order to accelerate convergence.

Results: The main contribution here is the problem formulation that allows to establish a linear dependency between team efficiency and job processing times. In order to estimate the benefits of the proposed concept, we compare schedules obtained under the simultaneous skill setting against those obtained under the sequential skill setting. As the principal criterion, we consider the number of assigned and actually performed jobs. The analysis reveals that an inclusion of sequential use of skill requires a longer processing time. Therefore, not all jobs assigned under the simultaneous setting may be performed. In fact, the decrease in service level is often substantial for medium- and large-sized instances and can even exceed 50% of scheduled jobs. The proposed heuristic generates high quality solutions within short computational time. Thereby, the proposed restart mechanism significantly reduces the number of required iterations.

Conclusions: The sequential skill modeling concept has the potential to be used in practice as it accurately estimates service times and guarantees a higher schedule reliability. In doing so, it helps to avoid mental and physical exhaustion of employees due to an excessive job assignment that could result from inappropriate estimation of job processing times.

1.5 Summary and Future Research

This thesis has taken steps to create an employee-friendly planning environment by incorporating a more employee-oriented perspective into teaming and routing decisions. This was done in Essay 1 by implementing employee relevant aspects such as fairness of workload distribution or working time minimization as optimization goals. Essay 2 suggests that stability of team composition can contribute to an increase of employee's efficiency and satisfaction. The robust planning approach proposed in Essay 3 can not only improve the operational performance in the presence of data uncertainty but also help avoid an excessive effort for employees that have to deal with uncertain job requirements. The same holds for the sequential use of skill concept provided in Essay 4. Adequate estimation of processing times prevents both stress and work overload that can lead to a decrease in employee engagement and productivity. Despite the progress achieved, there are still some promising avenues for future research. Further progress can be made by extending towards an employee empowerment concept that looks for opportunities to involve employees into the planning process. For instance, future scheduling models could reformulate team consistency by considering employee preferences not only for stability of team composition but also for collaboration with particular coworkers. Furthermore, an alternative multi-period planning concept can be developed under the premise that complete information is available regarding the type and the number of incoming requests or that requests are known for at least several days. In this case, the schedule could be created for multiple planning periods ahead instead of an interday day-to-day planning as well as team building decisions can be derived with regard not only to the previous period but also for future demand. In this way, team consistency might be supported over a longer time horizon.

Another opportunity for companies to promote employee's engagement is to account for an individual ability to learn by experience and to learn within the team. In this context, it might be interesting to extend the concept of sequential use of skill by learning effects. Thereby, two phenomena can be considered. In general, it is usually expected that repetitive works contribute to an increase of work experience. This results, in turn, in a decrease in time that employees need to complete their assigned task. In contrast, a forgetting curve can be used to describe a decline in employee's efficiency that happens in the course of time and in absence of ongoing learning processes. In the context of team scheduling, the situation is complicated by the fact that the difficulty level in working on each task and the individual learning rate might also be influenced by interaction between the team members. Depending on the extent of this interaction, progress can be guided in both directions. Furthermore, models available in the literature so far would suffer from incorporating learning curves as these are usually defined as non-linear functions which makes the models much more difficult to solve. Therefore, the analytic challenge that must, thus, be addressed is how learning and forgetting phenomena can be incorporated as a part of RSPMST optimization frameworks. To this end, the robust planning approach can be extended by alternative formulations of uncertainty sets. It would also be worthwhile to introduce more powerful metaheuristic approaches for solving large scale practical applications.

Bibliography

Ahmadpour, S. and Ghezavatil, V. (2019). Modeling and solving multi skilled resource constrained project scheduling problem with calendars in fuzzy condition. *Journal of Industrial Engineering International*, 15:179–197.

Al-Yakoob, S. M. and Sherali, H. D. (2007). Multiple shift scheduling of hierarchical workforce with multiple work centers. *Informatica*, 18:325–342.

Altner, D. S., Mason, E. K., and Servi, L. D. (2019). Two-stage stochastic days-off scheduling of multi-skilled analysts with training options. *Journal of Combinatorial Optimization*, 38:111–129.

Bellenguez-Morineau, O. (2008). Methods to solve multi-skill project scheduling problem. *4OR - A Quarterly Journal of Operations Research*, 6:85–88.

Bergman, R. (1994). The key is cross-training. *Hospitals and Health Networks*, 8.

Bertels, S. and Fahle, T. (2006). A hybrid setup for a hybrid scenario: Combining heuristics for the home health care problem. *Computers & Operations Research*, 33:2866–2890.

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

Campbell, G. M. (1999). Cross-utilization of workers whose capabilities differ. *Management Science*, 45:722–732.

Cassera, M. A., Zheng, B., Martinec, D. V., Dunst, C. M., and Swanström, L. L. (2009). Surgical time independently affected by surgical team size. *The American Journal of Surgery*, 198:216–222.

Castillo-Salazar, J. A., Landa-Silva, D., and Qu, R. (2014). A hybrid setup for a hybrid scenario: Combining heuristics for the home health care problem. *Annals of Operations Research*, 239:1–29.

Cordeau, J.-F., Laporte, G., Pasin, F., and Ropke, S. (2010). Scheduling technicians and tasks in a telecommunication company. *Journal of Scheduling*, 13:393–409.

Cordery, J. L. (1989). Multi-skilling: A discussion of proposed benefits of new approaches to labour flexibility with enterprises. *Personnel Review*, 18:13–22.

Cortés, C. E., Gendreau, M., Rousseau, L. M., Souyris, S., and Weintraub, A. (2014). Branch-and-price and constraint programming for solving a real-life technician dispatching problem. *European Journal of Operational Research*, 238:300–312.

Cuevas, R., Ferrer, J.-C., Klapp, M., and Muñoz, J.-C. (2016). A mixed integer programming approach to multi-skilled workforce scheduling. *Journal of Scheduling*, 19:91–106.

Dohn, A., Kohlind, E., and Clausen, J. (2009). The manpower allocation problem with time window and job-teaming constraints. *Computers & Operations Research*, 36:1145–1157.

Eiselt, H. A. and Marianov, V. (2008). Employee planning and workload allocation. *Computers & Operations Research*, 35:513–524.

Eitzen, G. and Panton, D. (2004). Multi-skilled workforce optimization. *Annals of Operations Research*, 127:359–372.

Ernst, A. T., Jiang, H., Krishnamoorthy, M., Owens, B., and Sier, D. (2004). An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, 127:21–144.

Estellon, B., Gardi, F., and Nouioua, K. (2009). High-performance local search for task scheduling with human resource allocation. *Lecture Notes in Computer Science*, 552:1–15.

Everbon, P., Flisberg, P., and Rönnqvist, M. (2006). Laps care - an operational system for staff planning of home care. *European Journal of Operational Research*, 171:962–976.

Firat, M. and Hurkens, C. A. J. (2012). An improved MIP-based approach for a multi-skill workforce scheduling problem. *Journal of Scheduling*, 15:363–380.

Gérard, M., Clautiaux, F., and Sadykov, R. (2016). Column generation based approaches for a tour scheduling problem with a multi-skilled heterogeneous workforce. *European Journal of Operational Research*, 252:1019–1030.

Hashimoto, H., Boussier, S., Vasquez, M., and Wilbaut, C. (2011). A GRASP-based approach for technicians and interventions scheduling for telecommunications. *Annals of Operations Research*, 183:143–161.

Ho, S. C. and Leung, J. M. Y. (2010). Solving a manpower scheduling problem for airline catering using metaheuristics. *European Journal of Operational Research*, 202:903–921.

Hurkens, C. A. J. (2009). Incorporating the strength of MIP modeling in schedule construction. *RAIRO-Operations Research*, 43:409–420.

Ilgen, D. R., Hollenbeck, J. R., Johnson, M., and Jundt, D. (2005). Teams in organization: From input-process-output models to imoi models. *Annual Review of Psychology*, 56:517–543.

Kalisch, B. J., Begeny, S., and Anderson, C. (2008). The effect of consistent nursing shifts on teamwork and continuity of care. *The Journal of Nursing Administration*, 38:132–137.

Kazemipoor, H., Tavakkoli-Moghaddam, R., Shahnazari-Shahrezaei, P., and Azaron, A. (2013). A differential evolution algorithm to solve multi-skilled project portfolio scheduling problems. *International Journal of Advanced Manufacturing Technology*, 64:1099–1111.

Kergosien, Y., Lente, C., and Billaut, J. C. (2009). Home health care problem. An extended multiple traveling salesman problem. *Multidisciplinary International Conference on Scheduling: Theory and Applications, Dublin 10-12 Aug 2009.*

Kovacs, A. A., Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2012). Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of Scheduling*, 15:579–600.

Leggat, S. G. (2007). Effective healthcare teams require effective team members: Defining teamwork competencies. *BMC Health Services Research*, 7:1–10.

Li, N. and Li, L. X. (2000). Modeling staffing flexibility: A case of china. *European Journal of Operational Research*, 124:255–266.

Lim, A., Rodrigues, B., and Song, L. (2004). Manpower allocation with time windows. *Journal of Operational Research Society*, 55:1178–1186.

Mathieu, J., Maynard, M. T., Rapp, T., and Gilson, L. (2008). Team effectiveness 1997-2007: A review of recent advancements and a glimpse into the future. *Journal of Management*, 34:410–476.

McCune, J. C. (1994). On the train gang. *Management Review*, 83:57–60.

Mello, A. S. and Ruckes, M. E. (2006). Team composition. *The Journal of Business*, 79:1019–1039.

Narashiman, R. (2000). An algorithm for multiple shift scheduling of hierarchical workforce on four-day or three-day workweeks. *INFOR*, 38:14–32.

Ozguven, C. and Sungur, B. (2013). Integer programming models for hierarchical workforce scheduling problems including excess off-days and idle labor times. *Applied Mathematical Modelling*, 37:9117–9131.

Park, P. S. (1991). The examination of worker cross-training in a dual resource constrained job shop. *European Journal of Operational Research*, 51:291–299.

Pastor, R. and Corominas, A. (2010). A bicriteria integer programming model for the hierarchical workforce scheduling problem. *Journal of Modeling in Management*, 5:54–62.

Pillac, V., Guéret, C., and Medaglia, A. L. (2013). A parallel metaheuristic for the technician routing and scheduling problem. *Optimization Letters*, 7:1525–1535.

Pinedo, M., Zacharias, C., and Zhu, N. (2015). Scheduling in the service industries: An overview. *Journal of Systems Science and Systems Engineering*, 24:1–48.

Piya, S. and Al-Hinai, N. (2014). Production planning under hierarchical workforce environment. *Proceedings of the World Congress on Engineering*, 2.

ROADEF (2007). Technicians and interventions scheduling for telecommunications. `http://www.roadef.org/challenge/2007/en/index.php`.

Seckiner, S. U., Gokcen, H., and Kurt, M. (2007). An integer programming model for hierarchical workforce scheduling problem. *European Journal of Operational Research*, 183:694–699.

Shahnazari-Shahrezaei, P., Tavakkoli-Moghaddam, R., and Kazemipoor, H. (2013). Solving a multi-objective multi-skilled manpower scheduling model by a fuzzy goal programming approach. *Applied Mathematical Modelling*, 37:5424–5443.

Shelton, P. M., Waite, A. M., and Makela, C. J. (2010). Highly effective teams: A relational analysis of group potency and perceived organizational support. *Advances in Developing Human Resources*, 12:93–114.

Techawiboonwong, A., Yenradee, P., and Das, S. K. (2006). A master scheduling model with skilled and unskilled temporary workers. *International Journal of Production Economics*, 103:798–809.

Tsang, E. and Voudouris, C. (1997). Fast local search and guided local search and their application to British Telecom's workforce scheduling problem. *Operations Research Letters*, 20:119–127.

Xu, J. and Chiu, S. Y. (2001). Effective heuristic procedures for a field technician scheduling problem. *Journal of Heuristics*, 7:495–509.

2 Essay 1

Technician Teaming and Routing with Service-, Cost- and Fairness-Objectives

Yulia Anoshkina[a], Frank Meisel[a]

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

Abstract: In workforce routing and scheduling there are many applications in which differently skilled workers must perform jobs that occur at different locations, where each job requires a particular combination of skills. In many such applications, a group of workers must be sent out to provide all skills required by a job. Examples are found in maintenance operations, the construction sector, health care operations, or consultancies. In this paper, we analyze the combined problem of composing worker groups (teams) and routing these teams under goals expressing service-, fairness-, and cost-objectives. We develop mathematical optimization models for an integrated solution and a sequential solution of the teaming- and routing-subproblems. Computational experiments are conducted to identify the tradeoff of better solution quality and computational effort that comes along with solving the subproblems within a combined monolithic model and within bi-level decomposition schemes. We further analyze the impact of the qualification of employees on the different objectives.

Keywords: Teaming, Routing, Workforce Scheduling, Integrated Solution, Bi-Level Decomposition

2.1 Introduction

Due to limited personnel resources, a growing service portfolio, and customization requirements demanded by customers, efficient use of the available workforce is one of the major business challenges for companies of the service sector. To support companies in efficiently using their workforce, related routing problems, sheduling problems and rostering problems have been investigated for decades, as is shown by the surveys of Ernst et al. (2013), Van den Bergh et al. (2013) and Castillo-Salazar et al. (2014).

The services offered by companies often consist of complex tasks that require multiple employees trained in different skills. Furthermore, services are often provided at the customers' locations, meaning that employees have to travel there. To cope with these challenges, companies form teams of employees in order to ease the organization of work schedules, the synchronization of workforce, and the transportation of workers to customer locations. Examples can be found in the installation and maintenance sector (Kovacs et al. (2012)), the telecommunication industry (Cordeau et al. (2010), Hashimoto et al. (2011)), construction sector (Firat and Hurkens (2012)), airline catering (Ho and Leung (2010)), or the home health care business (Dohn et al. (2009), Castillo-Salazar et al. (2014)).

In this paper we consider a combined teaming and routing problem (CTRP) for a given workforce and a given set of service jobs. This problem comprises the following decisions:

- grouping a given set of employees with various skills and experience levels into a set of teams,
- assigning given jobs with individual qualification requirements to sufficiently skilled teams and determining the job execution order for each team.

We consider different types of skills and different experience levels possessed by employees and required by jobs. Teams have to be built from the available workforce. The considered jobs are then assigned to teams with respect to the qualification requirements of jobs. This decision determines to a large extent the distribution of the workload among the teams. Eventually, deciding on the order in which each team performs its assigned jobs determines the completion times of jobs and, thus, the service quality perceived by customers as well as the actual working time of each team. Hence, the teaming of employees based on their individual skills has to be combined with the job assignment and routing in order to achieve an efficient use of the workforce, a high service quality, and a fair distribution of the workload.

We propose for this problem a monolithic optimization model that can strive for minimizing total job completion time (a proxy for service quality), minimizing total employee working time (a proxy for variable labor cost), and minimizing the maximum working time among teams (a proxy for the fairness of the workload distribution). We attempt to solve this problem using a standard MIP solver. As an alternative solution scheme we introduce a hierarchical model decomposition that respects the natural order of the involved decisions and resembles the strategies typically applied for teaming and routing problems in practice. More precisely, we consider the team building as first subproblem whose solution generates additional constraints for the subsequent job assignment and routing subproblem. For the teaming subproblem, we propose three alternative surrogate objectives in order to produce teams that have a good potential for delivering high quality solutions in the second subproblem. We evaluate the performance of the integrated monolithic model and the sequential decomposition approach on a large set of test instances. We analyze the computational results to provide insight into the performance of the different solution schemes and their impact on service quality, total employee working time, and a fair distribution of the workload among teams.

The paper is organized as follows. In Section 2.2, we provide a literature review.

We then give a formal description and a mathematical formulation of CTRP together with an example in Section 2.3. In Section 2.4, we explain the bi-level decomposition scheme. Section 2.5 contains the computational study that analyses the potentials and limitations of solving CTRP sequentially and integrally. Finally, concluding remarks are presented in Section 2.6.

2.2 Literature

Various types of workforce routing and scheduling problems have been investigated in the literature. A recent survey is provided by Castillo-Salazar et al. (2014). One stream of research studies problems where a single worker can perform a job. The problems are typically handled as extended multiple traveling salesman problems. For instance, Kergosien et al. (2009), Bertels and Fahle (2006) and Everbon et al. (2006) investigate among others the problem of workforce scheduling in the home health care sector where patients are dispersed geographically and require different types of cares. In order to perform non-clinical service on the patient premises, medical staff has to be sent to each patient within respective time windows in such a way that routing costs, personnel costs or overtime costs are minimized. The problem, however, is not unique to the home health care sector. For instance, Tsang and Voudouris (1997) present a case-oriented study for scheduling British Telecom's engineer workforce. The authors introduce skill factors indicating how much time each engineer needs to process a job. They developed fast local search algorithms for solving this problem. Xu and Chiu (2001) propose a general linear programming formulation of a technician scheduling problem with the comparison of three different solution procedures: greedy heuristic, local search algorithm and randomized adaptive search. Lim et al. (2004) investigate a similar problem in the context of port manpower planning where a service center dispatches engineers to perform various jobs at different locations in a port. The primary objective of this model is to minimize the number of required technicians and the secondary objective is to minimize the total travel distance. It is assumed there that all workers have identical qualification. Pillac et al. (2013) address a technician routing and scheduling problem under limited resources like tools and spare parts, which might be replenished at a central depot. The problem is solved by a parallel Adaptive Large Neighborhood Search (ALNS). Finally, Cortés et al. (2014) consider the assignment of technicians for a company that provides repair services in Santiago de Chile and propose a branch-and-price approach for solving this problem.

Another stream of research investigates problems where a team of workers needs to be assigned to each job rather than a single employee. However, many authors assume that worker teams are given without a need to consider team building decisions. For instance, Chuin Lau and Gunawan (2012) route a given set of homogeneous security teams for patroling a public transportation network. Dohn et al. (2009) consider a given workforce of inhomogeneous teams with different qualifications and introduce a binary parameter, which indicates whether or not a team is sufficiently qualified to execute a job. The problem is solved by column generation in a branch-and-price framework.

A number of studies that combine the assignment of technicians to teams and the scheduling of jobs for the created teams originated from the 2007 ROADEF challenge (Estellon et al. (2009), Hurkens (2009), Cordeau et al. (2010), Hashimoto et al. (2011), Kovacs et al. (2012)). Both, technicians and jobs, are characterized by a number of different skills and qualification levels. The jobs may vary in priority and involve precedence constraints. It is also allowed to outsource jobs at some cost, i.e. to exclude them from the assignment and routing decision making. For the solution of this problem, Estellon et al. (2009) propose a combination of a greedy algorithm and local search methods. As the main goal they consider the minimization of job completion times. Hurkens (2009) develops a two-phase MIP-based approach. In the first phase, a MIP model is applied to determine the outsourced jobs while in the second phase matching models determine the job-technician-assignments. Firat and Hurkens (2012) extend this matching approach by adding new matching mechanisms to the solution algorithm. Cordeau et al. (2010) formulate an optimization model for this problem with the goal to minimize the sum of weighted job completion times. They developed a meta-heuristic using a construction heuristic and an ALNS heuristic to solve this problem. Hashimoto et al. (2011) propose a decomposition approach similar to Hurkens, where optimization models are used to determine the jobs that are outsourced. A Greedy Randomized Adaptive Search Procedure is then applied to find the best schedule for the non-outsourced jobs. However, routing of teams is out of scope of all these studies, meaning that travel times and cost for sending teams from one job location to another is not considered in the planning. Kovacs et al. (2012) integrate routing of technicians into the model of Cordeau et al. (2010) and present a corresponding optimisation model for the minimization of total routing and outsourcing cost. For the solution of the resulting problem they also apply an ALNS heuristic. Further, Zamorano and Stolletz (2017) consider teaming and routing within a multi-period planning with the goal to minimize the sum of travel, waiting and overtime costs. In their problem, the size of each team is fixed. They propose an exact branch-and-price algorithm that can solve instances with up to three teams with two technicians each.

Apart from ROADEF 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. The maximum size of a team is restricted to only two employees in this problem. Finally, Fırat 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 gets assigned one of his/her most preferred jobs. In contrast to other studies, every technician can get assigned at most one job. The problem incorporates additional features like, for example, replacement of technicians. Since only a single job is assigned to each team in this problem, routing and scheduling decisions are out of scope of this study.

In our paper, we investigate the problem of teaming and routing of a set of employees with different qualification and experience levels. In contrast to the above mentioned studies, we concentrate on the comparison of service-, fairness-, and cost-objectives together with an analysis of the benefits and the computational burden of an integrated solution versus a sequential solution of the involved subproblems. Thereby we do not fix team sizes but leave it to the optimization to identify best possible teams. We also analyze the influence of these solution concepts on the efficiency of the obtained routes.

2.3 An Integrated Model for CTRP

2.3.1 Notation and Formal Problem Description

We are given a set of jobs J to process, a set of employees M available for performing the jobs, a set of skill domains K needed for performing the jobs, and a set of experience levels L. Each employee $m \in M$ is proficient in one or more skills $k \in K$ at a level of competence $l \in L$. The levels of competence express different factors like experience or specialization in the corresponding skill domain. Let binary matrix q_{mkl} denote the qualifications of employee m in which value 1 indicates that this employee is qualified in skill k at level l, 0 otherwise. We assume that $q_{mkl'} \leqslant q_{mkl}$ for all $l^{'} > l$, i.e. if employee m is experienced in skill k at level $l^{'}$, he/she can also perform jobs requiring lower experience levels l for this particular skill. For example, the qualification matrix q_{mkl} given below illustrates the qualification of an employee m in $|K| = 3$ skills for $|L| = 2$ experience levels. It shows that this employee is proficient in skill $k = 1$ at both levels, $l = 1$ and $l = 2$, but in skill $k = 2$ only at level $l = 1$ and in skill $k = 3$ not at all.

$$q_{mkl} = \begin{array}{c} \\ \\ \end{array}\begin{array}{ccc|c} k=1 & k=2 & k=3 & \\ 1 & 1 & 0 & l=1 \\ 1 & 0 & 0 & l=2 \end{array}$$

The jobs $j \in J$ differ in the number of required employees, their skills $k \in K$ and experience levels $l \in L$. Let integer matrix r_{jkl} denote for a job j the number of employees with qualification k and experience level l required for performing this job. The subsequent example shows a job j that requires one employee with skill $k = 1$ proficient at level $l = 2$ (which also covers $l = 1$) as well as two employees with skill $k = 3$ proficient at level $l = 1$.

$$r_{jkl} = \begin{array}{ccc|c} k=1 & k=2 & k=3 & \\ 1 & 0 & 2 & l=1 \\ 1 & 0 & 0 & l=2 \end{array}$$

In order to perform a job, a team of sufficiently qualified workers has to be composed. Each employee can be a member of at most one team. Each job must be carried out by exactly one team with appropriate skills, where teams can also be overqualified. Apart from the skill requirements, each job is characterized by a processing time p_j, which states the duration of the job. It is assumed that p_j is constant and independent of the composition of its assigned team. Eventually, the jobs occur at different locations. The corresponding network is modeled as an undirected graph $G = (J_0, E)$, where vertex set $J_0 = \{0\} \cup J = \{0, 1, ...|J|\}$ consists of the locations of jobs $1...|J|$ and a dummy job 0 (with $p_0 = 0$) that represents the depot where all teams start and end their routes. Edge set $E = \{(i, j)|i, j \in J_0\}$ represents travel links between different locations, with corresponding travel times d_{ij}.

The following decisions have to be made:

- Grouping of employees (teaming): determine a set of teams T, with $|T| = T_{max}$, by distributing the available workforce M.

- Job assignment and routing: assignment of jobs J to teams T and determination of an order in which each team performs its assigned jobs.

Note that the number of teams T_{max} might be given as an external parameter (e.g. due to organizational reasons, available number of transport vehicles, or the like) or it is derived from the number of jobs and the number of employees by $T_{max} = min\{|J|, |M|\}$.

To model the corresponding decisions, we introduce the following decision variables:

$$x_{mt} = \begin{cases} 1, \text{if employee } m \text{ is assigned to team } t, \\ 0, \text{otherwise.} \end{cases}$$

$$z_{tij} = \begin{cases} 1, \text{if team } t \text{ performs job } i \text{ directly before job } j, \\ 0, \text{otherwise.} \end{cases}$$

f_j completion time of job j,

WT_t working time of team t,

LWT longest working time among all teams.

2.3.2 Model Formulation

Using the introduced notation the mathematical model of CTRP is formulated as follows.

[CTRP]:

$$\textbf{MinFinish:} \text{ minimize } \sum_{j \in J} f_j \tag{2.1}$$

$$\textbf{MinLWT:} \text{ minimize } LWT \tag{2.2}$$

$$\textbf{MinTEWT:} \text{ minimize } \sum_{t \in T} \left(WT_t \cdot \sum_{m \in M} x_{mt} \right) \tag{2.3}$$

subject to:

$$\sum_{t \in T} x_{mt} \leqslant 1 \qquad \forall\ m \in M \tag{2.4}$$

$$\sum_{m \in M} x_{mt} \cdot q_{mkl} \geqslant r_{jkl} \cdot \sum_{i \in J_0} z_{tij} \qquad \forall\ t \in T, j \in J, k \in K, l \in L \tag{2.5}$$

$$\sum_{j \in J} z_{t0j} \leqslant 1 \qquad \forall\ t \in T \tag{2.6}$$

$$\sum_{t \in T} \sum_{i \in J_0} z_{tij} = 1 \qquad \forall\ j \in J \tag{2.7}$$

$$\sum_{i \in J_0} z_{tij} = \sum_{i \in J_0} z_{tji} \qquad \forall\ t \in T, j \in J_0 \tag{2.8}$$

$$\sum_{i \in J_0} \sum_{j \in J_0} (d_{ij} + p_j) \cdot z_{tij} = WT_t \qquad \forall\ t \in T \tag{2.9}$$

$$WT_t \leqslant LWT \qquad \forall\ t \in T \tag{2.10}$$

$$f_i + d_{ij} + p_j \leqslant f_j + Z \cdot (1 - z_{tij}) \qquad \forall\ t \in T, i \in J_0, j \in J \tag{2.11}$$

$$f_j \geqslant 0 \qquad \forall\ j \in J_0 \tag{2.12}$$

$$x_{mt}, z_{tij} \in \{0, 1\} \qquad \forall\ m \in M, t \in T, (i, j) \in E \tag{2.13}$$

We consider three objectives, one expressing service quality, one striving for a fair distribution of workload among the teams, and one expressing variable labor cost of the obtained solution. More precisely, (2.1) minimizes the sum of job completion times, which is the service oriented objective. In (2.2), we focus on the fairness of the workload distribution by minimizing the longest working time among all teams. Using this min-max-objective as a fairness criterion is well established in the literature on the vehicle routing problems with workload equity consideration, see (Matl et al. (2018)). Objective (2.3) minimizes the total employee working time, by multiplying the working time of each team by the number of workers in that team. This approximates the labor costs in settings where employees are paid time wages. Note that objective function (2.3) is not linear.

Constraint (2.4) stipulates that each employee is assigned to at most one team. Employees can be left unassigned if they are not needed for performing the given jobs in the optimal way. Constraint (2.5) demands that each job is performed by a team with

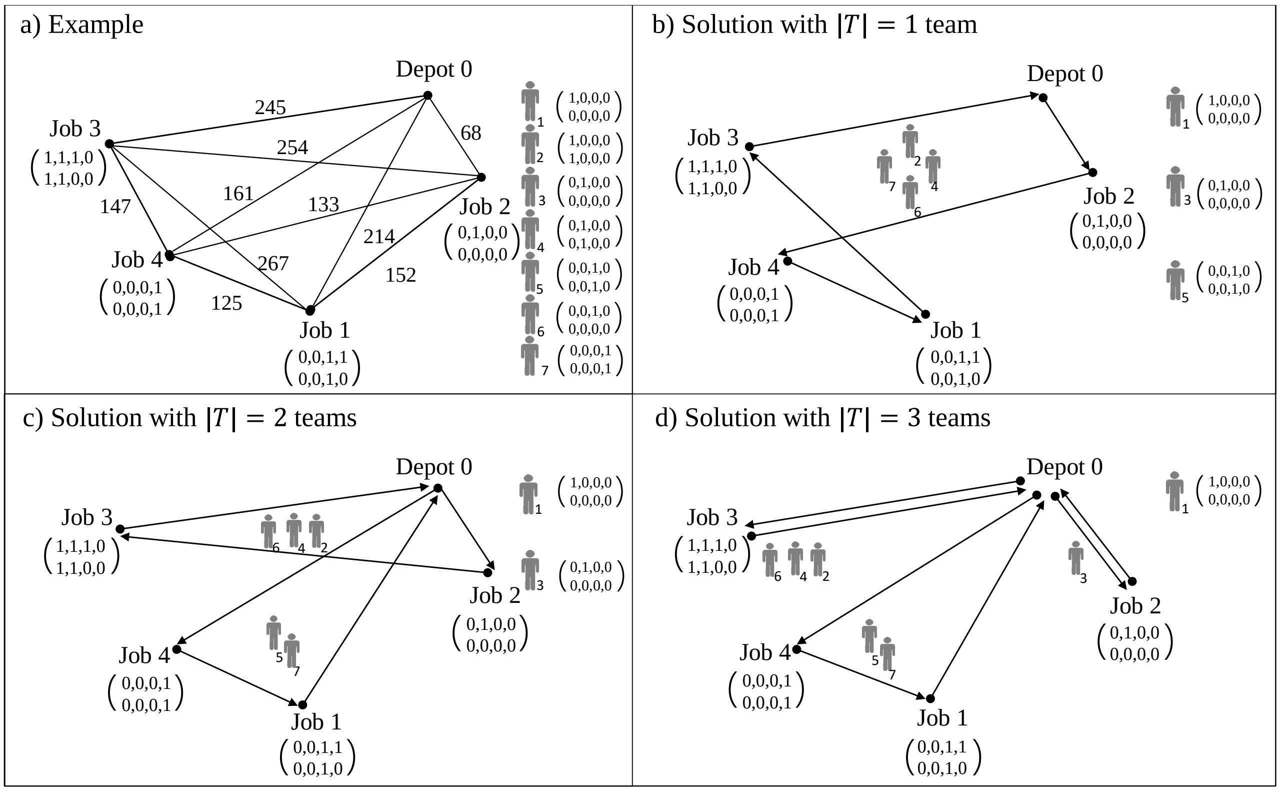

a) Example
b) Solution with |T| = 1 team
c) Solution with |T| = 2 teams
d) Solution with |T| = 3 teams
Depot 0
Job 1
Job 2
Job 3
Job 4
245
254
68
147
161
133
214
267
152
125

Figure 2.1: Solutions for the example instance

appropriate skills. Constraint (2.6) ensures that each team departs from the depot at most once. Constraint (2.7) states that each job j is visited once by one of the teams. Constraint (2.8) guarantees that each team visiting a node j also leaves this node. Constraint (2.9) specifies the value of the working time of each team, which is composed of the traveled times and job processing times. Constraint (2.10) defines the value of the longest working time among all teams. Constraint (2.11) defines the time at which job j is completed by team t. It requires that the completion time of the preceding job i plus the travel time from i to j and the processing time of job j is a lower bound for the completion time of job j. In this constraint Z denotes a sufficiently large positive value. Note that completion times f_j are actually not required if the objective is to minimize the longest working time (2.2) or the total employee working time (2.3). Still, we keep the corresponding constraints (2.11)-(2.12) for the purpose of subtour elimination also under these objectives. Constraints (2.12) and (2.13) specify domains of decision variables.

2.3.3 Examples

We illustrate the described problem by considering the following example instance. We are given $|J| = 4$ jobs and $|M| = 7$ employees with $|K| = 4$ different skills and $|L| = 2$ experience levels. Qualifications of employees, skill requirements of jobs, job locations and travel times are shown in Figure 2.1a. The jobs have processing times $p_1 = 250$, $p_2 = 320$, $p_3 = 350$, $p_4 = 180$. For reasons of simplicity, we assume that each employee is proficient in only one skill, and each job requires at most one employee per skill level. For example, employee 1 is qualified in skill 1 at level 1 only and employee 4 is qualified in skill 2 at both levels. Job 1 requires one employee with skill 3 at level 2 and one employee with skill 4 at level 1. The maximal number of teams that can be created here is $T_{max} = min\{|J|, |M|\} = min\{4, 7\} = 4$. However, there is no guarantee that the available workforce of an instance allows constructing up to T_{max} teams that can all be used for serving jobs. Actually, Figures 2.1b-2.1d illustrate solutions for $|T| = 1, 2, 3$ teams respectively, whereas a solution with $|T| = 4$ active teams does not exist for this particular problem. These three solutions are optimal with regard to objective function (2.1) under the preset number of teams. The solution in Figure 2.1b comprises a single team that consists of employees 2, 4, 6, 7. This team covers the skill requirements of all jobs. The total job completion time is 3858 time units. The solution in Figure 2.1c uses two teams with a total of five employees. The teams are composed such that the four jobs can be distributed among them, reducing the total finishing time to 2437. Eventually, in Figure 2.1d, a third team is built consisting of employee 3 only. The total finishing time is 2040. This is the least total finishing time achievable for this instance. The example illustrates that the formation of teams is crucial for obtaining high quality solutions and it is strongly interdependent with the job assignment and routing decisions.

We provide two further small example instances that demonstrate the conflicting

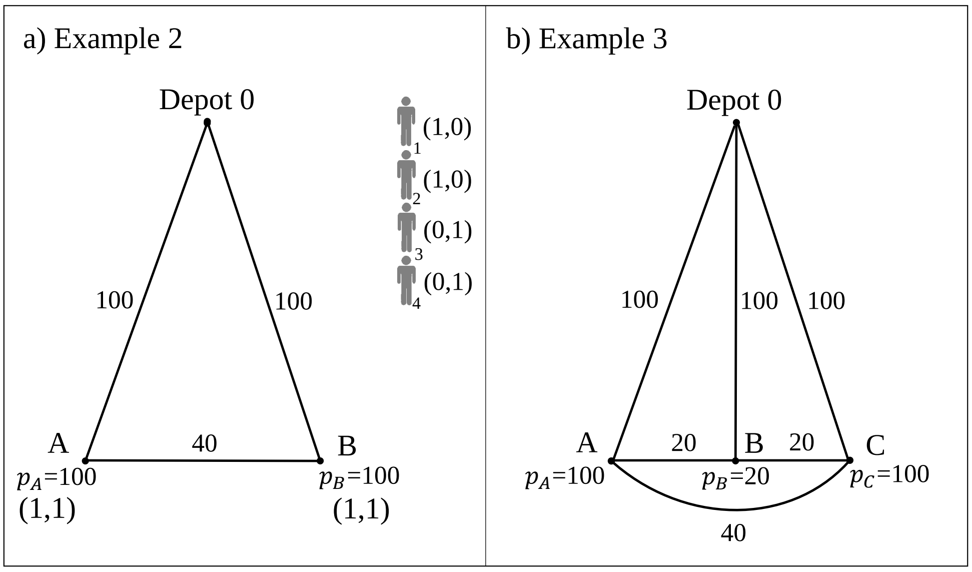


Figure 2.2: Example instances for illustrating conflicting objectives

character of the three objectives MinFinish, MinLWT and MinTEWT. The instance of Figure 2.2a consists of two jobs (A and B), four employees, two skills, and one experience level only. Each job requires two employees for being processed. It can been seen easily that a single team (for example $x_{1,1} = x_{3,1} = 1$) can perform the two jobs in any order (route (0,A,B,0) or route (0,B,A,0)) where both solutions show the same objective function values MinFinish=200+340=540, MinLWT=440 and MinTEWT=440·2=880. If two teams are built (for example $x_{1,1} = x_{3,1} = 1$ and $x_{2,2} = x_{4,2} = 1$), the two jobs can be distributed among them with routes (0,A,0) and (0,B,0). Objective function values are then MinFinish=200+200=400, MinLWT=max$\{300, 300\}$=300 and MinTEWT=300·2 +300·2=1200. Obviously, the second solution delivers minimum total finishing times and also minimum longest working time. However, the first solution has smaller total employee working time, which confirms the conflicting character of objective MinTEWT against the objectives MinFinish and MinLWT. Furthermore, the conflicting character of MinFinish and MinLWT is confirmed by the instance shown in Figure 2.2b, where three jobs (A, B, C) have to be processed. For reason of simplicity, we ignore here the team building decisions and assume that a single team can perform all three jobs. The optimal routing under objective MinLWT is (0,A,B,C,0) which affects MinLWT=460 and MinFinish=200+240+360=800. In contrast, the routing (0,B,A,C,0) achieves the minimum total finishing time of MinFinish=740 but a higher longest working time of MinLWT=480.

2.4 Bi-Level Decomposition

The presented CTRP model is nonlinear under objective function (2.3). Further, the problem contains the uncapacitated vehicle routing problem (VRP), which is known to be NP-hard (Lenstra and Kan (1981)). In order to reduce the complexity, to eliminate the nonlinearity, and to gain more insight into the solvability and the structure of high quality solutions, we suggest a bi-level decomposition framework. We decompose the problem into two subproblems: 1. teaming problem (TP) and 2. job assignment and routing problem (JARP) which are both described in subsections 2.4.1 and 2.4.2. In subsection 2.4.3 we then present the solution scheme that uses the two subproblems for generating solutions to the overall problem CTRP.

2.4.1 Stage 1: Teaming Problem (TP)

At this stage decisions are made for constructing teams. The goal is to form teams that are capable of performing all jobs. For this purpose we introduce a new binary decision variable y_{tj} which indicates whether the composed team t is qualified for performing job j:

$$y_{tj} = \begin{cases} 1, \text{if team } t \text{ is qualified for performing job } j, \\ 0, \text{otherwise.} \end{cases}$$

For the teaming we consider three surrogate objectives that strive for supporting the original service-, cost-, and fairness-objectives in different ways within the TP subproblem:

[TP]:

$$\textbf{MinEmp} \text{ minimize: } \sum_{m \in M} \sum_{t \in T} x_{mt} \tag{2.14}$$

$$\textbf{MaxFlex} \text{ maximize: } \sum_{t \in T} \sum_{j \in J} y_{tj} \tag{2.15}$$

$$\textbf{MinEmpFlex} \text{ minimize: } (1-\alpha) \sum_{m \in M} \sum_{t \in T} x_{mt} - \alpha \cdot \sum_{t \in T} \sum_{j \in J} y_{tj} \tag{2.16}$$

subject to:

$$\sum_{t \in T} x_{mt} \leqslant 1 \qquad \forall\ m \in M \tag{2.17}$$

$$\sum_{m \in M} x_{mt} \cdot q_{mkl} \geqslant r_{jkl} \cdot y_{tj} \qquad \forall\ t \in T, j \in J, k \in K, l \in L \tag{2.18}$$

$$y_{t0} = 1 \qquad \forall\ t \in T \tag{2.19}$$

$$\sum_{t \in T} y_{tj} \geqslant 1 \qquad \forall\ j \in J \tag{2.20}$$

$$x_{mt}, y_{tj} \in \{0,1\} \qquad \forall\ m \in M, t \in T, j \in J_0 \tag{2.21}$$

In (2.14) we minimize the number of employees that are used in the designed teams as a surrogate objective to the minimization of cost in (2.3). In (2.15) we maximize the number of job-team-assignment opportunities for the later routing model. This objective yields flexibility regarding the distribution of jobs among teams and, thus, a fair workload distribution in the subsequent subproblem. Eventually, we combine (2.14) and (2.15) within objective function (2.16) for trading off the number of used employees and the flexibility of job-team-assignments. In this function α denotes a value in the interval $[0,1]$. The larger α, the more we strive for qualified teams that can flexibly get assigned jobs. In contrary, α-values close to 0 indicate that we are more concerned with minimizing the number of employees. Constraint (2.17) ensures that each employee is assigned to at most one team. Constraint (2.18) detects which teams are qualified for which jobs. Constraint (2.19) expresses that the depot is accessible for all teams. Constraint (2.20) ensures that at least one qualified team is built for each job. Note that multiple teams might qualify for a same job. For example, decision variables $y_{13} = 1$ and $y_{23} = 1$ would indicate that both teams 1 and 2 are qualified for performing job 3. This opens up optimization potential for the subsequent job assignment and routing subproblem.

Note that minimizing the number of employees by (2.14) or (2.16) usually effects the construction of a single omnipotent team. In order to generate solutions with more than one team, one might preset a desired number of teams $|T|$ and add the following constraints:

$$\sum_{m \in M} x_{mt} \geqslant 1 \qquad \forall\ t \in T \qquad (2.22)$$

$$\sum_{j \in J} y_{tj} \geqslant 1 \qquad \forall\ t \in T \qquad (2.23)$$

Constraint (2.22) demands that each team contains at least one employee. Constraint (2.23) guarantees that each team is qualified for at least one job. We will use this extension later to identify the optimal number of teams under certain problem settings.

2.4.2 Stage 2: Job Assignment and Routing Problem (JARP)

Having solved TP at stage 1, the second-stage subproblem JARP is to assign jobs to teams and to determine a route for each team using the already introduced routing variables z_{tij}. The values of the stage 1 decision variables x_{mt} and y_{tj} are now fixed and serve as parameters at stage 2. In other words, the composition of teams is now fixed as well as the job-team-qualifications y_{tj}. For example, the x_{mt}-values shown below describe the teams composed for the example solution from Figure 2.1d.

$$x_{mt} = \begin{array}{c|ccc|l} & t=1 & t=2 & t=3 & \\ \hline & 0 & 0 & 0 & m=1 \\ & 1 & 0 & 0 & m=2 \\ & 0 & 0 & 1 & m=3 \\ & 1 & 0 & 0 & m=4 \\ & 0 & 1 & 0 & m=5 \\ & 1 & 0 & 0 & m=6 \\ & 0 & 1 & 0 & m=7 \end{array}$$

The corresponding y_{tj}-values for these teams are given below. They indicate that team 1 (see row 1) can perform job 3, while team 2 is qualified to execute jobs 1 and 4 and team 3 can execute job 2. Note that the first column (index $j = 0$) represents the depot which is accessible by all teams. For this small example, the y_{tj}-variables of stage 1 already fix the assignment of jobs to teams as there is only one team qualified for each job. For larger instances this is usually not the case.

$$y_{tj} = \begin{array}{c|ccccc|l} & j=0 & j=1 & j=2 & j=3 & j=4 & \\ \hline & 1 & 0 & 0 & 1 & 0 & t=1 \\ & 1 & 1 & 0 & 0 & 1 & t=2 \\ & 1 & 0 & 1 & 0 & 0 & t=3 \end{array}$$

The model for the job assignment and routing problem (JARP) is formulated as follows:
[JARP]:

$$\textbf{MinFinish} \text{ minimize: } \sum_{j \in J} f_j \tag{2.24}$$

$$\textbf{MinLWT} \text{ minimize: } LWT \tag{2.25}$$

$$\textbf{MinTEWT} \text{ mimimize: } \sum_{t \in T} \left(WT_t \cdot \sum_{m \in M} x_{mt} \right) \tag{2.26}$$

subject to (2.6) – (2.12) and

$$\sum_{i \in J_0} z_{tij} \leqslant y_{tj} \qquad \forall\ t \in T, j \in J_0 \tag{2.27}$$

$$z_{tij} \in \{0, 1\} \qquad \forall\ t \in T, (i, j) \in E \tag{2.28}$$

The three objectives (2.24) - (2.26) are identical to the original CTRP-objectives (2.1), (2.2) and (2.3). Note that the size of each team follows from $\sum_{m \in M} x_{mt}$ which is also a fixed parameter value now. Hence, in the comparison to model CTRP, objective (2.26) «minimize the total employee working time», is now a linear expression in the second stage subproblem JARP. Constraints (2.6) - (2.12) are taken from model CTRP. As the decision made here have to respect the decisions made at the previous stage 1, (2.27) guarantees that a team t can visit node j only if it is qualified for this job.

2.4.3 Solution Scheme

This section describes the solution scheme for the bi-level decomposition of CTRP. The basic process is illustrated in Figure 2.3. In the first step, we solve subproblem TP in order to construct teams and determine for each job the set of qualified teams. In the next step, we fix the corresponding decision variables. Afterwards, we solve subproblem JARP which delivers the best routing plan for the created teams.

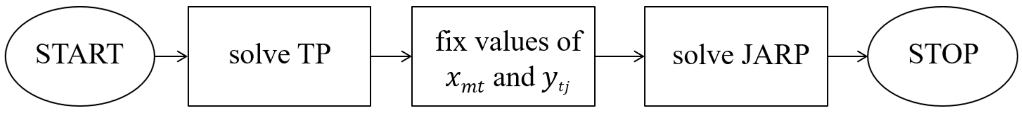


Figure 2.3: Basic solution scheme for bi-level decomposition

If MaxFlex (objective function (2.15)), is used within subproblem TP as the surrogate objective, the process illustrated in Figure 2.3 is executed exactly once. It returns a single solution to the overall problem CTRP. Clearly, there might be several optimal solutions to this first stage subproblem. In this situation, we simply use whatever solution is first generated by the solver that is applied to this subproblem. Further discussion of this issue is provided in Appendix A. If the objectives MinEmp (objective function (2.14)) or MinEmpFlex (objective function (2.16)) are considered within subproblem TP, the first step of the solution scheme may create just one single omnipotent team. After the third step, we then obtain a single solution to CTRP that involves this single team. Such a solution can be of high quality but we might miss even better solutions that use more than just one team. Therefore, we propose an extended solution scheme (Figure 2.4) that iteratively produces solutions for $|T| = 1, 2, \ldots T_{max}$ teams, where $T_{max} = min\{|J|, |M|\}$.

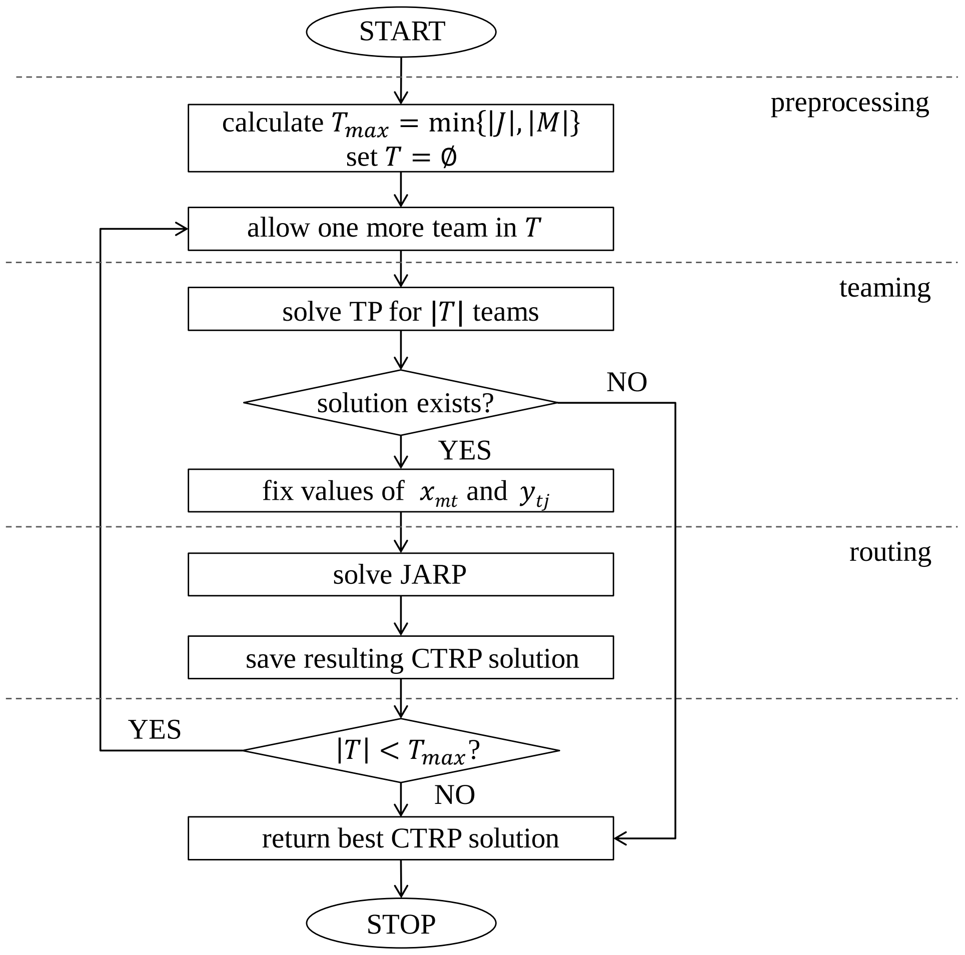

START
preprocessing
calculate $T_{max} = \min\{|J|, |M|\}$
set $T = \emptyset$
allow one more team in T
teaming
solve TP for $|T|$ teams
solution exists?
NO
YES
fix values of x_{mt} and y_{tj}
routing
solve JARP
save resulting CTRP solution
YES
$|T| < T_{max}$?
NO
return best CTRP solution
STOP

Figure 2.4: Extended solution scheme for iterating the number of teams

In this extended solution scheme, model TP is solved once for each number of teams with the additional constraints (2.22) and (2.23) to ensure that the prescribed number of useful teams is constructed. If a feasible solution is found for TP, we fix variables x_{mt} and y_{tj} and continue by solving JARP. As long as $|T| < T_{max}$, one more team is allowed and the procedure is repeated for this size of set T. Recall, that there is no guarantee that a prescribed number of useful teams can be generated for given sets M and J. Therefore, the overall process depicted in Figure 2.4 might terminate prematurely right after failing to solve TP for the current number of teams. Anyhow, the final step before terminating the procedure is to select the best CTRP solution among all solutions created in the conducted iterations. The algorithm returns this solution and stops.

2.5 Computational Study

2.5.1 Generation of Test Instances

For the experiments, we have generated 12 instance sets containing 10 instances each. The instance sets differ in the number of jobs and the number of technicians per instance. The smallest instances (set 1) contain $|J| = 4$ jobs and $|M| = 4$ employees whereas the largest instances (set 12) contain $|J| = 20$ jobs and $|M| = 20$ employees. For all instances we consider $|K| = 3$ skills and one competence level $|L| = 1$. We randomly draw the job processing times p_j from the uniform distribution $U[50, 250]$. The jobs and the depot are randomly located within an area of size 500 $\times$ 500 from which travel times d_{ij} are computed by the corresponding euclidean distances. The employee qualification matrix q_{mkl} of an employee m is generated such that each skill $k = 1, 2, 3$ appears at level $l = 1$ with independent probability of 0.5. From this, it is possible that an employee can be proficient in more than one skill. Furthermore, the generation process guarantees that each employee owes at least one skill (i.e. $\forall m \in M : \exists k \in K \Rightarrow q_{mk1} = 1$). The job requirement matrix r_{jkl} of a job j is generated such that each cell takes value 0, 1 or 2 with equal probability. In other words, concerning a particular skill k, job j either requires no employee with that skill ($r_{jkl} = 0$) or one employee ($r_{jkl} = 1$) or two employees ($r_{jkl} = 2$). The instances are provided at [url hidden for double-blind peer review process].

For the experiments, the MIP models were solved using CPLEX 12.7 on an Intel(R) Core (TM) i7-2600 3.40 GHz with 16 GB of RAM. We set a runtime limit of 3600 seconds per instance for the integrated model CTRP. For the bi-level model we apply this runtime limit just for the second stage because the first stage model is solved within at most a few seconds in all cases. If the second stage problem JARP strives for objective MaxFlex, this model is solved only once per instance and receives the full runtime limit. Under the objectives MinEmp and MinEmpFlex, JARP is solved up to T_{max} times per instance, see discussion in Section 2.4.3. The imposed time limit is therefore divided among the subproblems with each subproblem adhering to a runtime limit of $3600/T_{max}$ seconds.

2.5.2 Comparison of the Models for Different Objectives and Problem Sizes

The first experiment is conducted to test the performance of the integrated model and the bi-level model regarding the required solution times and the achieved solution quality. For this purpose, the whole instance set has been solved by different combinations of models and objectives. Table 2.1 reports aggregated computational results for each model and each instance set when striving for the minimization of total job completion times MinFinish (see Eq. (2.1)). The values reported in a row of this table are averages for the solutions to the 10 instances in the corresponding instance set. The first column of this table shows the instance size. The next five columns show results of model CTRP. They report the number of teams created in the solution (column T), the number of employees assigned to teams (column m), the objective function value (column Obj.), the used computation time (column CPU) and the optimality gap reported by CPLEX (column GAP). The further columns in this table report the results of the bi-level model. To analyze the impact of the submodel variants of the bi-level model, we solved the TP subproblem once under the surrogate objective MaxFlex and once under the surrogate objective MinEmp. In both cases, the objective for the second stage is MinFinish. The results reported in Table 2.1 for these two bi-level formulations are the number of teams in the solution (column T), the number of assigned employees (column m), the relative deviation (column RD) and the computation time (column CPU). Here, the relative deviation RD expresses the deviation of the objective function value achieved by the bi-level model to the objective function value achieved by CTRP. It is calculated as follows:

$$\text{RD} = \frac{\text{Obj(bi-level model) - Obj(CTRP)}}{\text{Obj(CTRP)}} \tag{2.29}$$

The results of Table 2.1 show that the integrated problem CTRP can be solved to optimality for instances of small size only. For medium sizes, a few instances can be solved to optimality whereas instances of size $|J| \times |M| = 10 \times 13$ or larger cannot be solved to optimality within the limited runtime. This is explained by the strong growth of the optimization model with increasing instance size. For example, the smallest instances comprise of 126 decision variables and 168 constraints on average. Medium instances ($|J| \times |M| = 10 \times 13$) have 1362 variables and 1563 constraints. The largest instances lead to models with 9262 variables and 10120 constraints. We also tested an extended runtime limit of 2 hours but did not observe significant improvements of solution quality from this. Nevertheless, integer feasible solutions are found for all instances within one hour of runtime. The bi-level decomposition using MaxFlex can be solved within shorter time especially for the medium size instances. The bi-level decomposition using MinEmp solves even the large instances within less than an hour. Although being faster to solve than the integrated CTRP, the decomposed models represent a heuristic to the overall problem. This results in substantial relative deviations RD ranging from 0.07 to 1.03 for bi-level decomposition with MinEmp. MinEmp is therefore not competitive to CTRP when striving for the overall objective MinFinish. For the bi-level decomposition with MaxFlex, RD is much lower with values ranging from -0.01 to 0.07. The negative RD value indicates that this approach has a potential for delivering solutions that are even better than those obtained by CTRP within the limited runtime. Together with the shorter runtime, the bi-level decomposition with surrogate objective MaxFlex is a useful alternative to solving the integrated model CTRP directly. This holds especially, if the planning has to be done on short notice (for example because jobs are incoming

Table 2.1: Minimization of total job completion time (MinFinish)

Instance	Integrated					Bi-level Decomposition							
	CTRP					MaxFlex				MinEmp			
$/J/ \times /M/$	T	m	Obj.	CPU	GAP	T	m	RD	CPU	T	m	RD	CPU
4 × 4	1.8	3.9	2359	0.09	0	1.7	3.8	0.01	0.02	1.8	3.3	0.07	0.03
4 × 8	3.7	7.9	1642	0.10	0	3.5	7.8	0.03	0.03	2.9	4.4	0.18	0.05
6 × 6	3.2	6.0	3069	8.72	0	3.1	5.9	0.02	0.94	3.0	4.6	0.23	0.35
6 × 12	5.6	11.9	2503	13.15	0	5.5	11.8	0.02	1.55	4.2	5.9	0.20	0.42
8 × 6	3.3	6.0	4862	507.38	0	2.8	5.9	0.06	21.58	2.9	4.9	0.22	20.69
8 × 12	6.4	11.8	3420	1955.21	8	5.6	11.7	0.04	49.30	4.9	7.3	0.35	21.81
10 × 7	3.3	6.9	6630	3315.40	77	3.2	6.7	0.03	860.59	3.6	5.9	0.29	1161.53
10 × 13	6.6	12.7	4433	3600.00	87	5.6	12.8	0.07	683.40	5.2	7.6	0.47	1020.72
15 × 8	3.7	7.9	12162	3600.00	95	3.1	7.8	0.04	3600.00	4.2	6.5	0.36	2177.80
15 × 15	7.6	15.0	7654	3600.00	98	6.1	15.0	0.07	3600.00	6.1	8.6	0.71	2204.78
20 × 10	4.5	10.0	16587	3600.00	99	4.1	10.0	-0.01	3600.00	5.0	7.0	0.65	2628.51
20 × 20	9.6	19.8	10628	3600.00	99	7.8	19.7	0.04	3600.00	7.0	8.9	1.03	2840.44
∅	5.1	10.1	6329	1983.33	47	4.5	10.1	0.04	1334.78	4.3	6.3	0.39	1006.50

closely before the service process starts) which requires lower response times. In such situation, the shorter runtimes of the bi-level decomposition might be considered more relevant than the slightly worse solution quality. Considering the number of teams and employees used in the solutions (columns T and m), CTRP and MaxFlex use almost all available employees and create the larger number of teams while MinEmp - as expected - assigns much fewer employees to a slightly lower number of teams. This might give an advantage to MinEmp if it comes to the minimization of labor cost related objectives, but it represents a systematic disadvantage when striving for service objectives like MinFinish.

The results in Table 2.2 belong to the fairness objective MinLWT. The general performance of the three solution approaches is similar to the previous experiment. Yet, CTRP can now solve almost all medium sized instances to optimality and both bi-level decompositions solve problems much faster under this objective. The CPU time of the bi-level models lies considerably below the time limit even for the largest instances. While MinEmp is extremely fast, its RD values are again very high, making it a less useful approach also under the overall objective MinLWT. In contrast, the average RD of the MaxFlex approach is comparably low (5%) making it a useful heuristic approach, especially for medium and large instances.

Table 2.3 summarizes the results of the minimization of the total employee working

Table 2.2: Minimization of the longest working time (MinLWT)

Instance	Integrated					Bi-level Decomposition							
	CTRP					MaxFlex				MinEmp			
|J| x |M|	T	m	Obj.	CPU	GAP	T	m	RD	CPU	T	m	RD	CPU
4 x 4	1.8	3.9	1259	0.08	0	1.7	3.8	0.03	0.02	1.8	3.3	0.12	0.02
4 x 8	3.2	7.3	922	0.06	0	3.4	7.7	0.02	0.04	2.8	4.3	0.20	0.07
6 x 6	3.0	5.8	1134	0.61	0	3.0	5.8	0.03	0.10	2.8	4.4	0.34	0.14
6 x 12	4.3	10.0	965	0.74	0	4.2	8.9	0.05	0.12	4.0	5.7	0.28	0.21
8 x 6	2.9	5.9	1391	10.55	0	2.6	5.7	0.03	0.63	2.9	4.9	0.26	0.23
8 x 12	5.3	11.3	1001	15.68	0	5.1	10.9	0.02	0.72	4.7	6.9	0.44	0.51
10 x 7	3.1	6.8	1511	160.12	0	3.0	6.5	0.03	1.31	3.4	5.7	0.39	0.52
10 x 13	5.7	12.6	1033	1840.38	3	5.0	11.8	0.12	2.75	4.8	7.1	0.69	1.07
15 x 8	3.8	7.8	1831	3240.22	43	3.1	7.8	0.04	953.28	3.5	5.8	0.55	4.15
15 x 15	6.9	14.6	1132	3600.00	63	5.9	14.8	0.13	1198.15	5.6	8.0	1.12	8.71
20 x 10	4.4	10.0	1834	3600.00	73	4.1	10.0	0.03	2423.23	4.1	6.0	0.93	127.29
20 x 20	8.3	19.5	1312	3600.00	77	6.8	18.2	0.11	1807.16	4.9	6.7	1.29	147.27
∅	4.3	9.8	1277	1339.04	22	4.0	9.5	0.05	532.25	3.9	5.8	0.55	24.17

time MinTEWT. This objective is nonlinear which is why it cannot be applied when solving model CTRP by a standard MIP solver like CPLEX. For providing a basis of comparison, we compute a lower bound for TEWT as described in Appendix B. Since computing the lower bound involves solving a variant of CTRP it can, unfortunately, not be computed for very large instances within a runtime of one hour. However, we obtain lower bounds at least for the first 8 instance sets. The average lower bound is reported for these instance sets in column LB. We furthermore report the GAP values (column GAP) expressing the deviation of the objective function achieved by the bi-level model to the reported LB value:

$$\text{GAP} = \frac{\text{Obj.(bi-level model) - LB}}{\text{LB}} \tag{2.30}$$

To compare the two decomposition approaches, we report in column RD the relative deviation of the objective function value achieved by this bi-level model to the objective function value achieved by MaxFlex. Furthermore, in column B/E/W we report for each instance set the number of instances where MinEmp performs better (B), no different (E=equal) or worse (W) than MaxFlex. For example, in row 4x4 column B/E/W shows values 2/8/0, meaning that MinEmp delivers better solutions for 2 instances, equal solutions for 8 instances and is worse in no case. The obtained results show that MinEmp achieves substantially lower total employee working times. This is confirmed by the (average) negative RD values for almost all instance sets. Furthermore, for the first 8

Table 2.3: Minimization of the total employee working time (MinTEWT)

Instance		Bi-level Decomposition											
		MaxFlex					MinEmp						
$\|J\| \times \|M\|$	LB	T	m	Obj.	GAP	CPU	T	m	Obj.	GAP	RD	B/E/W	CPU
4 × 4	3515	1.1	2.7	4044	15%	0.02	1.1	2.6	3835	9%	-0.05	2/8/0	0.03
4 × 8	3013	1.3	2.7	3352	11%	0.03	1.7	3.0	3211	7%	-0.04	4/5/1	0.07
6 × 6	3819	1.6	3.3	4677	22%	0.12	1.5	2.7	4219	10%	-0.10	5/3/2	0.18
6 × 12	3695	1.7	3.0	4028	9%	0.20	1.8	2.9	4103	11%	0.02	3/4/3	0.26
8 × 6	5073	1.7	4.1	6545	29%	0.41	1.8	3.4	6022	19%	-0.08	8/2/0	0.39
8 × 12	4828	2.2	4.4	5826	20%	1.39	1.5	2.7	5210	8%	-0.11	7/0/3	0.80
10 × 7	6078	1.8	4.2	7787	28%	3.17	2.6	4.7	7688	26%	-0.01	5/2/3	0.81
10 × 13	5763	2.3	5.2	7401	28%	5.91	2.3	3.8	6357	10%	-0.14	7/1/2	1.57
15 × 8	-	1.8	4.7	11171	-	41.04	2.2	4.1	10465	-	-0.06	5/1/4	10.04
15 × 15	-	2.1	4.9	10641	-	404.33	2.3	3.8	8926	-	-0.16	7/0/3	20.18
20 × 10	-	1.7	4.4	13872	-	531.14	2.9	4.6	12377	-	-0.11	9/1/0	188.79
20 × 20	-	2.2	5.0	13123	-	1466.43	2.3	3.7	10570	-	-0.19	9/1/0	280.93
∅	-	1.8	4.0	7705	20%	204.51	2.1	3.7	6915	12%	-0.09	6/2/2	42.00

instance sets, the average gap to the lower bound is 12%. Interestingly, the GAP values are relative stable over these sets, which indicates that the solution quality does hardly deteriorate with increasing problem size. Eventually, MinEmp delivers better solutions than MaxFlex for the majority of instances. A paired sample t-test based on the computational results of both methods yields a significant mean difference (with $p < 0.01$) confirming that method MinEmp outperforms method MaxFlex. Furthermore, MinEmp solves most instance sets much faster than MaxFlex. Therefore, MinEmp is the strategy to apply if a company strives for minimizing the total working time of employees. This can be explained as follows. As MaxFlex strives for highly qualified teams, it creates on average a small number of 1.8 teams per instance using on average 4.0 employees compared to MinEmp which creates more teams (2.1) using less employees (3.7). This effects longer working time for fewer but larger teams under MaxFlex explaining its inferior performance compared to MinEmp.

To summarize this experiment, the obtained results show that the decomposition approach yields an effective heuristic for all three objectives. However, for this approach, the appropriate surrogate objective needs to be selected for subproblem TP. Eventually, good solutions for CTRP-objectives MinFinish and MinLWT are achieved by the MaxFlex-decomposition approach whereas CTRP-objective MinTEWT is best solved by using the MinEmp-decomposition approach.

2.5.3 Weighted Bi-Level Model

The previous experiment showed that bi-level models MaxFlex and MinEmp perform quite differently under the CTRP-objectives MinFinish, MinLWT, MinTEWT. There is no model that outperforms the other one for all three objectives. For this reason, we investigate here whether a combination of the two bi-level models is advantageous. For this purpose, we apply to the first subproblem TP the weighted objective function (2.16) that combines MinEmp and MaxFlex using a weight α (with $0 \leqslant \alpha \leqslant 1$). Figure 2.5 shows for different values of α and selected instance sets, the minimum total employee working time MinTEWT of the final CTRP solutions obtained by the weighted bi-level model. We observe that the extreme $\alpha = 1$ (corresponding to MaxFlex) performs much weaker than $\alpha = 0$ (corresponding to MinEmp) as was revealed already by Table 2.3. However, for values $0 < \alpha < 1$ there are only minor improvements over $\alpha = 0$ that can be seen hardly in the figure.

To better reveal this improvement, we show in Table 2.4 the numerical results of the weighted bi-level model with $\alpha = 0.5$ for all three CTRP-objectives. For the objectives MinFinish and MinLWT, column RD expresses here the relative deviation of the objective function value obtained from the weighted bi-level model against the solutions of the best performing pure bi-level model MaxFlex that were reported in Tables 2.1 and 2.2. The RD-values reported for MinTEWT correspond to the relative deviation of

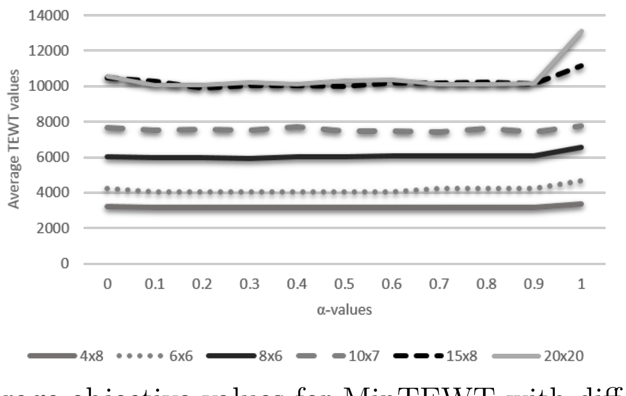


Figure 2.5: Average objective values for MinTEWT with different α-values

Table 2.4: Performance of the weighted bi-level model MinEmpFlex with $\alpha = 0.5$

Instance	MinFinish				MinLWT				MinTEWT			
|J| x |M|	T	m	RD	CPU	T	m	RD	CPU	T	m	RD	CPU
4 x 4	1.8	3.7	-0.01	0.03	1.8	3.7	-0.02	0.03	1.0	2.5	0.02	0.03
4 x 8	3.6	6.7	-0.01	0.12	3.2	6.2	-0.02	0.10	1.4	2.7	-0.01	0.10
6 x 6	3.2	5.6	-0.01	1.81	3.0	5.5	-0.02	0.28	1.5	2.8	-0.04	0.29
6 x 12	5.4	10.7	-0.01	7.83	4.0	8.4	-0.04	0.62	1.6	2.7	-0.06	0.78
8 x 6	3.3	6.0	-0.05	59.16	2.8	5.7	0.00	0.99	1.8	3.9	0.00	1.13
8 x 12	5.9	11.4	-0.01	333.63	4.7	11.0	0.00	4.78	1.7	3.1	0.00	4.07
10 x 7	3.4	6.7	-0.01	1690.61	3.0	6.6	0.00	8.06	2.1	4.5	-0.03	5.87
10 x 13	6.1	12.1	-0.03	2529.27	4.8	11.8	-0.07	44.21	2.0	3.8	-0.03	14.99
15 x 8	3.8	7.9	-0.05	2295.31	3.4	7.7	-0.02	659.39	2.0	4.4	-0.04	158.42
15 x 15	6.9	14.6	-0.02	2597.97	6.0	13.8	-0.06	1258.09	2.0	4.0	0.00	465.30
20 x 10	5.2	10.0	0.02	2629.32	4.3	9.9	-0.03	1299.76	2.2	4.2	-0.05	815.52
20 x 20	8.6	18.9	0.05	2939.99	7.1	17.9	-0.09	1524.17	2.1	3.6	-0.02	1538.49
∅	4.8	9.5	-0.01	1257.09	4.0	9.0	-0.03	400.04	1.8	3.5	-0.02	250.42

the weighted bi-level model against the solutions of the pure MinEmp model that were reported in Table 2.3. The numerous negative RD values observed for all three objectives in Table 2.4 confirm that the weighted bi-level approach clearly outperforms the best-performing pure bi-level models in almost all settings. The average improvement for the three objectives ranges from 1% to 3%, where the highest improvement of 9% is achieved for the largest instances under objective MinLWT.

2.5.4 Influence of Employee Qualifications

This experiment tests the impact of employee qualifications on the quality of the obtained solutions. For this purpose, we generated an additional employee qualification matrix where all employees are fully qualified in all three skill domains. We refer to this setting as high-skilled (HS). In this setting, teams are no longer built to cover the skills required by jobs but to meet requirements of jobs that need more than one worker for a particular skill-level ($r_{jkl} > 1$). As a benchmark for this setting, we use the results obtained from the original employee qualification matrices, which we refer to as low-skilled (LS). Furthermore, we solved all 120 instances with the HS qualifications under each modeling approach presented in this paper. Thereby, we use the same runtime limits as before, meaning that solutions of model CTRP are not necessarily optimal for larger instances.

Figures 2.6 to 2.8 show for LS and HS, as well as for each (bi-level) model, the average

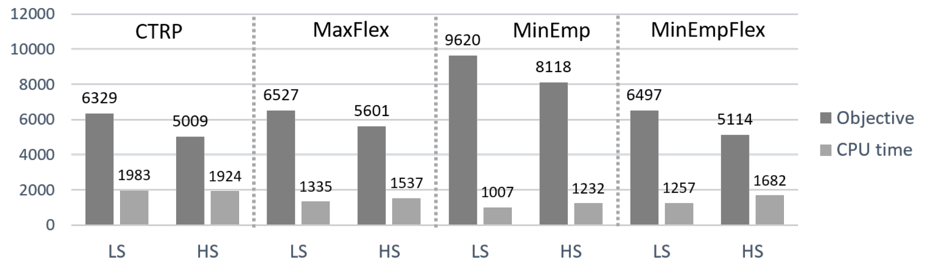


Figure 2.6: Low skilled (LS) and high skilled (HS) workforce performance under MinFinish

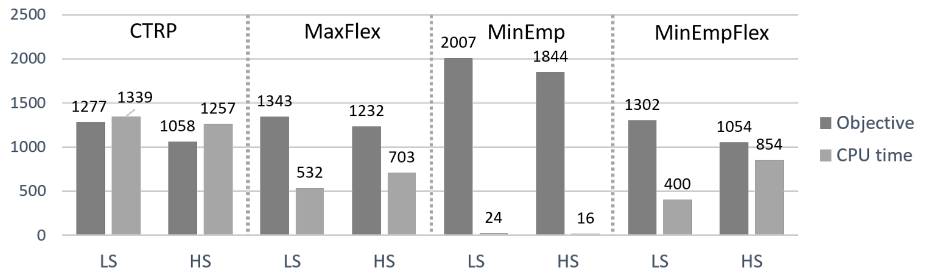


Figure 2.7: Low skilled (LS) and high skilled (HS) workforce performance under MinLWT

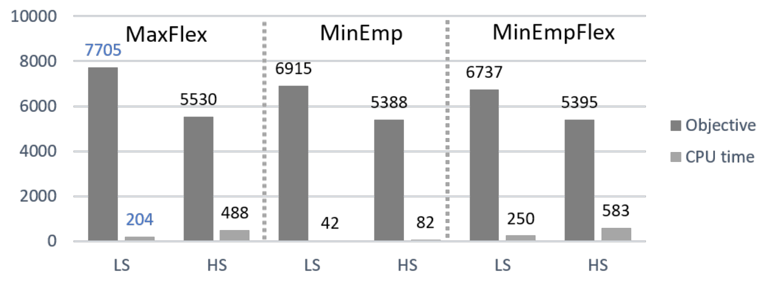


Figure 2.8: Low skilled (LS) and high skilled (HS) workforce performance under MinTEWT

objective function value and the average CPU time over all 120 instances when striving for MinFinish, MinLWT, and MinTEWT. The results for setting LS are those reported in Tables 2.1-2.4. As expected, we observe that higher qualifications can be used for finishing jobs faster, reducing longest working times of teams and diminishing total employee working time. These benefits can be achieved by all models proposed in this paper. Like before, the weighted bi-level model MinEmpFlex can be considered as a good compromise for quickly achieving solutions of very good quality in all considered settings. In some cases MinEmpFlex even produces better solutions than CTRP within the limited runtime, which explains why the average objective value of HS is smaller under MinEmpFlex than under CTRP in Figure 2.7.

Eventually, we analyze the impact of the number of qualification levels. Therefore,

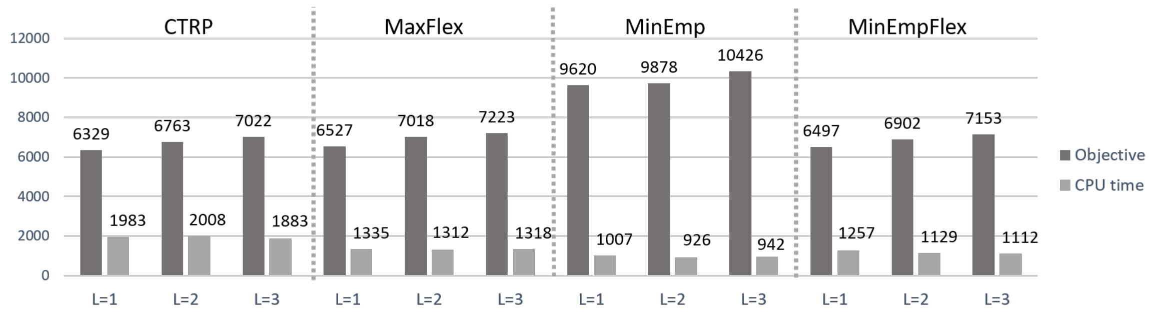


Figure 2.9: Varied number of skill levels under MinFinish

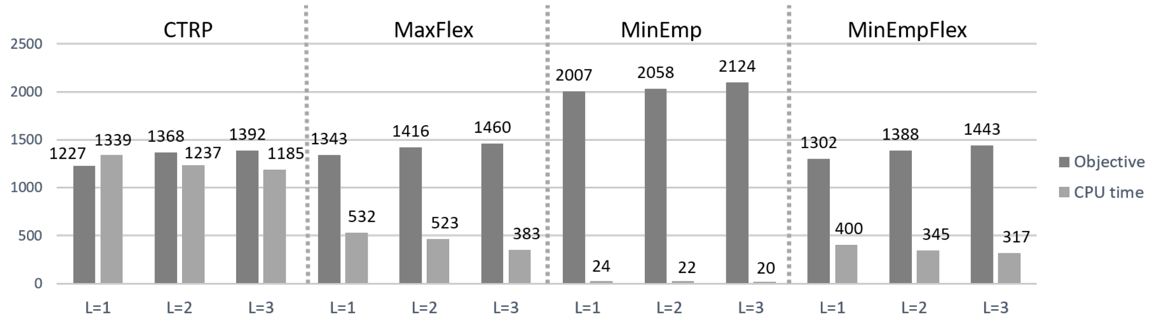


Figure 2.10: Varied number of skill levels under MinLWT

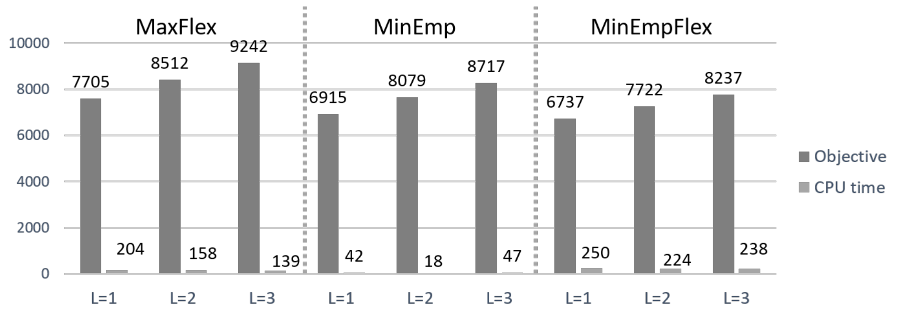


Figure 2.11: Varied number of skill levels under MinTEWT

based on the original qualification matrices, we generated additional skill levels $l = 2$ and $l = 3$ as follows. For employees that are proficient in a skill k at level $l = 1$ ($q_{mk1} = 1$), we assign the second skill level ($q_{mk2} = 1$) with probability 0.8. Furthermore, for employees that are proficient at level $l = 2$ ($q_{mk2} = 1$), we assign the third skill level ($q_{mk3} = 1$) again with probability 0.8. The elements in the job requirement matrix r_{jkl} at higher levels ($l = 2$, $l = 3$) take value $r_{j,k,l-1}$ or $max(r_{j,k,l-1} - 1, 0)$ with a probability of 0.5 each.

Figures 2.9-2.11 show the results obtained under the original low skill setting with $L = 1$ skill level in comparison with $L = 2$ and $L = 3$ skill levels for objectives MinFinish, MinLWT and MinTEWT and all applicable methods. The high skill setting has not been applied to $L = 2$ and $L = 3$ as results will not be affected for a perfectly skilled workfore where each worker is qualified for all skills at the highest level. The results in Figures 2.9-2.11 confirm the findings of the previous tests. MinEmpFlex provides very good solution quality at relatively low computation times in all situations. We observe slight increases in the objective function values when turning from $L = 1$ to $L = 2$ and $L = 3$ under all objectives and for all methods. This is an expected outcome, because the more diverse skill requirements reduce the flexibility of forming teams that are qualified for a particular job. This negatively effects the performance of the teams even if the problem can be solved to optimality. Interestingly, the solution times either stay the same or they even decrease when we solve problems with a higher number of skill levels. This shows that the methods can well cope with different numbers of skill levels.

2.6 Conclusions

In this paper we presented optimization models for the integrated and sequential solution of the combined manpower teaming and routing problem. We considered objectives representing the minimization of the total job finishing time, of the longest working time among teams and of the total employee working time. These objectives represent service, fairness, and cost aspects of the considered problem. The sequential solution approach uses a bi-level decomposition of the overall problem. It was shown that this decomposition effects a linearization of the total employee working time objective. For this reason, all the bi-level models can be solved using the CPLEX MIP solver. Our experimental results show that the decomposition approach represents a useful heuristic if an appropriate surrogate objective is chosen for the involved subproblems. Eventually, we could show that a weighted bi-level model, that uses a mixture of the surrogate objectives, consistently performed best. This approach provides good heuristic solutions even for large instances for which CPLEX cannot solve the combined model within the considered runtime limit. We also showed by experiment that the qualification of the employees has a strong impact on the quality of the obtained solutions, which confirms the importance of considering employee qualification within the combined manpower teaming and routing problem.

As future research, we will explore the possibilities of splitting and merging teams within the planning horizon to provide more flexibility to the operations planning. Also, we propose to investigate the interdependence of job processing times and the composition and skill levels of the assigned team.

Appendix A. Selection of Candidate TP Solution

Maximizing the job-team-assignment opportunities through MaxFlex-objective (2.15) can lead to several optimal solutions with same objective function value but differing compositions of teams. As an example, consider a problem with three employees, three jobs, three skills, and one skill level. Let the employee skill vectors be $q_1 = (1, 0, 0), q_2 = (0, 1, 0)$, and $q_3 = (0, 0, 1)$. Let the qualification requirement vectors of the three jobs be $r_1 = (1, 0, 0), r_2 = (0, 1, 0)$, and $r_3 = (0, 0, 1)$. Table 2.5 lists the five possible team compositions, where m_1, m_2, and m_3 refer to the three employees.

Table 2.5: Team composition variants

Team	Composition 1	Composition 2	Composition 3	Composition 4	Composition 5
1	m_1	m_1, m_2	m_2, m_3	m_1, m_2	m_3, m_2, m_3
2	m_2	m_3	m_1	m_2	-
3	m_3	-	-	-	-

Each of the five team compositions returns the same objective function value of MaxFlex = 3 and, thus, all these solutions are optimal for subproblem TP under this surrogate objective. A question is then, whether one solution should be prefered over the others when turning to the second stage problem. Here, one could argue that a team composition with a larger number of smaller teams is preferable as these teams can be dispatched flexibly to jobs. The objectives MinFinish and MinLWT would then benefit from a parallelization of jobs and the MinTEWT objective would benefit from not sending oversized teams to jobs. In this sence, team composition 1 from Table 2.5 would be considered the best alternative. In order to determine among all optimal solutions the one with the largest number of qualified teams, one should first solve model TP to obtain the optimal value for MaxFlex. Afterwards, the following model will be solved to obtain a team composition that reaches MaxFlex with a largest possible number of qualified teams (i.e. non-empty teams that are qualified for at least one job):

$$\text{maximize: } \sum_{t \in T} w_t \tag{2.31}$$

subject to:

$$(2.17) - (2.21)$$

$$w_t \leqslant \sum_{j \in J} y_{tj} \qquad \forall t \in T \tag{2.32}$$

$$\sum_{t \in T} \sum_{j \in J} y_{tj} = MaxFlex \tag{2.33}$$

$$w_t \in \{0, 1\} \qquad \forall t \in T \tag{2.34}$$

In this model, binary decision variable $w_t = 1$ indicates that team t is qualified for at least one of the jobs, 0 otherwise. Objective (2.31) maximizes the number of such qualified teams. Constraint (2.32) guarantees that w_t takes value 1 only for qualified teams. Constraint (2.33) ensures that the team composition leads to the same number of team-job-assignments as was generated in the first run of the TP model. Constraint (2.34) specifies the domain of the new decision variables. Note that there might still be multiple optimal solutions for this model, where further mechanism could be applied to select one of these solutions.

Nevertheless, we found from experiments that the strategy described here does eventually not lead to better CTRP solutions. For this reason, we kept the original mechanism of arbitrarily selecting any optimal solution at the TP-stage and describe this alternative mechanism only here in the appendix.

Appendix B. Lower Bound for MinTEWT

Since the CTRP objective MinTEWT is nonlinear, it is not possible to solve this problem by a MIP solver. In order to assess the solution quality obtained by the bi-directional heuristic solution approach under this objective, we describe here how to compute a lower bound for objective MinTEWT.

We first calculate for each job $j \in J$ the minimal number of employees that is needed to perform this job. This number is denoted by A_j. We obtain this value from solving the following reduced version of the TP model under the surrogate objective (2.14) once for each job $j \in J$.

$$\text{minimize: } A_j = \sum_{m \in M} x_m \tag{2.35}$$

subject to:

$$\sum_{m \in M} x_m \cdot q_{mkl} \geqslant r_{jkl} \qquad \forall k \in K, l \in L \tag{2.36}$$

$$x_m \in \{0,1\} \qquad \forall\ m \in M \tag{2.37}$$

As this model addresses only one job j at a time, the selected subset of employees represents the team of minimum size that is required for performing this job. For this reason, the reduced version of model TP can go without variables y_{tj}, a reduced set of constraints, and index t dropped from the original variables x_{mt}.

Having determined A_j for all jobs $j \in J$, we subsequently determine a lower bound on the total employee working time by solving the following variant of model CTRP.

$$\textbf{LB} \text{ minimize: } \sum_{t \in T} LBEWT_t \tag{2.38}$$

subject to:

$$(2.4) – (2.13)$$

$$LBEWT_t \geqslant \sum_{t \in T} A_j \cdot WT_t - Z \cdot (1 - \sum_{i \in J_0} z_{tij}) \qquad \forall j \in J, t \in T \qquad (2.39)$$

$$LBEWT_t \geqslant 0 \qquad \forall t \in T \qquad (2.40)$$

Here, the continous variable $LBEWT_t$ expresses a lower bound on the employee working time of team t. This value is estimated in (2.39) using the parameter A_j, which is a lower bound on the actual size of the team that serves job j, instead of the size of the team that is actually determined in this optimization model. This way, the nonlinearity in the computation of total employee working time is eliminated. Objective function (2.38) then sums up these variables over all teams to obtain a lower bound on the original objective function MinTEWT. Note that this model still includes the determination of teams to get meaningful routes (variables z_{tij}) and, thus, a tight lower bound. Therefore, it might be challenging to solve this lower bound formulation if instances get too large.

Bibliography

Bertels, S. and Fahle, T. (2006). A hybrid setup for a hybrid scenario: Combining heuristics for the home health care problem. *Computers & Operations Research*, 33:2866–2890.

Castillo-Salazar, J. A., Landa-Silva, D., and Qu, R. (2014). Workforce scheduling and routing problem: Literature survey and computational study. *Annals of Operations Research*, 239:1–29.

Chuin Lau, H. and Gunawan, A. (2012). The patrol scheduling problem. *Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012), Son, Norway*, pages 175–192.

Cordeau, J.-F., Laporte, G., Pasin, F., and Ropke, S. (2010). Scheduling technicians and tasks in a telecommunication company. *Journal of Scheduling*, 13:393–409.

Cortés, C. E., Gendreau, M., Rousseau, L. M., Souyris, S., and Weintraub, A. (2014). Branch-and-price and constraint programming for solving a real-life technician dispatching problem. *European Journal of Operational Research*, 238:300–312.

Dohn, A., Kolind, E., and Clausen, J. (2009). The manpower allocation problem with time window and job-teaming constraints. *Computers & Operations Research*, 36:1145–1157.

Ernst, A. T., Jiang, H., Krishnamoorthy, M., and Sier, D. (2013). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153:3–27.

Estellon, B., Gardi, F., and Nouioua, K. (2009). High-performance local search for task scheduling with human resource allocation. *Lecture Notes in Computer Science*, 552:1–15.

Everbon, P., Flisberg, P., and Rönnqvist, M. (2006). Laps care - an operational system for staff planning of home care. *European Journal of Operational Research*, 171:962–976.

Fırat, M., Briskorn, D., and Laugier, A. (2016). A branch-and-price algorithm for stable workforce assignments with hierarchical skills. *European Journal of Operational Research*, 251:676–685.

Firat, M. and Hurkens, C. (2012). An improved MIP-based approach for a multi-skill workforce scheduling problem. *Journal of Scheduling*, 15:363–380.

Hashimoto, H., Boussier, S., Vasquez, M., and Wilbaut, C. (2011). A GRASP-based approach for technicians and interventions scheduling for telecommunications. *Annals of Operations Research*, 183:143–161.

Ho, S. C. and Leung, J. M. Y. (2010). Solving a manpower scheduling problem for airline catering using metaheuristics. *European Journal of Operational Research*, 202:903–921.

Hurkens, C. A. (2009). Incorporating the strength of MIP modeling in schedule construction. *RAIRO-Operations Research*, 43:409–420.

Kergosien, Y., Lente, C., and Billaut, J. C. (2009). Home health care problem. An extended multiple traveling salesman problem. *Multidisciplinary International Conference on Scheduling: Theory and Applications, Dublin 10-12 Aug 2009.*

Kovacs, A. A., Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2012). Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of Scheduling*, 15:579–600.

Lenstra, J. K. and Kan, A. H. G. (1981). Complexity of vehicle routing and scheduling problem. *Networks*, 11:221–227.

Lim, A., Rodrigues, B., and Song, L. (2004). Manpower allocation with time windows. *Journal of Operational Research Society*, 55:1178–1186.

Matl, P., Hartl, R. F., and Vidal, T. (2018). Workload equity in vehicle routing problems: A survey and analysis. *Transportation Science*, 52:239–260.

Pillac, V., Guéret, C., and Medaglia, A. L. (2013). A parallel metaheuristic for the technician routing and scheduling problem. *Optimization Letters*, 7:1525–1535.

Tsang, E. and Voudouris, C. (1997). Fast local search and guided local search and their application to British Telecom's workforce scheduling problem. *Operations Research Letters*, 20:119–127.

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

Xu, J. and Chiu, S. Y. (2001). Effective heuristic procedures for a field technician scheduling problem. *Journal of Heuristics*, 7:495–509.

Zamorano, E. and Stolletz, R. (2017). Branch-and-price approaches for the multiperiod technician routing and scheduling problem. *European Journal of Operational Research*, 257:55–68.

3 Essay 2

Interday Routing and Scheduling of Multi-Skilled Teams with Consistency Consideration and Intraday Rescheduling

Yulia Anoshkina[a], Frank Meisel[a]

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

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.

Keywords: Multi-Skill Workforce Scheduling, Interday Planning, Intraday Rescheduling, Consistency, Synchronization

3.1 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), Kazirzadeh 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 Russel 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 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 3.2, 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 3.3. In Section 3.4, we explain our solution method. Section 3.5 contains the computational study that analyses the potentials and limitations of the proposed models and the heuristic. Concluding remarks are presented in Section 3.6.

3.2 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. Fırat 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 (2017) 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 (2017) 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.

3.3 Mathematical Optimization Models

3.3.1 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 $q_{mkl'} \leqslant q_{mkl}$ 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.

$$Q_m = \begin{array}{ccc|c} k=1 & k=2 & k=3 & \\ \hline 1 & 1 & 0 & l=1 \\ 1 & 0 & 0 & l=2 \\ 1 & 0 & 0 & l=3 \end{array}$$

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 $l = 1$).

$$R_j = \begin{array}{c} \\ \left(\begin{array}{ccc} k=1 & k=2 & k=3 \\ 1+1+\mathbf{1} & \mathbf{1} & 1+1 \\ 1+\mathbf{1}+0 & 0 & \mathbf{1}+\mathbf{1} \\ \mathbf{1}+0+0 & 0 & 0+0 \end{array}\right) \end{array} \begin{array}{l} \\ l=1 \\ l=2 \\ l=3 \end{array} = \left(\begin{array}{ccc} k=1 & k=2 & k=3 \\ 3 & 1 & 2 \\ 2 & 0 & 2 \\ 1 & 0 & 0 \end{array}\right) \begin{array}{l} \\ l=1 \\ l=2 \\ l=3 \end{array}$$

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.

3.3.2 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 $J_{d+1}...J_{|D|}$ 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, Figure 3.1 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

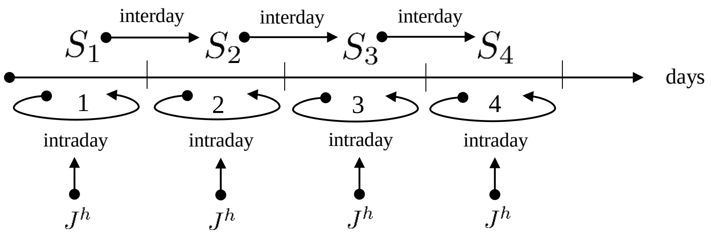


Figure 3.1: Planning procedure

Section 3.3.3 and the intraday model in Section 3.3.4.

3.3.3 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 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. Figure 3.2 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

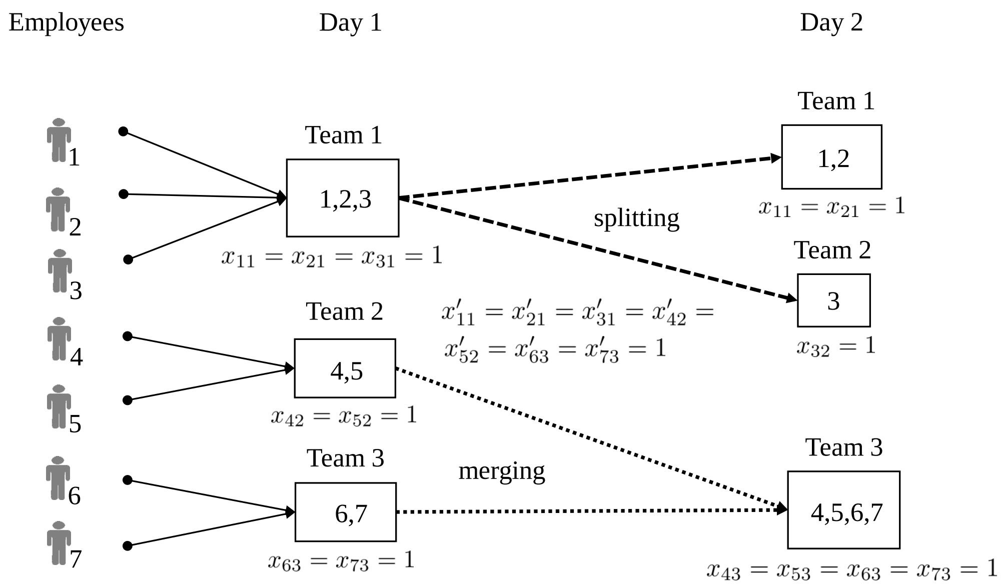


Figure 3.2: Rescheduling of teams in interday planning

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:

$$\text{minimize: } \alpha \cdot \sum_{m \in M} X_m - \beta \cdot \sum_{t \in T} \sum_{i \in J_d^0} \sum_{j \in J_d} z_{tij} + \gamma \cdot \sum_{t \in T} \sum_{j \in J_d} f_{tj} \tag{3.1}$$

subject to:

$$\sum_{t \in T} x_{mt} \leqslant 1 \qquad \forall m \in M \tag{3.2}$$

$$\sum_{m \in M} x_{mt} \cdot q_{mkl} \geqslant r_{jkl} \cdot \sum_{i \in J_d^0} z_{tij} \qquad \forall j \in J_d, k \in K, l \in L, t \in T \tag{3.3}$$

$$\sum_{j \in J_d} z_{t0j} \leqslant 1 \qquad \forall\ t \in T \tag{3.4}$$

$$\sum_{t \in T} \sum_{i \in J_d^0} z_{tij} \leqslant 1 \qquad \forall\ j \in J_d \tag{3.5}$$

$$\sum_{i \in J_d^0} z_{tij} = \sum_{i \in J_d^0} z_{tji} \qquad \forall\ j \in J_d^0, t \in T \tag{3.6}$$

$$f_{ti} + g_{ij} \leqslant s_{tj} + \mathcal{M} \cdot (1 - z_{tij}) \qquad \forall i \in J_d^0, j \in J_d, t \in T \tag{3.7}$$

$$s_{tj} + p_j \leqslant f_{tj} + \mathcal{M} \cdot \left(1 - \sum_{i \in J_d^0} z_{tij}\right) \qquad \forall j \in J_d, t \in T \tag{3.8}$$

$$s_{tj} \geqslant a_j - \mathcal{M} \cdot \left(1 - \sum_{i \in J_d^0} z_{tij}\right) \qquad \forall j \in J_d, t \in T \tag{3.9}$$

$$s_{tj} \leqslant b_j + \mathcal{M} \cdot \left(1 - \sum_{i \in J_d^0} z_{tij}\right) \qquad \forall j \in J_d, t \in T \tag{3.10}$$

$$x_{mt} - x'_{mt} \leqslant X_m \qquad \forall m \in M, t \in T \tag{3.11}$$

$$x'_{mt} - x_{mt} \leqslant X_m \qquad \forall m \in M, t \in T \tag{3.12}$$

$$s_{tj}, f_{tj} \geqslant 0 \qquad \forall j \in J_d^0, t \in T \tag{3.13}$$

$$x_{mt}, X_m, z_{tij} \in \{0,1\} \qquad \forall i,j \in J_d^0, m \in M, t \in T \tag{3.14}$$

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 (3.1) 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 (3.2) guarantee that each employee is assigned to at most one team. Constraints (3.3) ensure that created teams are sufficiently qualified for performing their assigned jobs. Constraints (3.4) demand that each team departs from the depot at most once. Constraints (3.5) state that each job is visited by at most one team. Constraints (3.6) ensure that each team visiting a node j also leaves this node. Constraints (3.7) 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, $\mathcal{M}$ denotes a sufficiently large positive value. Constraints (3.8) define the time at which job j is completed by team t. Together, (3.7) and (3.8) also avoid subtours in the solution. Constraints (3.9)-(3.10) reflect the time windows for the starting time of job j. Constraints (3.11)-(3.12) set the auxiliary variables X_m. Constraints (3.13)-(3.14) specify the domains of decision variables.

3.3.4 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. Figure 3.3 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 Figure 3.3a 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 Figure 3.3a 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 base-

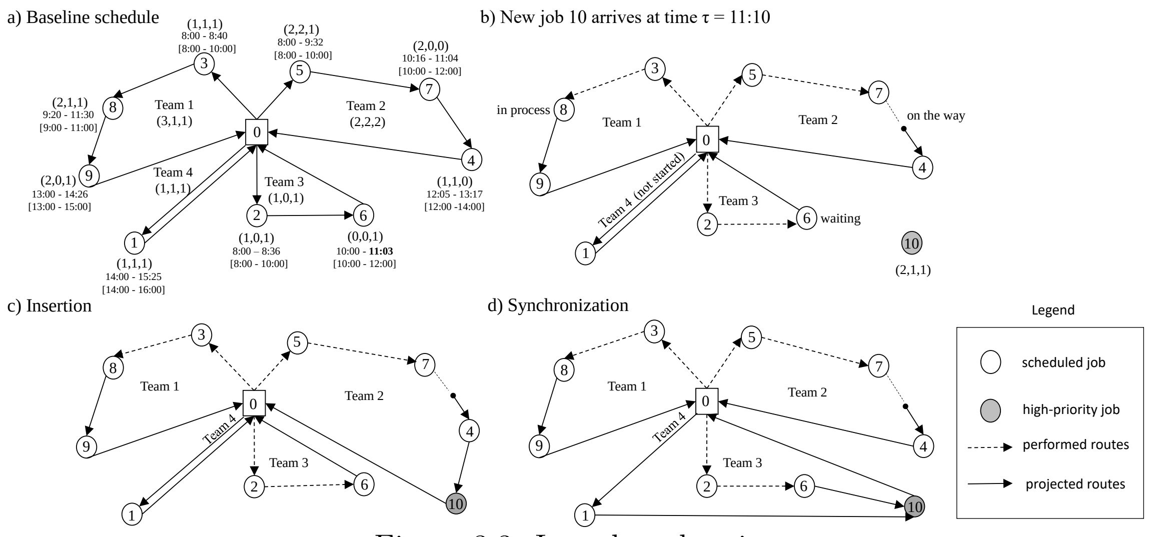


Figure 3.3: Intraday planning

line 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 Figure 3.3b, a single new high-priority job $J^h = \{10\}$ arrives at time $\tau = 11{:}10$. This job's skill requirements are $r_{10,k,1} = (2, 1, 1)$. Based on the above assumptions, we propose the following rescheduling strategies:

1. *Insertion*: Job $j \in J^h$ is inserted into the existing route of a sufficiently qualified team t. In the example in Figure 3.3c, the new job is inserted into the route of team 2 right after job 4.
2. *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 Figure 3.3d, 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 Figure 3.3b),
- the team is on the way to its next job (see team 2 in Figure 3.3b),
- the team has completed its last assigned job and is waiting for a new job (see team 3 in Figure 3.3b),
- the team has not started from the depot yet (see team 4 in Figure 3.3b).

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 Figure 3.3b, 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 3.3.3. 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.

$$\text{maximize: } \alpha \cdot \sum_{j \in J^h} h_j - \beta \cdot \sum_{t \in T} \sum_{i \in J_t} \sum_{j \in J^h} z_{tij} - \theta \cdot \sum_{t \in T} \sum_{j \in J_t \setminus \{J^h\}} S_{tj} - \gamma \cdot \sum_{t \in T} \sum_{j \in J_t} f_{tj} \tag{3.15}$$

subject to:

$$\sum_{t \in T} \left(\sum_{m \in M} x'_{mt} \cdot q_{mkl} \cdot \sum_{i \in J_t} z_{tij} \right) \geqslant r_{jkl} \cdot h_j \qquad \forall j \in J^h, k \in K, l \in L \tag{3.16}$$

$$\sum_{j \in J_t} z_{tD_tj} = 1 \qquad \forall t \in T \tag{3.17}$$

$$\sum_{i \in J_t} z_{tiD_t} = 0 \qquad \forall t \in T, D_t \neq 0 \tag{3.18}$$

$$\sum_{i \in J_t} z_{ti0} = 1 \qquad \forall t \in T \tag{3.19}$$

$$\sum_{i \in J_t} z_{tij} = 1 \qquad \forall t \in T, j \in J_t \setminus \{D_t, 0\} \setminus J^h \tag{3.20}$$

$$\sum_{i \in J_t} z_{tij} \leqslant 1 \qquad \forall j \in J^h, t \in T \tag{3.21}$$

$$\sum_{i \in J_t} z_{tij} = \sum_{i \in J_t} z_{tji} \qquad \forall t \in T, j \in J_t \setminus \{D_t, 0\} \tag{3.22}$$

$$f_{tD_t} = f_t^{\mathcal{T}} \qquad \forall t \in T \tag{3.23}$$

$$f_{ti} + g_{ij} \leqslant s_{tj} + \mathcal{M} \cdot (1 - z_{tij}) \qquad \forall t \in T, i \in J_t, j \in J_t \setminus \{0\} \tag{3.24}$$

$$s_{tj} + p_j \leqslant f_{tj} + \mathcal{M} \cdot \left(1 - \sum_{i \in J_t} z_{tij}\right) \qquad \forall t \in T, j \in J_t \backslash \{0\} \tag{3.25}$$

$$s_{tj} - s_{t'j} \leqslant \delta_{max} + \mathcal{M} \cdot \left(2 - \sum_{i \in J_t} z_{tij} - \sum_{i \in J_{t'}} z_{t'ij}\right) \qquad \forall j \in J^h, t, t' \in T, t \neq t' \tag{3.26}$$

$$s_{t'j} - s_{tj} \leqslant \delta_{max} + \mathcal{M} \cdot \left(2 - \sum_{i \in J_t} z_{tij} - \sum_{i \in J_{t'}} z_{t'ij}\right) \qquad \forall j \in J^h, t, t' \in T, t \neq t' \tag{3.27}$$

$$s_{tj} \geqslant a_j - \mathcal{M} \cdot \left(1 - \sum_{i \in J_t} z_{tij}\right) \qquad \forall t \in T, j \in J_t \backslash \{D_t, 0\} \tag{3.28}$$

$$s_{tj} \leqslant b_j + \mathcal{M} \cdot \left(1 - \sum_{i \in J_t} z_{tij}\right) \qquad \forall t \in T, j \in J_t \backslash \{D_t, 0\} \tag{3.29}$$

$$s_{tj} \geqslant f_t^{\tau} + g_{D_t, j} - \mathcal{M} \cdot \left(1 - \sum_{i \in J_t} z_{tij}\right) \qquad \forall t \in T, j \in J_t \backslash \{D_t, 0\} \tag{3.30}$$

$$s_{tj} - s'_{tj} \leqslant S_{tj} \qquad \forall t \in T, j \in J_t \backslash J^h \tag{3.31}$$

$$s'_{tj} - s_{tj} \leqslant S_{tj} \qquad \forall t \in T, j \in J_t \backslash J^h \tag{3.32}$$

$$s_{tj}, f_{tj}, S_{tj} \geqslant 0 \qquad \forall t \in T, j \in J_t \tag{3.33}$$

$$h_j, z_{tij} \in \{0, 1\} \qquad \forall t \in T, i, j \in J_t \tag{3.34}$$

Objective function (3.15) 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 (3.16) demand that high-priority jobs are performed by teams with appropriate skills where multiple teams might be involved in serving a job. Constraints (3.17) 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 (3.18) forbid to return to the virtual depot D_t and, also, to insert jobs before the virtual depot into the route. Constraints (3.19) stipulate that all teams return to the depot 0 at the end of their tour. Constraints (3.20) ensure that each team performs all its already assigned jobs. Constraints (3.21) state that each high-priority job can be visited by a team at most once. Constraints (3.22) balance the flow ensuring that each team visiting a node j also leaves this node. Constraints (3.23)-(3.25) define the start and completion times of jobs. Note that, similar to the interday model, Constraints (3.24)-(3.25) allow to avoid subtours in the solution. Constraints (3.26)-(3.27) 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 (3.28)-(3.29) state that all jobs, if scheduled, have to be performed within the predefined time windows. Constraints (3.30) 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^τ 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 (3.31)-(3.32) define the values S_{tj}. Constraints (3.33)-(3.34) specify domains of decision variables.

3.4 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 3.4.1), improvement by merging and splitting of teams (Subsection 3.4.2), improvement by swapping of jobs (Subsection 3.4.3) and randomized disturbance for diversifying the search (Subsection 3.4.4).

3.4.1 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 $\sum_{m \in M} q_{mkl} \cdot x_{mt} \geqslant r_{jkl}$. 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:

$$\mu_{tj} = \sum_{k\in K}\sum_{l\in L}\left(\sum_{m\in M} q_{mkl}\cdot x_{mt} - r_{jkl}\right) \qquad \forall t \in T, j \in J_t^{qual} \quad (3.35)$$

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 4-14 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:

$$\text{maximize: } \beta\cdot\sum_{i\in J^{temp}}\sum_{j\in J^{temp}} z_{tij} - \gamma\cdot\sum_{j\in J^{temp}} f_{tj} \quad (3.36)$$

subject to: (3.4), (3.6) – (3.10) and

$$\sum_{i\in J^{temp}} z_{tij} \leqslant 1 \qquad \forall j \in J^{temp}\backslash J_t \quad (3.37)$$

$$\sum_{i\in J^{temp}} z_{tij} = 1 \qquad \forall j \in J_t \quad (3.38)$$

$$s_{tj}, f_{tj} \geqslant 0 \qquad \forall j \in J^{temp} \tag{3.39}$$

$$z_{tij} \in \{0,1\} \qquad \forall i,j \in J^{temp} \tag{3.40}$$

Objective (3.36) maximizes the number of performed jobs as the main goal and the job completion times as a subordinate goal ($\beta > \gamma$). Constraints (3.37) demand that each newly considered job can be visited by the considered team at most once. Constraints (3.38) guarantee that all jobs J_t that were already assigned to team t in previous iterations are still contained in the route. Constraints (3.4) and (3.6)-(3.10), 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.

3.4.2 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:

$$\vartheta_j = \sum_{k \in K} \sum_{l \in L} r_{jkl} \qquad \forall j \in J^{un} \tag{3.41}$$

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:

$$\mathcal{E}_{jt} = \sum_{k \in K} \sum_{l \in L} \max \left\{ r_{jkl} - \sum_{m \in M} q_{mkl} \cdot x_{mt^M} - \sum_{m \in M} q_{mkl} \cdot x_{mt}, 0 \right\} \qquad \forall j \in J^{un}, t \in T \tag{3.42}$$

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.

3.4.3 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 (3.36)-(3.40), 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).

3.4.4 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 (3.1) 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.

3.5 Computational Study

The computation experiments aim at testing the performance of the models and methods described in Sections 3.3 and 3.4 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 seconds for the interday planning and of 200 seconds 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.

3.5.1 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 least 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-hour workday with a total of 480 minutes. 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 $\delta_{max} = 30$.

3.5.2 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 3.1 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 to 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

Table 3.1: Interday model CPLEX

Instance	Problem size		Consistency setting										Service setting									
$\|J_d\| \times \|M\|$	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%

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 seconds (column GAP_{3600}) and after an extended runtime limit of 7200 seconds (column GAP_{7200}). The optimality gap expresses the deviation of the objective function achieved by the interday model to the lower bound value LB 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 3.1, 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 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 3.4. 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

Table 3.2: 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

in Appendix C. Results obtained by the heuristic are reported in Tables 3.2 and 3.3. 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 3.2 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 3.3.

Table 3.2 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.

Table 3.3: Interday fix-and-optimize heuristic, service setting

Instance	Initial solution				Splitting and Merging					Swapping					Randomized disturbance							
$\lvert J_d\rvert \times \lvert M\rvert$	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

Regarding the service setting, Table 3.3 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 to 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 four minutes 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.

3.5.3 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 3.4 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 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 seconds, which is considerably below the runtime limit of 200 seconds. This indicates

Table 3.4: Intraday model CPLEX

Instance	$\lvert J^h\rvert = 1$						$\lvert J^h\rvert = 3$						$\lvert J^h\rvert = 5$					
$\lvert J\rvert \times \lvert M\rvert$	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%	-	-	-	-	-	-

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 to 5 of them for most of the instances. Here, not all instances are solved to optimality ($GAP > 0$) within the runtime limit of 200 seconds. For the largest instances (400×80, 500×100) not even a feasible solution is obtained within the runtime limit of 200 seconds 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 3.4.

We now briefly analyze how the solution quality responds to differing values of job

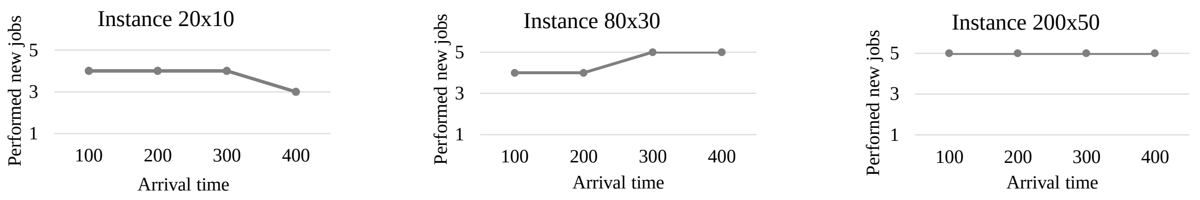


Figure 3.4: Variation of arrival time

arrival time τ. For this purpose, we changed τ from its original value 100 to values 200, 300 and 400. Figure 3.4 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 3.4) and even worse than the optimal solution achieved under the more restrictive $\tau \geqslant 300$. 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 3.5. Comparing Tables 3.4 and 3.5 for $|J^h| = 1$,

Table 3.5: 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

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 performed jobs for most instances but requires much lower computational times of at most 9 seconds. For $|J^h| = 5$, all instances are solved feasibly now and within just a few seconds. The computational time does not exceed one minute 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.

3.6 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 two hours, 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 Figure 3.2. 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:

$$\text{minimize:} \sum_{m \in M} \sum_{m' \in M} Y_{mm'} \tag{3.43}$$

$$x_{mt} + x_{m't} - 1 \leqslant y_{mm'} \qquad \forall m, m' \in M, t \in T \tag{3.44}$$

$$2 - x_{mt} - \sum_{t' \in T \setminus \{t\}} x_{m't'} \geqslant y_{mm'} \qquad \forall m, m' \in M, t \in T \tag{3.45}$$

$$\sum_{t \in T} x_{mt} \geqslant y_{mm'} \qquad \forall m, m' \in M \qquad (3.46)$$

$$y_{mm'} - y'_{mm'} \leqslant Y_{mm'} \qquad \forall m, m' \in M \qquad (3.47)$$

$$y'_{mm'} - y_{mm'} \leqslant Y_{mm'} \qquad \forall m, m' \in M \qquad (3.48)$$

$$x_{mt}, y_{mm'}, Y_{mm'} \in \{0, 1\} \qquad \forall m, m' \in M, t \in T \qquad (3.49)$$

Objective (3.43) maximizes team consistency by minimizing changes in employee pairing. This objective can be combined with the other objectives as done in (3.1). Constraints (3.44) enforce $y_{mm'} = 1$ if employees m and m' are in the same team. Constraints (3.45) set $y_{mm'} = 0$ if employees m and m' are assigned to different teams. If employee m is not assigned to any team, Constraints (3.46) set $y_{mm'} = 0$ for all m'. Constraints (3.47)-(3.48) 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 3.6. Compared with the results in Table 3.1, 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 Tables 3.1 and 3.6, 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

Table 3.6: Interday model CPLEX (alternative consistency formulation)

Instance	Problem size		Consistency setting						Service setting					
$\lvert J_d \rvert \times \lvert M \rvert$	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 3.7: Comparison of different α and β weights in the objective

	Instance	Service setting (original measure)						Service setting (alternative measure)					
α, β	$\lvert J_d \rvert \times \lvert M \rvert$	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%

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.

Furthermore, we analyze how the solutions react to variations of weights α and β in the objective function. Table 3.7 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 Tables 3.1 and 3.6. 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 3.3.3, 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 3.1. 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 3.1. 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 $J_d$
1:  $J_t \leftarrow \{0\} \; \forall t \in T$
2:  generate sets $J_t^{qual} \; \forall t \in T$
3:  sort jobs $j \in J_t^{qual}$ in ascending order of $\mu_{tj} \; \forall t \in T$
4:  while $T \neq \emptyset$ do
5:      $t \leftarrow arg\ min_{t' \in T}\{|J_{t'}^{qual}|\}$    ▷ select team with fewest jobs in set $J_t^{qual}$
6:      while $|J_t| < \lambda^{max}$ and $J_t^{qual} \neq \emptyset$ do
7:          $J^{temp} \leftarrow J_t \cup (\lambda^{max} - |J_t|)$ first jobs from set $J_t^{qual}$    ▷ build set of temporary jobs
8:          $J_t^{qual} \leftarrow J_t^{qual} \backslash J^{temp}$    ▷ remove new considered jobs from $J_t^{qual}$
9:          solve routing problem (3.36)-(3.40) for team $t$
10:         update set $J_t$ according to jobs processed in obtained route
11:     end while
12:     $J_{t'}^{qual} \leftarrow J_{t'}^{qual} \backslash J_t \; \forall t' \in T, t' \neq t$    ▷ remove assigned jobs from $J_t^{qual}$ of all not yet examined teams
13:     $T \leftarrow T \backslash \{t\}$    ▷ remove considered team $t$ from the list $T$
14: end while
```

Algorithm 2: Interday Planning: Splitting and Merging of Teams

```
    Input: solution of Algorithm 1, set of still unprocessed jobs $J^{un}$
 1: create free team list $T^F$
 2: sort jobs $j \in J^{un}$ in descending order of difficulty factor $\vartheta_j$
 3: for $j \in J^{un}$ do
 4:     if aggregated teams in $T^F$ are qualified for job $j$ then
 5:         $t^M \leftarrow \emptyset$
 6:         while $t^M$ is not qualified for job $j$ do                          ▻ find teams to merge
 7:             $t \leftarrow arg\ min_{t' \in T^F}\{\mathcal{E}_{jt'}\}$        ▻ choose team $t$ with minimum skill gap factor
 8:             assign all employees from team $t$ to team $t^M$                 ▻ extend merged team
 9:             $T^F \leftarrow T^F \setminus \{t\}$                              ▻ remove the considered team from free team list
10:         end while
11:         create $J^{qual}_{t^M}$ for team $t^M$                               ▻ identify the set of jobs for which the merged team is qualified
12:         while $|J_{t^M}| < \lambda^{max}$ and $J^{qual}_{t^M} \neq \emptyset$ do
13:             $J^{temp} \leftarrow J_{t^M} \cup (\lambda^{max} - |J_{t^M}|)$ first jobs from set $J^{qual}_{t^M}$   ▻ build set of temporary jobs
14:             $J^{qual}_{t^M} \leftarrow J^{qual}_{t^M} \setminus J^{temp}$    ▻ remove new considered jobs from $J^{qual}_{t^M}$
15:             solve routing problem (3.36)-(3.40) for team $t^M$
16:             update set $J_{t^M}$ according to obtained route
17:         end while
18:         $J^{un} \leftarrow J^{un} \setminus J_{t^M}$                         ▻ remove assigned jobs from $J^{un}$
19:     end if
20: end for
```

Algorithm 3: Interday Planning: Swapping of Jobs

```
    Input: solution of Algorithm 2
 1: initialize set of teams for job swapping $T^S \leftarrow \{t | t \in T, J_t \neq \emptyset\}$
 2: for $(t, t') \in T^S \times T^S$ with $t < t'$ do
 3:     create sets $J^S_{t',t}$ and $J^S_{t,t'}$                 ▻ job sets that can be moved between teams $t$ and $t'$
 4:     if $J^S_{t',t} \neq \emptyset$ and $J^S_{t,t'} \neq \emptyset$ then
 5:         for $(j^1, j^2) \in J^S_{t',t} \times J^S_{t,t'}$ do
 6:             solve routing problem (3.36)-(3.40) for team $t$ with job set $J^{temp} \leftarrow J_t \setminus \{j^2\} \cup \{j^1\}$
 7:             if $j^1$ is served in the obtained route then
 8:                 solve routing problem (3.36)-(3.40) for team $t'$ with job set $J^{temp'} \leftarrow J_{t'} \setminus \{j^1\} \cup \{j^2\}$
 9:                 if $j^2$ is served in the obtained route and total job completion time is reduced then
10:                     save new schedules for teams $t$ and $t'$
11:                     $J_t \leftarrow J^{temp}$, $J_{t'} \leftarrow J^{temp'}$
12:                 end if
13:             else
14:                 continue with next $j^1$
15:             end if
16:         end for
17:     end if
18: 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
1: $obj^{best} \leftarrow \infty$
2: for $i = 1$ to $I$ do
3:     exchange $N$ employee assignments in $x'_{mt}$
4:     generate an initial solution (Algorithm 1)
5:     improve by splitting and merging of teams (Algorithm 2)
6:     calculate objective value $obj$ of current solution
7:     if $obj < obj^{best}$ then
8:         save new solution
9:         $obj^{best} \leftarrow obj$
10:    end if
11: end for
12: 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. Figure 3.5 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

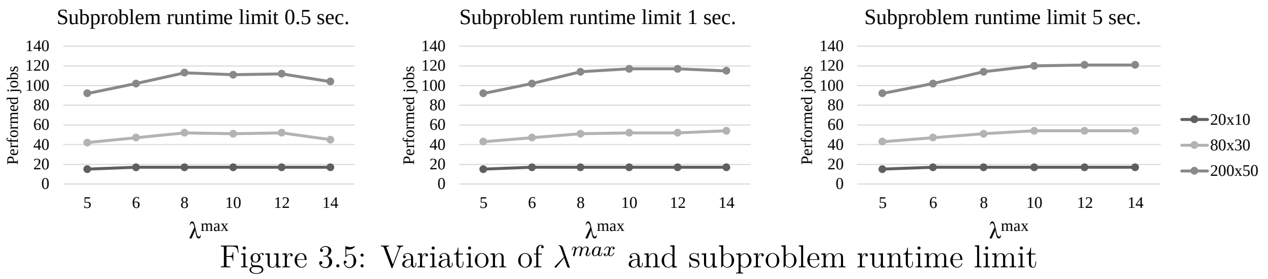


Figure 3.5: Variation of λ^{max} and subproblem runtime limit

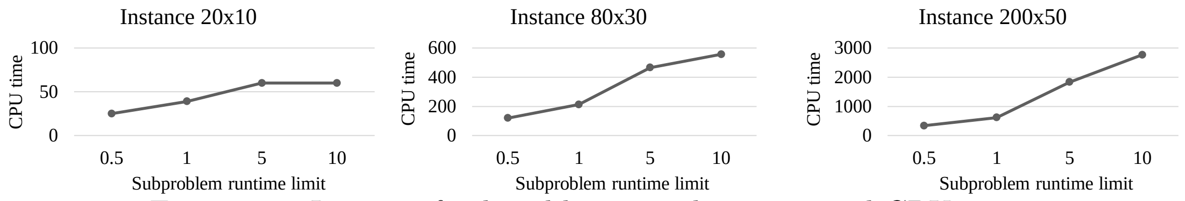


Figure 3.6: Impact of subproblem time limit on total CPU time

for the subproblem has a major influence on the total CPU time needed for solving an instance, see Figure 3.6. 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 Figures 3.5 and 3.6, we find that a good compromise can be reached by setting $\lambda^{max} = 8$ and a runtime limit of 0.5 seconds, which we apply in all further experiments.

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. Figure 3.7 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 $I = 30$.

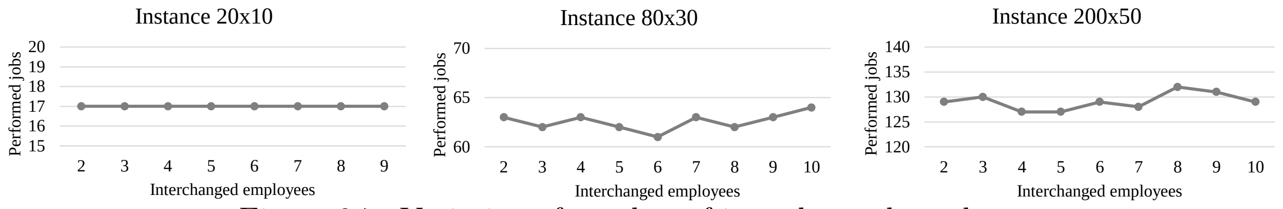


Figure 3.7: Variation of number of interchanged employees

Appendix D. Insertion of High-Priority Jobs

Table 3.8 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 3.8: Insertion of high-priority jobs for instance 8×8

	Baseline Schedule		$\lvert J^h \rvert = 1$		$\lvert J^h \rvert = 3$		$\lvert J^h \rvert = 5$	
	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

Bibliography

Anoshkina, Y. and Meisel, F. (2019). Technician teaming and routing with service-, cost-and fairness-objectives. *Computers & Industrial Engineering*, 135:868–880.

Balas, E. (1989). The prize collecting traveling salesman problem. *NETWORKS*, 19:621–636.

Borenstein, Y., Shah, N., Tsang, E., Dorne, R., and Alsheddy, A. (2010). On the partitioning of dynamic workforce scheduling problem. *Journal of Scheduling*, 13:411–425.

Cappanera, P., Goeveia, L., and Scutellá, M. G. (2013). Models and valid inequalities to asymmetric skill-based routing problems. *EURO Journal on Transportation and Logistics*, 2:29–55.

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

Chen, X., Thomas, B. W., and Hewitt, M. (2017). Multi-period technician scheduling with experience-based service times and stochastic customer. *Computers & Operations Research*, 82:1–17.

Coelho, L. C., Cordeau, J. F., and Laporte, G. (2012). Consistency in multi-vehicle inventory routing. *Transportation Research Part C Emerging Technologies*, 24:270–287.

Cordeau, J.-F., Laporte, G., Pasin, F., and Ropke, S. (2010). Scheduling technicians and tasks in a telecommunication company. *Journal of Scheduling*, 13:393–409.

De Bruecker, P., Van Den Bergh, J., Beliën, J., and Demeulemeester, E. (2015). Workforce planning incorporating skills: State of the art. *Journal of Scheduling*, 13:393–409.

Estellon, B., Gardi, F., and Nouioua, K. (2009). High-performance local search for task scheduling with human resource allocation. *Lecture Notes in Computer Science*, 552:1–15.

Feillet, D., Garaix, T., Lehuédé, F., Péton, O., and Quadri, D. (2014). A new consistent vehicle routing problem for the transportation of people with disabilities. *Networks*, 63:211–224.

Fırat, M., Briskorn, D., and Laugier, A. (2016). A branch-and-price algorithm for stable workforce assignments with hierarchical skills. *European Journal of Operational Research*, 251:676–685.

Firat, M. and Hurkens, C. (2012). An improved MIP-based approach for a multi-skill workforce scheduling problem. *Journal of Scheduling*, 15:363–380.

Gröer, C., Golden, B., and Wasil, E. (2009). The consistent vehicle routing problem. *Manufacturing & Service Operations Management*, 11:630–643.

Hashimoto, H., Boussier, S., Vasquez, M., and Wilbaut, C. (2011). A GRASP-based approach for technicians and interventions scheduling for telecommunications. *Annals of Operations Research*, 183:143–161.

Ho, S. C. and Leung, J. M. Y. (2010). Solving a manpower scheduling problem for airline catering using metaheuristics. *European Journal of Operational Research*, 202:903–921.

Hurkens, C. A. (2009). Incorporating the strength of MIP modeling in schedule construction. *RAIRO-Operations Research*, 43:409–420.

Kalisch, B. J., Begeny, S., and Anderson, C. (2008). The effect of consistent nursing shifts on teamwork and continuity of care. *The Journal of Nursing Administration*, 38:132–137.

Kazirzadeh, A., Saddoune, M., and Soumis, F. (2017). Airline crew scheduling: Models, algorithms and data sets. *EURO Journal of Tranportation and Logistics*, 6:111–137.

Khalfay, A., Crispin, A., and Crockett, K. (2017). Applying the intelligent decision heuristic to solve large scale technician and task scheduling problem. *International Conference on Intelligent Decision Technologies*, 72:71–81.

Kovacs, A. A., Golden, B. L., Hartl, R. F., and Parragh, S. N. (2014). Vehicle routing problem in which consistency consideration are important: A survey. *Networks*, 64:192–213.

Kovacs, A. A., Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2012). Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of Scheduling*, 15:579–600.

Maenhout, B. and Vanhoucke, M. (2018). A perturbation matheuristic for the integrated personnel shit and task re-scheduling problem. *European Journal of Operational Research*, 269:806–823.

Mathlouthi, I., Gendreau, M., and Potvin, J. Y. (2018). Mixed integer programming for a muli-attribute technicians routing and scheduling problem. *Information Systems and Operational Research*, 56:33–49.

Paraskevopoulos, D. C., Laporte, G., Repoussis, P. P., and Tarantilis, C. D. (2016). Resource constrained routing and scheduling: Review and research prospect. *European Journal of Operational Research*, 263:737–754.

Patric, J., Puterman, M. L., and Queyranne, M. (2017). Dynamic multipriority scheduling for a diagnostic resource. *Operations Research*, 56:1507–1525.

Petrakis, I., Hass, C., and Bichler, M. (2012). On the impact of real-time information on field service scheduling. *Decision Support System*, 53:282–293.

Pillac, V., Guéret, C., and Medaglia, A. (2012). On the dynamic technician routing and scheduling problem. *ODYSSEUS 2012 - 5th International Workshop on Freight Transportation and Logistics, May 2012, Mikonos, Greece 194*, 16:1–11.

ROADEF (2007). Technicians and interventions scheduling for telecommunications. `http://www.roadef.org/challenge/2007/en/index.php`.

Russel, D., Rosati, R. J., Rosenfeld, P., and Marren, J. M. (2011). Continuity in home health care: Is consistency in nursing personnel associated with better patient outcomes? *Journal for Healthcare Quately*, 33:33–39.

Siferd, S. P. and Benton, W. C. (1994). A decision model for shift scheduling of nurses. *European Journal of Operational Research*, 74:519–527.

Smilowitz, K., Nowak, M., and Jiang, T. (2013). Workforce management in periodic delivery operations. *Tranportation Science*, 47:214–230.

Van Eck, M. L., Firat, M., Nuijten, W. P. M., Sidorova, N., and Van der Aalst, W. M. P. (2017). Human performance-aware scheduling and routing of a multi-skilled workforce. *Complex Systems Informatics and Modeling Quarterly*, 12:1–21.

Zamorano, E. and Stolletz, R. (2017). Branch-and-price approaches for the multiperiod technician routing and scheduling problem. *European Journal of Operational Research*, 257:55–68.

4 Essay 3

Robust Optimization Approaches for Routing and Scheduling of Multi-Skilled Teams under Uncertain Job Skill Requirements

Yulia Anoshkina[a], Marc, Goerigk[b], Frank Meisel[a]

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

[b] Network and Data Science Management, University of Siegen, Germany

Abstract: We consider a combined problem of teaming and scheduling of multi-skilled employees that have to perform jobs with uncertain qualification requirements. We propose two modeling approaches that generate solutions that are robust to possible data variations. Both approaches use variants of budgeted uncertainty, where deviations in qualification requirements are bounded by a constraint. In the first approach, we aggregate uncertain constraints to ensure that the total number of job qualifications present at a job is not less than a worst-case value. We show that these values can be computed beforehand, resulting in a robust model with little additional complexity compared with the nominal model. In our second approach, we bound the overall qualification deviation over all jobs. While this approach is more complex, we show that it is still possible to derive a compact problem formulation by using a linear programming formulation for the adversarial problem based on a dynamic program. The performance of both approaches is analyzed on a test bed of instances which were originally provided for a deterministic problem version. Our experiments show the effectiveness of the proposed approaches in the presence of data uncertainty and reveal the price and gain of robustness.

Keywords: Multi-Skilled Workforce Scheduling, Robust Optimization, Budgeted Uncertainty

4.1 Introduction

This paper addresses a combined problem of routing and scheduling of multi-skilled workforce as it is faced by many service-oriented companies that provide installation, construction, maintenance or delivery services at customer locations. Each service job to conduct requires employees with different skill domains and at different levels of expertise. Therefore, teams of technicians have to be formed according to job qualification requirements that express the number of employees with specific skills and required experience in the corresponding domains. In order to increase productivity and to decrease labor costs, companies may prefer to hire multi-skilled employees that can be easily assigned to various jobs as required. This provides more flexibility and allows a company to focus on customer satisfaction. As teams may be capable of serving multiple jobs, optimal routing plans have to be found for the formed teams. From this, the investigated Routing and Scheduling Problem of Multi-Skilled Teams (CTRP) can be considered as an extension of the Vehicle Routing Problem (VRP). Due to its practical relevance, the CTRP has gained an increasing attention during the last decade and has been investigated extensively from different perspectives. In this paper, we demonstrate how the CTRP can be solved in the presence of data uncertainty. In general, different sources for data variations are existent but we focus here on uncertainty of the qualification requirements of a job. Such qualification requirements are usually derived from communication with customers. As a customer is not necessarily an expert in the corresponding field, the required skill types and levels of competence for executing a job may be wrongly assessed by the customer when issuing the job. Also the company may misproject these requirements before having executed the job due to a lack of information. Moreover, the routing decisions can be affected by variations of travel times due to traffic conditions or by delays in job processing due to differing employee working speeds. In light of these findings, it becomes important to have a robust planning approach that ensures solution reliability also in the presence of possible data variations. However, despite the substantial progress in the field of robust optimization, we are not aware of any approaches that have been so far presented for the formation of worker teams and their job-routing as is addressed in this paper.

Recently, Anoshkina and Meisel (2019) analyzed the deterministic version of the CTRP and the potential of decomposition techniques for reducing the complexity of the planning. Furthermore, CTRP was considered from a multi-period perspective by additionally emphasizing team consistency in Anoshkina and Meisel (2020). In the present study, we take the first step to deal with data variations in the context of scheduling of multi-skilled teams. Namely, we concentrate on developing a linear optimization framework incorporating *demand uncertainty* which we define as a variation of job qualification requirements. Our contribution is then threefold: (i) We propose a first robust model formulation based on the concept of budgeted uncertainty sets as proposed by Bertsimas and Sim (2003), where the uncertainty affects each job independently. (ii) We propose a second robust model formulation, where the uncertainty is restricted by a global constraint over all jobs. (iii) We test extensively the model performance under the two different robustness strategies by analyzing the impact of the uncertain demand on the feasibility and quality of solution.

The outline of this paper is as follows. In Section 4.2, we review the relevant literature on workforce teaming and scheduling. We also discuss important robustness concepts that constitute the foundation of our later investigation. In Section 4.3, we present a mathematical formulation of the deterministic problem version. In Section 4.4, we develop two robust optimization models based on different budgeted uncertainty sets. Section 4.5 presents experimental results and analyzes the performance of the two robustness strategies. Finally, Section 4.6 concludes the paper and outlines future research.

4.2 Literature

The combined problem of teaming and scheduling of a multi-skilled workforce was first addressed in the works of Estellon et al. (2009), Hurkens (2009), Cordeau et al. (2010), Hashimoto et al. (2011), and Fırat and Hurkens (2012). The initial focus of the research has been on scheduling aspects. Specifically, it was considered how multi-skilled employees can be grouped into teams and assigned to a set of jobs where jobs require multiple skills at different competence levels. Following these initial contributions, an increasing number of extensions has been presented. For instance, such features as routing of teams (Kovacs et al. (2012)), multi-period planning (Zamorano and Stolletz (2017)), employee preferences for performing a specific job (Fırat et al. (2016)), alternative heuristic solution methods (Khalfay et al. (2017)), decomposition techniques (Anoshkina and Meisel (2019)) as well as team consistency and rescheduling (Anoshkina and Meisel (2020)) have been studied. A more detailed description of these studies is provided by Anoshkina and Meisel (2020). Throughout, the authors assumed a deterministic setting where all input data is completely known with certainty.

However, real-world situations typically involve data uncertainty. Therefore, considerable research has been conducted in developing robust programs that find solutions which perform well despite variations in the input data. A substantial progress in the theory of robust optimization has been achieved with concepts presented by Ben-Tal and Nemirovski (1999) and Bertsimas and Sim (2004). More precisely, Ben-Tal and Nemirovski (1999) showed that robust counterparts of linear programs with ellipsoidal uncertainty set are computationally tractable and can be solved as conic quadratic problems. Bertsimas and Sim (2004) developed the concept of budgeted uncertainty that enabled to reformulate non-linear robust constraints as linear functions. For general surveys on robust optimization, we refer the interested reader to Buchheim and Kurtz (2018); Gabrel et al. (2014); Goerigk and Schöbel (2016).

The mentioned approaches have opened an avenue for research in many optimization areas. For instance, Sungur et al. (2008) addressed a VRP with uncertain customer demand and proposed three robust formulations based on convex, box and ellipsoidal uncertainty sets. Following the idea of Bertsimas and Sim (2003), Ordóñez (2010) presented a robust formulation of the VRP incorporating two additional sources of uncertainty occurring in travel time and travel costs. This line of research was continued for a number of problem extensions. For instance, Lee et al. (2012) addressed a VRP with deadlines and uncertainty arising in travel times and customer demand. Han et al. (2014) combined stochastic programming with robust optimization for the solution of a VRP with uncertain travel times where penalties are imposed if travel time exceeds a preset time limit. Demand uncertainty was also studied by Cao et al. (2014) in the context of open VRPs where vehicles do not necessarily return to the depot after delivering goods. Chen et al. (2016) analyzed a routing problem arising in road maintenance, in which each part of a road network has to be monitored by a service vehicle. Thereby, service times are subject to uncertainty due to various factors like road conditions or accidents. De La Vega et al. (2019) investigated a VRP with time windows (VRPTW) and multiple delivery men where a specific number of workers is required to execute deliveries and customer demand becomes known only when a vehicle arrives at a customer location. A further VRPTW model with both demand and travel time uncertainty was provided by Munari et al. (2019) where the authors used a two-index vehicle flow formulation. The main advantage of this formulation is that the robust counterpart can be derived directly from the underlying deterministic model and, thus, does not require additional constraints associated with uncertain parameters.

Compared with the large number of studies addressing robust VRPs, relatively few papers have been published on robust personnel scheduling. Carello and Lanzarone (2014) developed a robust optimization model for a home health care problem with demand uncertainty and continuity of care. Continuity of care means that all services required by a patient are provided by the same specialist over a long period. The demand is considered uncertain due to possible variations in the physical conditions of patients. Nguyen and Montemanni (2016) proposed a nonlinear mixed-integer programming formulation for taking into account uncertainty in nurse availability. Souyris et al. (2013) examined the problem of dispatching technicians under stochastic service times. Specifically, the authors developed two different solution concepts distinguishing between processing time uncertainty related to customers or to technicians. Finally, we are aware of only one robust optimization approach dealing with scheduling of multi-skilled employees where the workforce demand is subject to uncertainty. At the example of a service industry company, Henao et al. (2016) investigate how multi-skilled employees can be effectively distributed between departments over a planning horizon of one week. Thereby, the problem also incorporates decisions about training of employees specialized only in one domain. The goal is to minimize staff training, shortage and surplus costs. Operations management for the assignment of jobs to teams and routing decisions, as is considered in our paper, are out of scope of their study.

As results generated by robust programs can deviate significantly from deterministic solutions, a further stream of research focuses on methods and algorithms that reduce this so-called price of robustness. Complementing the work of Bertsimas and Sim (2003), Poss (2013) presented the concept of variable budgeted uncertainty, where dualization techniques are applied to more general uncertainty polytopes. It was shown by experiments that the proposed approach can yield better results and reduces the price of robustness by 18%. Furthermore, the robust optimization methodology was extended by a class of two-stage robust optimization concepts, see e.g. Adjiashvili et al. (2015); Ben-Tal et al. (2004); Buchheim and Kurtz (2017); Hanasusanto et al. (2015); Liebchen et al. (2009). These approaches consider problems where decisions can be taken sequentially. Therefore, a subset of decisions is implemented before the specific data realization becomes known whereas the remaining decisions can be taken after the uncertainty has resolved. For a general survey on these approaches, we refer to Yanıkoğlu et al. (2019).

To the best of our knowledge none of the mentioned concepts has been applied so far to the CTRP. To close this gap, we develop two alternative robust optimization models that incorporate demand uncertainty in the context of routing and scheduling of multi-skilled teams.

4.3 Deterministic Model (DM)

The deterministic (nominal) version of the multi-skilled workforce routing and scheduling problem can be described as follows. We are given a set of employees M and a set of jobs J as well as an extension of this set as $J^0 = \{0\} \cup J = \{0, 1, \ldots, |J|\}$ where 0 refers to a depot. Each job $j \in J$ is characterized by a service requirement r_{jkl} that gives the number of employees with qualification in skill $k \in K$ and experience level $l \in L$ required for performing job j. Here, K denotes the set of skill domains and L the set of experience levels. The competences of employee $m \in M$ are described by a binary matrix q_{mkl} where an element takes value 1 if the employee is qualified in skill $k \in K$ at level $l \in L$ and 0 otherwise. As each job can require more than one employee, employees have to be grouped into teams in order to meet a job's qualification requirements r_{jkl} for all $k \in K$ and $l \in L$. The maximal number of teams T to build is specified by the minimum of the number of employees and the number of jobs considered in a problem instance. More precisely, if we consider a problem with 10 employees and 5 jobs, at most $T = min\{10, 5\}$ teams are required (or can be built). Note that each job $j \in J$ has to be carried out by exactly one team, whereas a team might perform several jobs one after the other. Thereby, the completion time of each job cannot be later than a maximal working time e_{max} that is given for each team. Further, all services associated with job j are provided at the customer's location. To each pair of jobs $(i, j) \in J^0 \times J^0$, we thus assign a travel time d_{ij} that is needed by a team to go from i to j. Additionally, each job j has a processing time p_j that indicates the amount of time that a team has to stay at customer location j. Here, we assume that p_j is given and constant, i.e. p_j does not depend on the team composition or working environment.

A corresponding deterministic model has been provided by Anoshkina and Meisel (2020). Here we present a slightly modified formulation that constitutes the foundation of our robust approach. The formulation uses the following decision variables. The binary decision variable x_{mt} indicates if employee m is assigned to team t or not. The routes of teams are denoted by binary decision variables z_{tij} that define if team t travels directly from job i to job j or not. The continuous scheduling variable s_{tj} specifies the start time of job j by team t. Similar, f_{tj} denotes the completion time of job j executed by team t. Using the introduced notation the deterministic model is formulated as follows.

$$\text{maximize: } \alpha \cdot \sum_{t \in T} \sum_{j \in J^0} \sum_{j \in J} z_{tij} - \beta \cdot \sum_{t \in T} \sum_{j \in J} f_{tj} \tag{4.1}$$

subject to:

$$\sum_{t \in T} x_{mt} \leqslant 1 \qquad \forall m \in M \tag{4.2}$$

$$\sum_{m \in M} x_{mt} \cdot q_{mkl} \geqslant r_{jkl} \cdot \sum_{i \in J^0} z_{tij} \qquad \forall j \in J, k \in K, l \in L, t \in T \tag{4.3}$$

$$\sum_{j \in J} z_{t0j} \leqslant 1 \qquad \forall t \in T \tag{4.4}$$

$$\sum_{t \in T} \sum_{i \in J^0} z_{tij} \leqslant 1 \qquad \forall j \in J \tag{4.5}$$

$$\sum_{i \in J^0} z_{tij} = \sum_{i \in J^0} z_{tji} \qquad \forall j \in J^0, t \in T \tag{4.6}$$

$$f_{ti} + d_{ij} \leqslant s_{tj} + \mathcal{M} \cdot \left(1 - z_{tij}\right) \qquad \forall i \in J^0, j \in J, t \in T \tag{4.7}$$

$$s_{tj} + p_j \leqslant f_{tj} + \mathcal{M} \cdot \left(1 - \sum_{i \in J^0} z_{tij}\right) \qquad \forall j \in J, t \in T \tag{4.8}$$

$$f_{tj} \leqslant e_{max} \qquad \forall j \in J, t \in T \tag{4.9}$$

$$s_{tj}, f_{tj} \geqslant 0 \qquad \forall j \in J^0, t \in T \tag{4.10}$$

$$x_{mt}, z_{tij} \in \{0, 1\} \qquad \forall i, j \in J^0, m \in M, t \in T \tag{4.11}$$

The main goal of the model is to maximize the service level, which we define as the number of performed jobs. Minimization of the total job completion time is considered as a subordinate objective. Weights α and β are used for expressing different priorities of these two objectives. Constraints (4.2) forbid to assign an employee to more than one team. Constraints (4.3) guarantee that each formed team t has an appropriate qualification for processing all jobs that are assigned to this team. Constraints (4.4) indicate that each active team starts from the depot. Constraints (4.5) impose that each job can be served by at most one team. Constraints (4.6) are the flow balancing constraints. Constraints (4.7)-(4.8) define the start and completion times of job j performed by team t. Here, $\mathcal{M}$ denotes a sufficiently large positive value. Note that Constraints (4.7)-(4.8) also prevent subtours in the solution. Constraints (4.9) bound the longest working time for all teams. Constraints (4.10)-(4.11) specify the domains of decision variables.

4.4 Robust Formulations with Uncertain Job Qualification Requirements

We present two robust problem formulations, which are based on different models to treat uncertainty in skill demand. For the ease of notation, we denote the two models by **RM1** (first robust model) and **RM2** (second robust model), respectively.

4.4.1 First Robust Model (RM1)

Our aim is to generate solutions that are insensitive to demand deviations. By demand deviation, we understand the variation of skill vectors in a job requirement matrix r_{jkl}. In the deterministic model presented in Section 4.3, only Constraints (4.3) are affected by the variation of job skill requirements r_{jkl}. Note that only one element of r_{jkl} is examined in each qualification constraint. In the standard technique for robust optimization (Bertsimas and Sim (2004)), we require constraint-wise uncertainty instead. Otherwise, we would hedge against the worst case in each parameter, which results in overly conservative solutions. To avoid this conservatism, we follow the approach of Bohle et al. (2010) and Henao et al. (2016) and extend the original deterministic model by redundant constraints expressing the aggregated qualification requirement, which we compute as the sum of technicians in all skill domains required on all levels of competence:

$$\sum_{m \in M} \sum_{k \in K} \sum_{l \in L} x_{mt} \cdot q_{mkl} \geqslant \sum_{k \in K} \sum_{l \in L} r_{jkl} \cdot \sum_{i \in J^0} z_{tij} \qquad \forall j \in J, t \in T \qquad (4.12)$$

We model the uncertain demand $\tilde{r}_{jkl}$ for all $j \in J$ as an independent, random variable bounded on the interval $\tilde{r}_{jkl} \in [r_{jkl}, r_{jkl} + \hat{r}_{jkl}]$, where r_{jkl} denotes the nominal value and $\hat{r}_{jkl}$ the maximal deviation allowed for r_{jkl}. For each random variable $\tilde{r}_{jkl}$, we define a level of variability ζ_{jkl} ranging within $[0, 1]$. From this, the skill requirement variation is formulated as follows:

$$\tilde{r}_{jkl} = r_{jkl} + \hat{r}_{jkl} \cdot \zeta_{jkl} \qquad \forall j \in J, k \in K, l \in L \qquad (4.13)$$

Furthermore, we assume that any skill and any qualification level can be exposed to uncertainty. The level of uncertainty for a job j is controlled by parameter $\Gamma_j \in \mathbb{N}$ that presets the maximum skill and experience deviation allowed for this job. More precisely, Γ_j represents an upper bound on the sum of skill and experience deviation weights ζ_{jkl} over all skill domains $k \in K$ and levels $l \in L$. From this, Γ_j serves to adjust the robustness of the solution against the level of conservatism of a decision maker (Bertsimas and Sim (2003)). For instance, if $\Gamma_j = 0$, a decision maker assumes that no element of r_{jkl} is likely to change. This corresponds to a risk seeking attitude where no protection against demand uncertainty is incorporated in the planning. In contrast, $\Gamma_j = |K| \cdot |L|$ assumes that all elements of r_{jkl} are subject to uncertainty, which corresponds to a very risk averse decision maker. This guarantees the maximal level of protection against all possible variations but, at the same time, results in the most conservative solution. Based on the previous notation, the uncertainty set for each job U_j^Γ is defined as follows:

$$U_j^\Gamma = \left\{ \tilde{r}_j \in \mathbb{R}^{|K|\cdot|L|} \mid \tilde{r}_{jkl} = r_{jkl} + \hat{r}_{jkl} \cdot \zeta_{jkl},\ 0 \leqslant \zeta_{jkl} \leqslant 1\ \forall k \in K, l \in L,\ \sum_{k\in K}\sum_{l\in L} \zeta_{jkl} \leqslant \Gamma_j \right\}$$

The aim of the robust model is to find solutions that remain feasible for all possible qualification requirements $\tilde{r}_j \in U_j^\Gamma$ for each job j.

Using the uncertainty set U_j^Γ, the robust counterpart of (4.12) can be formulated as

$$\sum_{m\in M}\sum_{k\in K}\sum_{l\in L} x_{mt} \cdot q_{mkl} \geqslant \sum_{k\in K}\sum_{l\in L} \tilde{r}_j \cdot \sum_{i\in J^0} z_{tij} \qquad \forall j \in J, \tilde{r}_j \in U_j^\Gamma, t \in T \tag{4.14}$$

Formulation (4.14) is intractable in its current form since it contains an infinite number of constraints for all realizations of the continuous parameters ζ_{jkl} within the uncertainty set U_j^Γ. To approach this issue, note that $\sum_{i\in J^0} z_{tij}$ is either 0 or 1. Hence, there are only two cases we need to consider: If $\sum_{i\in J^0} z_{tij} = 0$, constraint (4.12) is always fulfilled. If $\sum_{i\in J^0} z_{tij} = 1$, then we need to calculate the maximum value that $\sum_{k\in K}\sum_{l\in L} \tilde{r}_{jkl}$ can possibly take. Denoting this value by $\bar{r}_j$, constraint (4.14) becomes

$$\sum_{m\in M}\sum_{k\in K}\sum_{l\in L} x_{mt} \cdot q_{mkl} \geqslant \bar{r}_j \cdot \sum_{i\in J^0} z_{tij} \qquad \forall j \in J, t \in T \tag{4.15}$$

To calculate $\bar{r}_j$, we need to solve the problem

$$\begin{aligned} \bar{r}_j &= \max \left\{ \sum_{k\in K}\sum_{l\in L} (r_{jkl} + \hat{r}_{jkl} \cdot \zeta_{jkl}) : \sum_{k\in K}\sum_{l\in L} \zeta_{jkl} \leqslant \Gamma_j,\ 0 \leqslant \zeta_{jkl} \leqslant 1\ \forall k \in K, l \in L \right\} \\ &= \sum_{k\in K}\sum_{l\in L} r_{jkl} + \max \left\{ \hat{r}_{jkl} \cdot \zeta_{jkl} : \sum_{k\in K}\sum_{l\in L} \zeta_{jkl} \leqslant \Gamma_j,\ 0 \leqslant \zeta_{jkl} \leqslant 1\ \forall k \in K, l \in L \right\} \end{aligned}$$

Calculating this value can be done by sorting the vector $\hat{r}_{jkl}$ in descending order, and then summing up the Γ_j many largest values. Then, the robust counterpart of the nominal model becomes

$$\text{maximize: } \alpha \cdot \sum_{t\in T}\sum_{i\in J^0}\sum_{j\in J} z_{tij} - \beta \cdot \sum_{t\in T}\sum_{j\in J} f_{tj} \tag{4.16}$$

subject to: (4.2) – (4.11) and

$$\sum_{m\in M}\sum_{k\in K}\sum_{l\in L} x_{mt} \cdot q_{mkl} \geqslant \bar{r}_j \cdot \sum_{i\in J^0} z_{tij} \qquad \forall j \in J, t \in T \tag{4.17}$$

We further extend this model by measuring how much the required right-hand side constraint (4.17) is exceeded. This excess creates an additional benefit for the objective function, i.e., we reward additional robustness in the solution with some weight μ. To this end, we introduce a new variable ρ_{jt} that measures the slack of the right-hand side. This yields model **RM1**:

$$\text{maximize: } \alpha \cdot \sum_{t\in T}\sum_{i\in J^0}\sum_{j\in J} z_{tij} - \beta \cdot \sum_{t\in T}\sum_{j\in J} f_{tj} + \mu \cdot \sum_{t\in T}\sum_{j\in J} \rho_{jt} \tag{4.18}$$

subject to: (4.2) – (4.11) and

$$\sum_{m\in M}\sum_{k\in K}\sum_{l\in L} x_{mt} \cdot q_{mkl} \geqslant \bar{r}_j \cdot \sum_{i\in J^0} z_{tij} + \rho_{jt} \qquad \forall j \in J, t \in T \tag{4.19}$$

$$\rho_{jt} \leqslant \mathcal{M} \cdot \sum_{i\in J^0} z_{tij} \qquad \forall j \in J, t \in T \tag{4.20}$$

$$\rho_{jt} \geqslant 0 \qquad \forall j \in J, t \in T \tag{4.21}$$

The additional constraint (4.20) is required to ensure that the excess is only taken into account if the job j is actually performed by team t.

4.4.2 Second Robust Model (RM2)

In the previous model, uncertainty sets were applied job-wise, which makes it possible to find a robust counterpart with little computational overhead. It has the drawback that solutions may still become overly conservative, as worst-case scenarios are assumed for each job separately. Furthermore, the aggregation of constraints means that it is ignored with what skills we hedge against uncertainty, as long as the total number of skills present is sufficient. We now follow a more nuanced approach to model uncertainty, which avoids both problems.

Consider the skill requirement r_{jkl} for job j, skill k, level l. Let us assume we build a team that reaches a qualification level $\sum_{m\in M} x_{mt} \cdot q_{mkl}$. The buffer is then defined as $b_{jkl} = \sum_{m\in M} x_{mt} \cdot q_{mkl} - r_{jkl}$.

Let us assume there is an adversary who tries to find a scenario to disrupt as many jobs as possible. The adversary can increase the required skill level r_{jkl} under the following conditions: increasing r_{jkl} by one unit has some cost c_{jkl}, which reflects that higher level skills are less likely than lower level skills (c_{jkl} increases with l) and that it should be more expensive to increase the demand of skills k that are less likely to be relevant as judged by expert knowledge. The adversary has a global budget Γ he can use for skill requirement increases. A job is disrupted if the requirements in one skill and level are not met.

Given a fixed team and schedule, we hence want to solve the following adversary problem:

$$\text{maximize:} \quad \sum_{j\in J} \left(\sum_{t\in T} \sum_{i\in J^0} z_{tij} \right) \cdot \zeta_j \tag{4.22}$$

$$\text{subject to:} \quad \zeta_j \leqslant \sum_{k\in K} \sum_{l\in L} \zeta_{jkl} \qquad \forall j \in J \tag{4.23}$$

$$\sum_{j\in J} \sum_{k\in K} \sum_{l\in L} (b_{jkl} + 1) \cdot c_{jkl} \cdot \zeta_{jkl} \leqslant \Gamma \tag{4.24}$$

$$\zeta_j \in \{0,1\} \qquad \forall j \in J \tag{4.25}$$

$$\zeta_{jkl} \in \{0,1\} \qquad \forall j \in J, k \in K, l \in L \tag{4.26}$$

Here, binary variable ζ_{jkl} indicates if job j is prevented by increasing the requirements in skill k at level l. Binary variable ζ_j indicates if job j is prevented overall. Using constriant (4.23), we enforce that ζ_j can only be active if at least one of the ζ_{jkl} variables is active as well. Constraint (4.24) ensures that the total budget is restrcited to Γ, where the costs on the left-hand side correspond to the required investment to make a job infeasible. Note that in an optimal solution, one would not increase multiple variables ζ_{jkl} for the same job j, but only choose the cheapest possibility. As the buffers b_{jkl} depend on the assignment x_{mt}, we do not remove these variables from the problem.

Unfortunately, it is not possible to relax this formulation of the adversarial problem without changing its objective value. This means that a compact robust formulation cannot be obtained by simply dualizing the linear relaxation of the adversarial problem. In the following, we show that a compact formulation can still be obtained by using a dynamic programming formulation.

Let us denote by $F(j, \gamma)$ the maximum number of jobs from $j' \in \{0, \dots, j\}$ that can be interrupted with a budget $\gamma \in \{0, \dots, \Gamma\} =: \Gamma^0$. We have $F(0, \gamma) = 0$ for all $\gamma \in \Gamma^0$, and the recursion

$$F(j, \gamma) = \max \left\{ F(j-1, \gamma), 1 + F(j-1, \gamma - \min_{k,l}(b_{jkl} + 1) \cdot c_{jkl}) \right\}$$

The value $F(|J|, \Gamma)$ is then equal to the objective value of the adversarial problem. We can also see this dynamic program as a longest path problem. We define a set of nodes $V = J^0 \times \Gamma^0$ and arcs $A = \{(j, \gamma, j', \gamma') \in V \times V : j' > j, \gamma' > \gamma\}$. The adversary problem is then equivalent to solving

$$\text{maximize:} \quad \sum_{a=(j,\gamma,j',\gamma') \in A} \left(\sum_{t \in T} \sum_{i \in J^0} z_{tij'} \right) \cdot \tilde{c}_a p_a \tag{4.27}$$

$$\text{subject to: } p \text{ is a path from } (0,0) \text{ to } (|J|, \Gamma) \tag{4.28}$$

where

$$\tilde{c}_{j\gamma,j'\gamma'} = \begin{cases} 1 & \text{if } \exists k,l : (b_{j'kl} + 1) \cdot c_{j'kl} \leqslant \gamma' - \gamma \\ 0 & \text{else} \end{cases}$$

Dualizing this gives the model

$$\text{minimize: } u_{|J|,\Gamma} \tag{4.29}$$

$$\text{subject to: } u_{00} = 0 \tag{4.30}$$

$$u_{j'\gamma'} \geqslant u_{j\gamma} + \left(\sum_{t\in T}\sum_{i\in J^0} z_{tij'}\right)\cdot \tilde{c}_{j,\gamma,j',\gamma'} \qquad \forall (j,\gamma,j',\gamma') \in V \times V : j' > j, \gamma' > \gamma \tag{4.31}$$

Combining this dual adversarial model with the deterministic formulation, we obtain the following compact formulation **RM2** for the robust RSPMST:

$$\text{maximize: } \alpha \cdot \sum_{t\in T}\sum_{j\in J^0}\sum_{j\in J} z_{tij} - \nu \cdot u_{|J|\Gamma} + \mu \cdot \sum_{j\in J}\sum_{k\in K}\sum_{l\in L}\sum_{t\in T} \rho_{jklt} - \beta \cdot \sum_{t\in T}\sum_{j\in J} f_{tj} \tag{4.32}$$

subject to: (4.2), (4.4) – (4.9) and

$$\sum_{m\in M} x_{mt}\cdot q_{mkl} \geqslant r_{jkl}\cdot \sum_{i\in J^0} z_{tij} + \rho_{jklt} \qquad \forall j \in J, k \in K, l \in L, t \in T \tag{4.33}$$

$$\rho_{jklt} \leqslant \mathcal{M}\cdot \sum_{i\in J^0} z_{tij} \qquad \forall j \in J, k \in K, l \in L, t \in T \tag{4.34}$$

$$u_{00} = 0 \tag{4.35}$$

$$u_{j'\gamma'} \geqslant u_{j\gamma} + v_{j',\gamma'-\gamma} \qquad \forall j' \in J, j \in J^0 : j' > j, \; \gamma', \gamma \in \Gamma^0 : \gamma' \geqslant \gamma \tag{4.36}$$

$$\mathcal{M}\cdot (v_{j\gamma} + (1 - \sum_{i\in J^0} z_{tij})) \geqslant \gamma - (\sum_{m\in M} q_{mkl}\cdot x_{mt} - r_{jkl} + 1)\cdot c_{jkl} + 1 \qquad \forall \gamma \in \Gamma^0, j \in J, k \in K, l \in L, t \in T \tag{4.37}$$

$$s_{tj}, f_{tj}, \rho_{jklt} \geqslant 0 \qquad \forall j \in J^0, t \in T, k \in K, l \in L \tag{4.38}$$

$$x_{mt}, z_{tij} \in \{0,1\} \qquad \forall i,j \in J^0, m \in M, t \in T \tag{4.39}$$

$$v_{j\gamma} \in \{0,1\} \qquad \forall j \in J, \gamma \in \Gamma^0 \tag{4.40}$$

$$u_{j\gamma} \geqslant 0 \qquad \forall j \in J^0, \gamma \in \Gamma^0 \tag{4.41}$$

The objective function consists of four components. The first component is to maximize the number of jobs that are taken on. This is reduced by the number of jobs that can be interrupted by the adversary, weighed with a factor ν. With a factor ν slightly smaller than one, we ensure that it is better to plan a job and then have it canceled, than not planning the job at all. The third component is to maximize buffer sizes, similar to model **RM1**. The last component is the travel time. Constraints (4.33)-(4.34) are modified qualification requirements. Constraints (4.35)-(4.37) are used to calculate $u_{|J|\Gamma}$, the number of interrupted jobs. To this end, the binary variable $v_{j\gamma}$ is forced to be 1 if $\sum_{t\in T}\sum_{i\in J^0} z_{tij} = 1$ (i.e., the job is being taken) and γ is sufficiently large to disrupt job j.

4.5 Computational Study

In this section, we describe the results of a computational study that aims at comparing the performance of the models described in Sections 4.3 and 4.4 where we explore the effect of robust planning on the scheduling decisions. Next, we describe our experimental setup followed by a presentation of the obtained results.

4.5.1 Experimental Setup

Our experiments are based on the 12 instance sets of Anoshkina and Meisel (2019). Each instance set contains 10 instances and is distinguished according to the number of jobs and available employees. The first set contains small instances with 4 jobs and 4 employees each, while the last set includes large instances with 20 jobs and 20 employees each. All instances are available online at `www.scm.bwl.uni-kiel.de/de/forschung/research-data`.

In order to estimate the extent to which the skill variations can impact the solution quality, we use employee and job qualification matrices with $|K| = 3$ skills and $|L| = 3$ skill levels. From this, the maximum possible scaled skill deviation for each job in **RM1** is $\Gamma_j = 3 \cdot 3 = 9$. In our experiments, we limit Γ_j for all jobs to value $\Gamma_j = 4$, which corresponds to a medium level of risk aversion. In contrast, the uncertainty budget Γ for **RM2** has to be defined individually for all instances and instance sets. Therefore, preliminary experiments were conducted to estimate Γ manually. For **RM1**, we set the maximal skill deviation for each job as $\hat{r}_{jk1} = 2$, $\hat{r}_{jk2} = 1$ and $\hat{r}_{jk3} = 1$. To evaluate the skill deviation for **RM2**, we define the cost matrix c_{jkl} randomly as follows. The cost of increasing the skill requirement at level $l = 1$ are set to 1 and 2 with an equal probability. Thereby, $c_{jk1} = 1$ means that the corresponding r_{jkl} element is more likely to be changed. Similar, $c_{jk2} \in \{3, 4\}$ and $c_{jk3} \in \{5, 6\}$. The maximal working time e_{max} is set to 540 minutes for all instances and models. Putting emphasis on the service quality, we use the following parameters for evaluating the objective functions: $\alpha = 1$, $\beta = 0.0001$, $\mu = 0.01$ and $\nu = 0.99$.

All tests have been run on an Intel(R) Core (TM) i7-8700 3.20 GHz with 32 GB of RAM. We used CPLEX 12.10 for solving the mixed-integer programming models using a runtime limit of 3600 seconds per instance.

4.5.2 Price of Robustness

The first experiment is conducted to test the extent to which the proposed linear models can be solved to optimality and to examine the effect of the robust planning on scheduling decisions. In particular, we analyze the so-called price of robustness indicating the extent to which the optimal robust solution differs from the non-robust deterministic solution. As performance measure, we consider the difference in the achieved service levels, which we associate with the number and the complexity of performed jobs.

Table 4.1 reports average results for all instance sets and each modeling approach obtained by CPLEX. The first column of the table shows the problem size. The next five columns display results for the deterministic optimization model from Section 4.3, where the reported values are averages for the solutions of 10 instances in the corresponding instance set. The first column Z shows the number of performed jobs. The second column C indicates the average complexity of performed jobs. We define the job complexity as the average required skill in all skill domains and at all levels of competence

Table 4.1: Performance metrics for price of robustness

Instance	**DM**					**RM1**					**RM2**				
$\|J\| \times \|M\|$	Z	C	T	E	F	Z	C	T	E	F	Z	C	T	E	F
4 x 4	2.1	5.1	1.8	3.6	692	1.7	3.9	1.3	4.0	591	1.5	3.7	1.0	4.0	514
4 x 8	3.2	5.0	3.1	7.8	1116	2.9	4.6	2.5	8.0	1055	2.1	4.2	1.6	8.0	765
6 x 6	3.7	4.6	3.1	5.5	1329	2.8	4.0	1.8	6.0	1047	2.7	4.1	1.8	6.0	1010
6 x 12	4.8	5.0	4.7	11.0	1706	4.6	4.9	3.6	12.0	1787	3.9	4.5	2.9	12.0	1506
8 x 6	4.4	4.8	3.3	5.6	1553	3.3	4.0	2.2	6.0	1215	3.1	4.5	1.8	6.0	1136
8 x 12	6.2	5.1	5.7	11.5	2206	5.5	4.8	4.1	12.0	2062	5.0	4.6	3.5	12.0	1892
10 x 7	5.1	4.9	3.8	6.7	1750	3.8	4.5	2.4	7.0	1366	4.2	4.7	2.4	7.0	1574
10 x 13	7.7	5.5	6.4	12.5	2802	6.5	4.7	4.6	13.0	2443	6.1	5.2	4.1	12.9	2327
15 x 8	6.7	4.9	4.6	8.0	2376	4.8	3.7	3.0	8.0	1738	4.1	5.1	2.4	7.9	1500
15 x 15	10.3	5.4	7.6	14.6	3897	8.2	4.8	5.0	15.0	3115	6.4	5.3	4.0	14.9	2452
20 x 10	9.0	4.7	6.0	9.9	3157	6.8	4.3	3.5	10.0	2456	6.6	4.3	4.2	10.0	2423
20 x 20	13.7	5.5	10.4	19.9	5001	10.9	4.6	6.8	20.0	4037	8.3	4.9	5.3	19.8	3057

for those jobs that are processed in a solution, i.e., $C = \sum_{j \in J^{sol}} \sum_{k \in K} \sum_{l \in L} r_{jkl}/|J^{sol}|$, where $J^{sol} \subset J$ denotes the jobs of the solution. Further, columns T and E specify the number of active teams in the route plans and the number of employees assigned to these teams. The next column F gives the total job completion time. The corresponding results for the robust optimization models are presented in the middle and at the right of the table.

Based on Table 4.1, the following differences in the performance of the models can be observed. As expected, we see that **DM** generates many solutions with a higher service level than (column Z) **RM1** and **RM2**. This is because **DM** considers only nominal qualification requirements without taking risks of data variation into account. Thereby, we see that the number of performed jobs increases for instances with a larger number of available employees ($|J| < |M|$). In general, **RM2** is more conservative as the service level achieved under **RM2** is slightly lower for nearly all instances, compared to **RM1**. Another aspect is the complexity of the performed jobs. Looking at columns C, we see a clear tendency for **RM1** and **RM2** to avoid an assignment of more challenging jobs. However, no direct correlation can be derived. Comparing both models, we observe lower (see e.g. instances 4×4, 4×8 or 6×12) as well as higher complexity values (see e.g. instances 6×6, 8×6, 10×7).

Considering the number of teams and employees used in the solutions (columns T

Table 4.2: Performance metrics for computation times

Instance	**DM**			**RM1**			**RM2**		
$\lvert J\rvert \times \lvert M\rvert$	*CPU*	*GAP*	*Opt.*	*CPU*	*GAP*	*Opt.*	*CPU*	*GAP*	*Opt.*
4 x 4	0.02	0	10	0.02	0	10	0.16	0	10
4 x 8	0.02	0	10	0.04	0	10	0.46	0	10
6 x 6	0.15	0	10	0.70	0	10	7.06	0	10
6 x 12	0.15	0	10	1.27	0	10	143.37	0	10
8 x 6	0.64	0	10	0.79	0	10	425.68	5	9
8 x 12	128.82	0	10	346.74	0	10	1669.68	22	6
10 x 7	142.72	0	10	237.35	0	10	1468.76	17	7
10 x 13	1621.64	2	7	2720.58	7	4	3522.07	55	1
15 x 8	2865.84	33	3	2883.81	29	3	3600.00	81	0
15 x 15	3600.00	29	0	3600.00	24	0	3600.00	73	0
20 x 10	3600.00	54	0	3600.00	42	0	3600.00	76	0
20 x 20	3600.00	32	0	3600.00	23	0	3600.00	74	0

and E), we observe that **RM1** assigns employees to a consistently lower number of teams than **DM**. This indicates that larger teams are created in order to guarantee a greater schedule reliability in the presence of possible data variations. Moreover, we observe a further decrease of T when comparing **RM1** and **RM2** for the majority of instances. Similar to **RM1**, all employees are involved. An exception to this are large-sized instances that could not be solved to optimality, see instances 10×13 - 15×15 and 20×20. This also demonstrates a similar tendency to increase the level of protection by increasing the team size.

A further examination shows that, compared to **DM**, **RM1** and **RM2** result in a lower total job completion time due to a lower number of performed jobs.

Table 4.2 provides statistics for the consumed runtime expressed in seconds (column *CPU*) and the optimality gap in percent (columns *GAP*) reported by CPLEX after the runtime limit of 3600 seconds per instance. The optimality gap is computed as $GAP =$ (Objective - LB)/LB where „Objective“ denotes the value of the objective function achieved by the model and „LB“ gives the lower bound value reported by CPLEX. Column *Opt.* gives the number of instances solved to optimality in each instance set.

The obtained results show that the computational time increases with an increase of the instance size. Looking at column *Opt.*, we see that already small instances containing less that 10 jobs could not be solved to optimality within the preset runtime limit. According to Table 4.2, there is no substantial difference in the complexity of **DM** and **RM1**. **RM1** delivers almost the same number of optimal solutions as **DM** does. Thereby, **RM1** demonstrates only slightly lower $GAPs$ values compared to **DM**, see instances 15×8 - 20×20. In contrast, **RM2** requires a considerably higher computational effort due to a much larger number of variables and constraints. Already for instances of size 8×6, we observe a positive average GAP and considerably larger CPU times.

4.5.3 Benefit of Robustness

The next two experiments assess the effect of data changes on the solution feasibility. In other words, we test how many planned jobs the teams can actually perform when uncertain skill requirements realize in the schedule execution. For this purpose, we generate for each optimization approach $1,000$ demand scenarios per instance set. Thus, the results are averages over $S = 10 \cdot 1,000 = 10,000$ scenarios. We start by generating scenarios of type **RM1** which are modeled with $\Gamma_j = 3$, i.e. 3 elements are varied in the original qualification requirement matrix of each job. The obtained results are reported in Table 4.3. The first column shows the problem size. Columns A in each block give the average of the absolute number of performed jobs while columns R indicate the average relative proportion of processed jobs in all scenarios in percent. For a scenario s, we compute R_s as $R_s = A_s/Z$ where Z refers to the number of originally performed jobs for the corresponding model. Columns B show the relative frequency in percent with which each robust model outperforms **DM**. Finally, we report in columns W the relative frequency with which the service level attained by each robust model is lower than the nominal one.

The results for **DM** show that a considerable number of job assignments becomes infeasible. In fact, the relative service level R drops below 20% for the most instances. This means that although the deterministic model inserts a lot of jobs in a solution, it finally fails to process these jobs due to uncertain job requirements and insufficiently

Table 4.3: Scenarios of type RM1 (Best average service level per row is highlighted in bold)

Instance	**DM**		**RM1**				**RM2**			
$\lvert J\rvert \times \lvert M\rvert$	A	R	A	R	B	W	A	R	B	W
4 x 4	0.50	21.33	**1.11**	63.96	50.63	2.64	1.10	71.36	50.69	2.23
4 x 8	1.02	33.35	1.46	50.12	44.84	12.90	**1.73**	86.75	57.12	4.29
6 x 6	0.65	18.23	1.76	67.08	73.50	5.74	**1.88**	73.12	82.37	0.50
6 x 12	1.66	34.93	2.59	55.13	62.60	12.75	**3.07**	80.78	81.17	1.67
8 x 6	0.78	18.30	1.96	60.80	75.05	4.82	**2.12**	70.78	85.06	1.82
8 x 12	1.37	22.77	3.02	54.11	78.49	5.44	**3.13**	61.72	76.56	5.79
10 x 7	0.96	20.12	**2.17**	56.90	73.29	2.89	2.00	51.69	68.05	4.28
10 x 13	1.39	18.23	2.98	45.96	77.00	9.15	**3.32**	58.04	88.67	2.12
15 x 8	1.21	18.55	**2.63**	53.74	73.50	5.84	2.51	66.51	73.10	8.87
15 x 15	1.64	16.19	**4.23**	51.55	91.00	1.65	3.75	60.75	88.58	3.58
20 x 10	1.55	17.47	**3.30**	49.08	81.87	4.15	2.80	48.29	69.10	15.03
20 x 20	1.70	11.78	5.03	46.06	96.20	0.92	**5.17**	72.34	90.94	3.04

qualified teams. This low reliability can be substantially moderated by **RM1**, which is confirmed by significantly higher R values ranging between 45% and 67%. However, note that these values already lie below 100% for $\Gamma_j = 3$. This is because $\Gamma_j = 4$ guarantees the solution feasibility only for the aggregated skill level but not for every single element of matrix r_{jkl}. This means that a solution can become infeasible also for skill deviation that is below the defined uncertainty budget $\Gamma_j = 4$. Moreover, for all instances, we observe significantly higher absolute numbers of still feasible job assignments (see column A). Looking at column B, we see that **RM1** outperforms **DM** in 45% to 96% of all scenarios while W ranges between 1% and 13% only. Although **RM1** achieved a lower service level than **DM** in the first experiment (see Table 5.1), it now performs clearly stronger under the uncertain problem setting. Also **RM2** is superior to **DM**. In fact, **RM2** delivers better results in 50% to 90% of all scenarios, see column B. Moreover, compared to **RM1**, we even observe a higher absolute and relative service level achieved under **RM2** for instances 4×8 to 8×12, 10×13 and 20×20. However, solutions for larger instances are less immunized against this type of uncertainty as they are solved under lower Γ compared to small- and medium-sized instances. This is due to a higher problem complexity and, thus, computational effort required by **RM2** that does not allow to further increase uncertainty budget within the preset time limit.

Table 4.4: Scenarios of type RM2 (Best average service level per row is highlighted in bold)

Instance	**DM**		**RM1**				**RM2**			
$\|J\| \times \|M\|$	A	R	A	R	B	W	A	R	B	W
4 x 4	0.60	26.60	1.04	64.84	46.03	7.31	**1.14**	77.39	51.56	4.98
4 x 8	1.05	33.75	1.47	50.97	43.40	14.44	**1.69**	84.75	54.42	6.71
6 x 6	0.97	26.86	1.85	69.62	64.66	9.90	**1.98**	75.19	74.24	2.76
6 x 12	1.57	33.96	2.29	48.88	53.63	15.30	**3.03**	79.83	79.32	2.40
8 x 6	1.04	26.61	1.86	57.15	59.77	11.32	**2.21**	73.79	75.44	3.01
8 x 12	1.50	24.47	2.94	52.89	75.57	5.02	**3.23**	63.91	81.24	6.31
10 x 7	1.28	25.68	2.24	58.91	65.55	9.87	**2.27**	57.85	71.19	8.05
10 x 13	1.93	25.52	3.30	50.13	66.67	13.58	**3.46**	60.54	76.22	7.21
15 x 8	1.55	23.55	**2.79**	57.27	67.83	10.29	2.76	71.43	71.17	9.18
15 x 15	2.45	23.92	**4.64**	56.48	87.19	3.41	4.09	66.31	78.06	7.31
20 x 10	2.04	22.77	**3.93**	58.26	83.78	4.38	3.16	53.38	67.58	13.94
20 x 20	2.58	18.40	**5.71**	52.03	93.57	1.95	5.30	72.91	86.73	5.89

To achieve a fair comparison between the two robust approaches, we conduct a second experiment to evaluate the quality of solutions under scenarios of type **RM2**. For this purpose, we create further $10,000$ scenarios ($1,000$ for each instance set) with uncertainty budget $\Gamma = 10 \cdot |J|$. Following definition in Section 4.2, the scenarios are modeled such that the uncertainty budget is bounded over all jobs. To simulate different skill realizations, the sequence in which the jobs are considered is defined randomly for each scenario. For each considered job, one element is changed in matrix r_{jkl}. Thereby, skill domain and the competence level are selected randomly with equal probability of $1/|K|$ and $1/|L|$. The process is continued until the budget is reached.

The obtained results are summarized in Table 4.4. Here, we see that the general trends are similar to those in the previous experiment. Compared to both robust approaches, we observe a low solution feasibility for **DM** with R values that lie under 30% for the majority of instances. With **RM1** this ratio is increased to further 48% to 69%. Moreover, we observe an increase in R by further 6% to 34% when comparing **RM1** and **RM2**. In absolute terms, **RM2** outperforms **RM1** in 67% of cases. This holds especially for instances that could be solved to optimality. Whereas, for the four last instances, **RM1** is again superior to **RM2**. Analyzing the performance of each single approach under different uncertainty settings, we observe higher B and lower W values in the previous experiment for both **RM1** and **RM2**. This is explained by a higher uncertainty imposed by scenarios of type **RM1** and, thus, a lower number of feasible job assignments in **DM**-solution (see column A in Table 4.3 and Table 4.4).

4.5.4 Variation of Uncertainty Budget

To give a more detailed understanding of the differences between the two robust planning approaches, we conduct a sensitivity analysis by varying the uncertainty budget for the generation of scenarios. Note that we do not recompute the model solutions in the simulation but evaluate feasibility of each scenario based on job assignments reported in Table 4.1. For scenarios of type **RM1**, Γ_j is varied on the interval [0, 6]. Here, value 0 means that no skill deviations are considered, whereas value 6 means that six elements of the job requirement matrix can deviate from their nominal values simultaneously. For scenarios of type **RM2**, Γ is varied on the interval $[0, 16 \cdot |J|]$. Figures 4.1 and 4.2 show the impact of different parameter settings on the service level for three selected instance sets. Each plot relates to the average solutions of $10,000$ scenarios generated under different Γ-settings. Scenarios were generated according to the process described in the previous subsection.

The results demonstrate that the service level is inversely correlated with the level of uncertainty. We can see a decline in the number of performed jobs with higher Γ_j and Γ as the data becomes more and more uncertain. Thereby, marginal cost of robustness increase with an increase of the instance size. In general, solution quality in **RM1**-scenarios worsens at a strong rate. The service level drops to substantially lower values already in the middle of the examined interval. This is an expected outcome. As the uncertainty budget defined for **RM1** is aggregated job-wise, Γ_j-variations are more challenging and incur a higher price of robustness. Further, we observe that **RM1** achieves the same or even better service level at the interval $\Gamma_j \in [0, 3]$, whereas **RM2** generates a more robust solution for $\Gamma_j \in [4, 6]$, see Figure 4.1. To this end, the both proposed approaches significantly outperform **DM**.

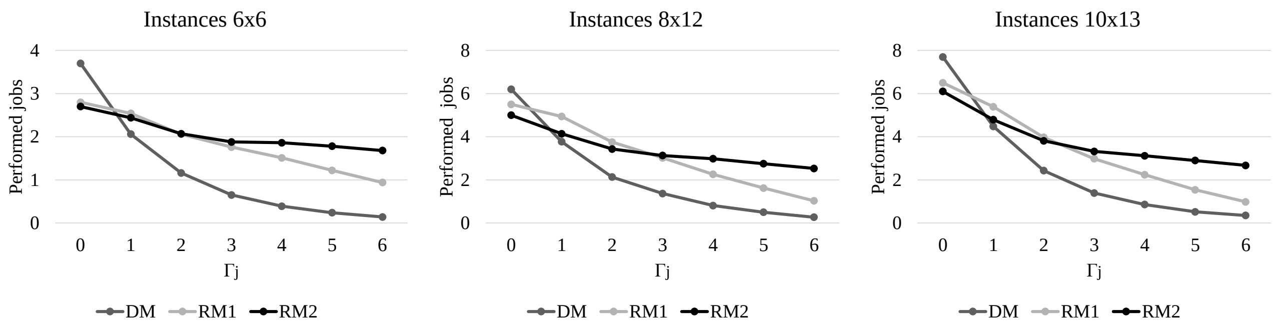


Figure 4.1: Influence of uncertainty on the number of performed jobs with scenarios of type RM1

A similar pattern emerges in Figure 4.2. Here, we also see that **RM1** is superior to **RM2** if the expected skill variation is relatively low. However, the break even point is now reached after only $\Gamma = 8$. This is explained by the fact that **RM2** is more conservative with our choice of Γ. This confirm the results reported in Table 4.1 where the service level achieved under **RM1** is consistently higher compared to **RM2**. From this, a higher level of data uncertainty is required by **RM2** to produce a significant impact on performance indicator.

To summarize our results, we can conclude that all proposed robust approaches can successfully handle the demand uncertainty. **RM2** provides a higher solution feasibility than **RM1** for our choice of Γ, but tends to give the best performance on a wide range of levels of uncertainty, and even on scenarios that are generated in the style of **RM1**. The advantage of **RM1**, on the other hand, is its reduced computational complexity,

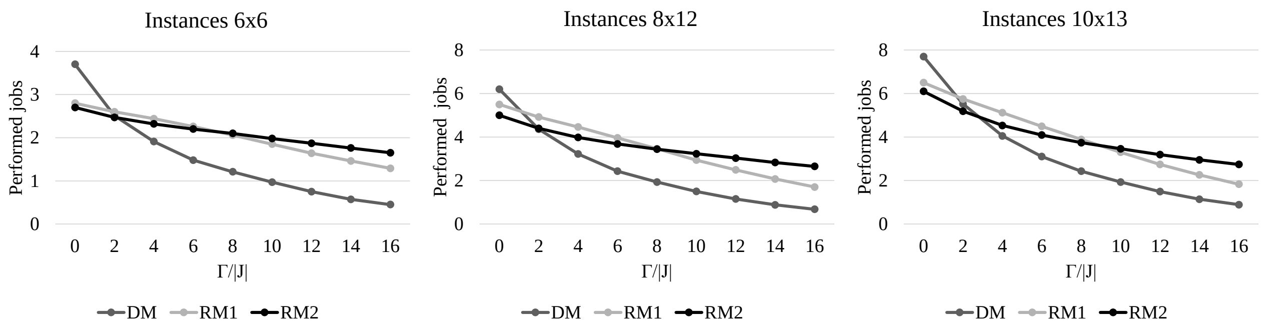


Figure 4.2: Influence of uncertainty on the number of performed jobs with scenarios of type RM2

which makes it possible to find robust solutions even on the largest instances.

4.6 Conclusions and Future Research

In this paper, we have investigated the problem of routing and scheduling multi-skilled teams under demand uncertainty where the variations of job qualification requirements are captured through uncertainty sets. For the solution of the problem, we have developed and analyzed two robust modeling approaches. Computational experiments showed that deviations in qualification requirements can have an extremely negative impact on the quality and the feasibility of the obtained solutions in a non-robust planning. This can be significantly moderated by the proposed robust approaches. The degree of solution robustness can be controlled not only by choosing uncertainty budget Γ but also by choosing an appropriate method to model the uncertainty set. Specifically, we demonstrated that a higher protection against any demand variations is provided if the demand uncertainty can be distributed over the complete customer network where uncertainty cost are defined for each particular skill. Alternatively, uncertainty might be aggregated for each job separately. This allows to reach a reasonable compromise between the risk aversion and the achieved service level.

As this study represents a first step to incorporate uncertainty into scheduling of multi-skilled teams, there are still many promising avenues for future research. For instance, this study can be extended to other variants of uncertainty sets. In practice, the changes in job qualification requirements are often coupled with changes in job processing times. From this, it could be interesting to model interdependencies between these two parameters in the context of robust optimization. Finally, as the optimization models cannot be solved to optimality for large-scale problems, it would be worthwhile to develop further heuristic approaches.

Bibliography

Adjiashvili, D., Stiller, S., and Zenklusen, R. (2015). Bulk-robust combinatorial optimization. *Mathematical Programming*, 149:361–390.

Anoshkina, Y. and Meisel, F. (2019). Technician teaming and routing with service-, cost-and fairness-objectives. *Computers & Industrial Engineering*, 135:868–880.

Anoshkina, Y. and Meisel, F. (2020). Interday routing and scheduling of multi-skilled teams with consistency consideration and intraday rescheduling. *EURO Journal on Transportation and Logistics*, 9:1–18.

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

Ben-Tal, A. and Nemirovski, A. (1999). Robust solutions of uncertain linear programs. *Operations Research Letters*, 25:1–13.

Bertsimas, D. and Sim, M. (2003). Robust discrete optimization and network flows. *Mathematical Programming*, 98:49–71.

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

Bohle, C., Maturana, S., and Vera, J. (2010). A robust optimization approach to wine grape harvesting scheduling. *European Journal of Operational Research*, 200:245–252.

Buchheim, C. and Kurtz, J. (2017). Min–max–min robust combinatorial optimization. *Mathematical Programming*, 163:1–23.

Buchheim, C. and Kurtz, J. (2018). Robust combinatorial optimization under convex and discrete cost uncertainty. *EURO Journal on Computational Optimization*, 6:211–238.

Cao, E., Lai, M., and Yang, H. (2014). Open vehicle routing problem with demand uncertainty and its robust strategies. *Expert Systems with Applications*, 41:3569–3575.

Carello, G. and Lanzarone, E. (2014). A cardinality-constrained robust model for the assignment problem in home care services. *European Journal of Operational Research*, 236:748–762.

Chen, L., Gendreau, M., Hà, M. H., and Langevin, A. (2016). A robust optimization approach for the road network daily maintenance routing problem with uncertain service time. *Transportation Research Part E: Logistics and Transportation Review*, 85:40–51.

Cordeau, J.-F., Laporte, G., Pasin, F., and Ropke, S. (2010). Scheduling technicians and tasks in a telecommunication company. *Journal of Scheduling*, 13:393–409.

De La Vega, J., Munari, P., and Morabito, R. (2019). Robust optimization for the vehicle routing problem with multiple deliverymen. *Central European Journal of Operations Research*, 27:905–936.

Estellon, B., Gardi, F., and Nouioua, K. (2009). High-performance local search for task scheduling with human resource allocation. In *International Workshop on Engineering Stochastic Local Search Algorithms*, pages 1–15. Springer.

Fırat, M., Briskorn, D., and Laugier, A. (2016). A branch-and-price algorithm for stable workforce assignments with hierarchical skills. *European Journal of Operational Research*, 251:676–685.

Fırat, M. and Hurkens, C. A. (2012). An improved MIP-based approach for a multi-skill workforce scheduling problem. *Journal of Scheduling*, 15:363–380.

Gabrel, V., Murat, C., and Thiele, A. (2014). Recent advances in robust optimization: An overview. *European Journal of Operational Research*, 235:471–483.

Goerigk, M. and Schöbel, A. (2016). Algorithm engineering in robust optimization. In *Algorithm engineering*, pages 245–279. Springer.

Han, J., Lee, C., and Park, S. (2014). A robust scenario approach for the vehicle routing problem with uncertain travel times. *Transportation Science*, 48:373–390.

Hanasusanto, G. A., Kuhn, D., and Wiesemann, W. (2015). K-adaptability in two-stage robust binary programming. *Operations Research*, 63:877–891.

Hashimoto, H., Boussier, S., Vasquez, M., and Wilbaut, C. (2011). A GRASP-based approach for technicians and interventions scheduling for telecommunications. *Annals of Operations Research*, 183:143–161.

Henao, C. A., Ferrer, J. C., Muñoz, J. C., and Vera, J. (2016). Multiskilling with closed chains in a service industry: A robust optimization approach. *International Journal of Production Economics*, 179:166–178.

Hurkens, C. A. (2009). Incorporating the strength of MIP modeling in schedule construction. *RAIRO-Operations Research*, 43:409–420.

Khalfay, A., Crispin, A., and Crockett, K. (2017). Applying the intelligent decision heuristic to solve large scale technician and task scheduling problem. *International Conference on Intelligent Decision Technologies*, 72:71–81.

Kovacs, A. A., Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2012). Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of Scheduling*, 15:579–600.

Lee, C., Lee, K., and Park, S. (2012). Robust vehicle routing problem with deadlines and travel time/demand uncertainty. *Journal of the Operational Research Society*, 63:1294–1306.

Liebchen, C., Lübbecke, M., Möhring, R., and Stiller, S. (2009). The concept of recoverable robustness, linear programming recovery, and railway applications. *Robust and Online Large-Scale Optimization. Lecture Notes in Computer Science*, 5868:1–27.

Munari, P., Moreno, A., De La Vega, J., Alem, D., Gondzio, J., and Morabito, R. (2019). The robust vehicle routing problem with time windows: Compact formulation and branch-price-and-cut method. *Transportation Science*, 53:1043–1066.

Nguyen, T. V. L. and Montemanni, R. (2016). Robust home health care services: An enhanced matheuristic optimization. *The Dalle Molle Institute for Artificial Intelligence Research, Tech. Rep. TR-01-16.*

Ordóñez, F. (2010). Robust vehicle routing. In *Risk and optimization in an Uncertain World*, pages 153–178. INFORMS.

Poss, M. (2013). Robust combinatorial optimization with variable budgeted uncertainty. *4OR*, 11:75–92.

Souyris, S., Cortés, C. E., Ordóñez, F., and Weintraub, A. (2013). A robust optimization approach to dispatching technicians under stochastic service times. *Optimization Letters*, 7:1549–1568.

Sungur, I., Ordónez, F., and Dessouky, M. (2008). A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty. *IIE Transactions*, 40:509–523.

Yanıkoğlu, İ., Gorissen, B. L., and den Hertog, D. (2019). A survey of adjustable robust optimization. *European Journal of Operational Research*, 277:799–813.

Zamorano, E. and Stolletz, R. (2017). Branch-and-price approaches for the multiperiod technician routing and scheduling problem. *European Journal of Operational Research*, 257:55–68.

5 Essay 4

Routing and Scheduling of Multi-Skilled Teams with Simultaneous and Sequential Use of Skill

Yulia Anoshkina[a],

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

Abstract: In this paper, we consider a problem of teaming and scheduling of multi-skilled employees that have to perform a set of geographically distributed jobs. In particular, we focus on a competence-based estimation of job processing times. Based on a hierarchical competence concept, this work proposes a linear optimization model that incorporates a so-called sequential use of skill where employees can contribute to multiple skill domains sequentially and the job processing time is considered as a function of team size and competence level. The effect of sequential use of skill on large sized instances is evaluated by means of an adaptive large neighborhood search heuristic (ALNS). Extensive experiments are conducted to analyze the effectiveness of the proposed approach and to reveal the impact on the achieved service level.

Keywords: Multi-Skilled Workforce Scheduling, Team Scheduling, Sequential Use of Skill, Hierarchical Skill, Competence-Based Model, Adaptive Large Neighborhood Search

5.1 Introduction

A problem where a set of employees proficient in different skill domains at different levels of competence have to be grouped and routed to perform a set of jobs is referred to as a service technician routing and scheduling problem (STRSP), see Kovacs et al. (2012).

Such a problem is faced by telecommunication, construction or consultancy companies, among others. Since the first investigation of such a problem in 2007, a number of competing solutions together with numerous extensions have been proposed, see e.g. Estellon et al. (2009), Hurkens (2009), Cordeau et al. (2010), Kovacs et al. (2012), Fırat et al. (2016), Zamorano and Stolletz (2017), Khalfay et al. (2017), Anoshkina and Meisel (2019). Detailed literature surveys with a discussion of applications and solution methods can be found in Paraskevopoulos et al. (2016), De Bruecker et al. (2015) or Anoshkina and Meisel (2020). Although a number of different aspects has been investigated, the role of skill distribution within a team has received less attention. Specifically, it was assumed that employees can contribute to the job progress in all domains in that they are qualified simultaneously. According to the classification of workforce planning problems by De Bruecker et al. (2015), the problem investigated in our paper involves so-called hierarchical skills. This class of skills summarizes the following features:

1. experience: skills are discretized into several qualification levels,
2. substitution: higher skilled employees can perform tasks requiring a lower level of qualification,
3. workload: employees with a higher skill level can perform more tasks,
4. working speed: employees with a higher skill level can perform certain tasks better or faster.

When modeling skill requirements, previous optimization studies focused on the first three determinants of hierarchical skills. Regarding working speed, previous papers assume that job processing times remain constant and independent on other factors, see e.g. Cordeau et al. (2010), Kovacs et al. (2012), Firat and Hurkens (2012), Hashimoto et al. (2011) and Anoshkina and Meisel (2019), Anoshkina and Meisel (2020). However, working speed has a direct impact on job processing times and, thus, represents an important scheduling aspect. Thereby, in the context of team scheduling, working speed entails more the effect of team composition rather than the performance of a single employee. The empirical study of Cassera et al. (2009) demonstrates that team size is the second most important factor after job complexity accounting for variability in processing time. However, in quantitative studies, competence-based performance is usually considered in other contexts. For instance, studies devoted to project management (Walter and Zimermann (2016), Karam et al. (2017)) underline that each worker can execute only one operation (skill) of a project task at any time period. Thereby, the time required to perform an operation in a particular skill domain is negatively related to the efficiency in practicing this skill by the assigned employees. To integrate employee's efficiency into project scheduling, a number of non-linear optimization models have been presented in recent years. Thus, the primary interest lies in exploring the evolution of an employee's skill levels over time. The most common representation of this evolution is the non-linear learning curve, see Heimerl and Kolish (2010), Gutjahr et al. (2010), Attia et al. (2014), Qin et al. (2016), Zha and Zhang (2014). A more recent contribution (Chen et al. (2020)) accounts for a relation between an employee's efficiency and product quality.

Apart from project scheduling, Malachowski and Korytkowski (2016) propose a static competence model for deriving and updating qualifications of multi-skilled employees undertaking repetitive tasks. Thereby, individual learning rates and experiences serve as a basis for defining minimal and maximal durations of operations in each working station. Chen et al. (2016) incorporate an experienced-based learning function in the daily routing of technicians. For the solution of the problem, the authors propose a modeling framework based on a Markov decision process together with a variant of a so-called record-to-record travel algorithm. In a further study, Chen et al. (2017) extend the problem by stochastic customer demand which becomes known only on the day of service.

Although helpful for our analysis, the cited works demonstrate some significant differences from the problem investigated in this paper. Project scheduling does not involve teaming decisions. Furthermore, the key aspect of our study is not the evolution of employee efficiency over the planning horizon but an adequate estimation of job processing times at the considered planning period. Our literature survey did not reveal any prior studies that consider the problem from this perspective. To bridge this gap, we propose a concept of *sequential use of skill* that assumes that each employee can perform only one job operation at a time. If an employee has to contribute to the job progress in more than one skill domain, all required operations are performed in a sequential manner, i.e. one-by-one. The job processing time varies with the number and competence level of assigned employees. Our contribution is then threefold.

- First, we develop a competence-based performance model for multi-skilled teams that can derive the job-skill-contribution of each single employee and use it to estimate job processing times.
- Second, we introduce a linear optimization model that involves a skill-based estimation of job processing times as a function of team size and competence level.
- Third, we propose an ALNS algorithm that includes three different team construction schemes designed specifically for the problem considered. To this end, we adopt destroy and repair operators to the sequential skill setting and propose an additional restart mechanism to diversify and accelerate the search.

The remainder of this paper is structured as follows. Section 5.2 presents the background of the problem together with the linear formulation of the sequential STRSP.

Section 5.3 includes the description of the heuristic solution approach. Section 5.4 describes the test problems and the computational experiments. Finally, Section 5.5 concludes with findings of this work and areas for future research.

5.2 Mathematical Optimization Model

5.2.1 Basic Notation

The multi-skilled scheduling and routing problem can be defined on a connected graph $G = (J^0, E)$, where $J^0 = \{0\} \cup J$ is a vertex set and $E = \{(i,j)|i,j \in J^0\}$ is the corresponding set of edges. An arc (i,j) represents the possibility to travel between two vertices i and j. Each trip between vertices (i,j) implies a non-negative travel time d_{ij}. Vertex 0 represents a depot and vertices $J = \{1, ...|J|\}$ denote the locations of jobs $1...|J|$. Each job is associated with service that has to be started within the prescribed time window $[a_j, b_j]$, where a_j and b_j denote the earliest and the latest start times correspondingly. The services have to be performed by a set of employees trained in different skill domains at specific levels of competence. The job skill requirement $R_j(k,l)$ represents a two-dimensional matrix, where columns $k \in K$ refer to skill domains and rows $l \in L$ refer to experience levels. A matrix element r_{jkl} gives the required number of employees qualified in domain k at experience level l for performing job j. The matrix in the middle of Figure 5.1 shows an example of such a skill requirement. In this example, two employees have to be trained in skill domain $k = 1$, one at least at level $l = 3$ and another one at least at level $l = 2$. The underlying skill vectors of these employees are $(1,1,1)$ and $(1,1,0)$. Similar, one employee proficient at level $l = 2$ is needed in domain $k = 2$, and one further employee must be trained at least at level $l = 1$ of skill $k = 3$.

In order to satisfy all service requirements, employees are grouped into teams. Each job $j \in J$ can be performed by at most one team that has an appropriate qualification level for performing this job. Note that the team can also be overqualified. We analyze the overqualification from two perspectives: in terms of aggregated competence level

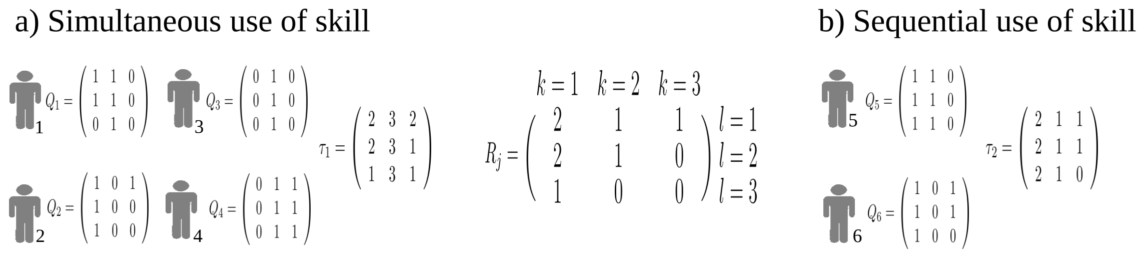

a) Simultaneous use of skill
b) Sequential use of skill
$Q_1 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$
1
$Q_3 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$
3
$Q_2 = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$
2
$Q_4 = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$
4
$\tau_1 = \begin{pmatrix} 2 & 3 & 2 \\ 2 & 3 & 1 \\ 1 & 3 & 1 \end{pmatrix}$
$R_j = \begin{array}{c|ccc|c} & k=1 & k=2 & k=3 & \\ & 2 & 1 & 1 & l=1 \\ & 2 & 1 & 0 & l=2 \\ & 1 & 0 & 0 & l=3 \end{array}$
$Q_5 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$
5
$Q_6 = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$
6
$\tau_2 = \begin{pmatrix} 2 & 1 & 1 \\ 2 & 1 & 1 \\ 2 & 1 & 0 \end{pmatrix}$

Figure 5.1: Use of skill

in each skill domain possessed by the team members as well as in terms of number of employees in the team. For a better understanding, we illustrate in Figure 5.1 two different team compositions conceivable for job j. The figure shows individual skill-qualification vectors of each employee $m \in M$ in a binary matrix Q_m that is built according to the first three principles of the hierarchical skill concept as follows.

1. *Experience*: the elements q_{mkl} of matrix Q_m show employee m's proficiency in skill domain $k \in K$ at level $l \in L$.

2. *Substitution*: qualification vectors are arranged in such a way that $q_{mkl'} \leqslant q_{mkl}$, where $l' > l$. From this, an employee that is trained in domain k at a particular level l, is also qualified for all lower competence levels in the respective domain. For instance, employee 1 is proficient in skill $k = 1$ exactly at the level $l = 2$ required by job j and is even overqualified in domain $k = 2$, see Figure 5.1a.

3. *Workload*: employee m can perform operations in more than one skill domain if $\sum_{k \in K} q_{mk1} > 1$.

The aggregated competence levels of team t in each domain are defined as the sum of all individual skills possessed by team members and are summarized in team skill matrices τ_t, where values $\tau_t > r_{jkl}$ indicate team skill overqualification.

Eventually, teams must be sufficiently large to have the required expertise. As a job's skill requirement matrix is arranged monotonically decreasing in the skill levels, the required team size Max_j represents the sum of skill requirements at the lowest level

$l = 1$:

$$Max_j = \sum_{k \in K} r_{jk1} \tag{5.1}$$

Thus, we have $Max_j = 2 + 1 + 1 = 4$ for R_j in Figure 5.1. If the team size corresponds exactly to Max_j, work can be allocated across the team members such that each employee contributes to the job progress only in one particular skill domain, see Figure 5.1a. Hence, all service operations are performed simultaneously. We refer to this as *simultaneous use of skill*.

Since each employee can possess multiple skills, job j might be performed by a team with a lower team size, see Figure 5.1b. The minimal team size Min_j corresponds to the maximum qualification requirement at level $l = 1$:

$$Min_j = \max_{k \in K}\{r_{jk1}\} \tag{5.2}$$

Here, we have $Min_j = \max\{2, 1, 1\} = 2$ for R_j in Figure 5.1. In this scenario, the assigned employees have to perform operations in more than one skill domain. As we assume that single operations involved in a job have to be performed one-by-one, we refer to this as *sequential use of skill*.

5.2.2 Modeling Simultaneous Use of Skill

Optimization models for the STRSP with simultaneous use of skill have been proposed by Cordeau et al. (2010), Kovacs et al. (2012), Zamorano and Stolletz (2017) and Anoshkina and Meisel (2019, 2020). The central idea of simultaneous skill modeling is that the job processing time p_j is a given parameter that remains constant and independent of the team size no matter whether the team size is different from Max_j. To later derive the sequential STRSP, we first describe here an adapted version of the model with simultaneous use of skill as provided by Anoshkina and Meisel (2020). The following decision variables are introduced. The binary decision variable x_{mt} indicates if an employee m is assigned to team t. The routing of teams is denoted by a further 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. Using the introduced notation, the optimization model is formulated as follows.

$$\text{maximize: } \alpha \cdot \sum_{t\in T}\sum_{i\in J^0}\sum_{j\in J} z_{tij} - \beta \cdot \sum_{m\in M}\sum_{t\in T} x_{mt} - \gamma \cdot \sum_{t\in T}\sum_{j\in J} f_{tj} \tag{5.3}$$

subject to:

$$\sum_{t\in T} x_{mt} \leqslant 1 \qquad \forall m \in M \tag{5.4}$$

$$\sum_{m\in M} x_{mt} \cdot q_{mkl} \geqslant r_{jkl} \cdot \sum_{i\in J^0} z_{tij} \qquad \forall j \in J, k \in K, l \in L, t \in T \tag{5.5}$$

$$\sum_{j\in J} z_{t0j} \leqslant 1 \qquad \forall\ t \in T \tag{5.6}$$

$$\sum_{t\in T}\sum_{i\in J^0} z_{tij} \leqslant 1 \qquad \forall\ j \in J \tag{5.7}$$

$$\sum_{i\in J^0} z_{tij} = \sum_{i\in J^0} z_{tji} \qquad \forall\ j \in J^0, t \in T \tag{5.8}$$

$$f_{ti} + d_{ij} \leqslant s_{tj} + \mathcal{M} \cdot (1 - z_{tij}) \qquad \forall i \in J^0, j \in J, t \in T \tag{5.9}$$

$$s_{tj} + p_j \leqslant f_{tj} + \mathcal{M} \cdot \left(1 - \sum_{i\in J^0} z_{tij}\right) \qquad \forall j \in J, t \in T \tag{5.10}$$

$$s_{tj} \geqslant a_j - \mathcal{M} \cdot \left(1 - \sum_{i\in J^0} z_{tij}\right) \qquad \forall j \in J, t \in T \tag{5.11}$$

$$s_{tj} \leqslant b_j + \mathcal{M} \cdot \left(1 - \sum_{i\in J^0} z_{tij}\right) \qquad \forall j \in J, t \in T \tag{5.12}$$

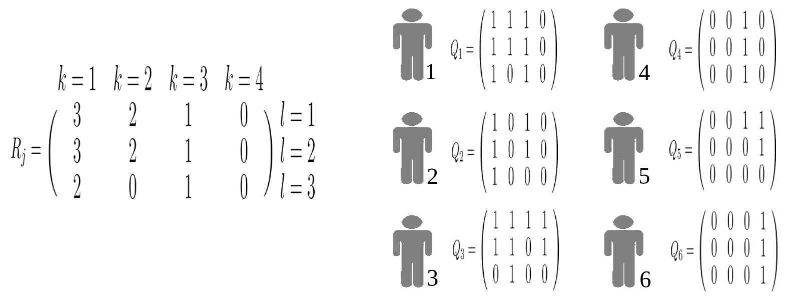
$$s_{tj}, f_{tj} \geqslant 0 \qquad \forall j \in J^0, t \in T \tag{5.13}$$

$$x_{mt}, z_{tij} \in \{0, 1\} \qquad \forall i, j \in J^0, m \in M, t \in T \tag{5.14}$$

The main goal of the model is to maximize the service quality. For this purpose, the first component of the objective function (5.3) maximizes the number of jobs that are assigned to teams and, thus, processed in the solution. The second component minimizes the number of assigned employees to avoid inefficient team composition. The third component refers to the minor objective of minimizing the total job completion time. Weights α, β and γ express different priorities of the three objectives. We assume $\alpha \gg \beta \geqslant \gamma$. Constraints (5.4) state that each employee can be a part of at most one team. Constraints (5.5) ensure that the team composition is appropriate to cover all skills required by each job assigned to this team. Constraints (5.6) demand that each team starts its tour at the depot at most once. Constraints (5.7) guarantee that each job can be performed at most once by at most one team. Constraints (5.8) balance the flow of team at each job j. Constraints (5.9) specify the time at which team t starts performing job j. Constraints (5.10) determine the completion time of job j if performed by team t. Constraints (5.11)-(5.12) state that the jobs are started within the time windows. Constraints (5.13)-(5.14) specify the domains of decision variables.

5.2.3 Modeling Sequential Use of Skill

In this subsection, we formalize the sequential skill setting and provide the corresponding optimization model. First, we discuss the computation of the *effective team size.* For each job-team pair (j, t), we interpret the effective team size as the number of those employees in team t that can actually contribute to the work progress of job j. Here, we adopt the *working speed* principle of the hierarchical skill concept as follows. Considering an individual job contribution, we assume that employees that are qualified in a skill domain at a lower qualification level than is required by the job can contribute to the work progress partially but need more time to complete the task. Hence, the job processing time varies with the level of competence of assigned employees. As a numerical example consider the scenario shown in Figure 5.2 with a job requiring qualifications in 3 out of 4 skill domains and a team of 6 employees. Following equation (5.1), we find that $Max_j = 6$. Though Max_j is equal to the number of employees in the given team, not all employees can progress at a similar rate. For example, employee 5 can perform


k=1 k=2 k=3 k=4
Rj = (3 2 1 0; 3 2 1 0; 2 0 1 0) l=1 l=2 l=3
1 Q1 = (1 1 1 0; 1 1 1 0; 1 0 1 0)
2 Q2 = (1 0 1 0; 1 0 1 0; 1 0 0 0)
3 Q3 = (1 1 1 1; 1 1 0 1; 0 1 0 0)
4 Q4 = (0 0 1 0; 0 0 1 0; 0 0 1 0)
5 Q5 = (0 0 1 1; 0 0 0 1; 0 0 0 0)
6 Q6 = (0 0 0 1; 0 0 0 1; 0 0 0 1)
Figure 5.2: Effective team size

operations in domains $k = 3$ and $k = 4$, but, since the respective job does not require any operations in domain $k = 4$, employee 5 can contribute only in domain $k = 3$. Furthermore, the competence level of employee 5 for skill $k = 3$ is significantly below the required one. Moreover, employee 6 does not possess any skills required by the job. To identify such patterns, we specify an individual *job-skill-coefficient* ε_{mjk} that indicates to what extent an employee m can be used to perform a particular skill k required by job j. As Constraints (5.5) guarantee that the job skill requirements are covered at all experience levels including the highest ones independing on the team composition, it suffices to examine the minimal job skill requirement r_{jkl}^{min} in order to define ε_{mjk}. To derive r_{jkl}^{min}, we iteratively split each qualification requirement in matrix R_j into single skill vectors and select the vector with the least highest experience level. For matrix R_j in Figure 5.2, we obtain:

$$R_j = \begin{array}{c} \begin{array}{cccc} k=1 & k=2 & k=3 & k=4 \end{array} \\ \begin{pmatrix} 1+1+\mathbf{1} & 1+\mathbf{1} & \mathbf{1} & \mathbf{0} \\ 1+1+\mathbf{1} & 1+\mathbf{1} & \mathbf{1} & \mathbf{0} \\ 1+1+\mathbf{0} & 0+\mathbf{0} & \mathbf{1} & \mathbf{0} \end{pmatrix} \end{array} \qquad r_{jkl}^{min} = \begin{array}{cc} \begin{array}{cccc} k=1 & k=2 & k=3 & k=4 \end{array} & \\ \begin{pmatrix} 1 & 1 & 1 & 0 \\ \mathbf{1} & \mathbf{1} & 1 & 0 \\ 0 & 0 & \mathbf{1} & 0 \end{pmatrix} & \begin{array}{c} l=1 \\ l=2 \\ l=3 \end{array} \end{array}$$

More formally, for a job j and a skill k, the minimal required level l_{jk}^{min} is equivalent to the maximal qualification level in matrix r_{jkl}^{min} (see bold values in example matrix r_{jkl}^{min}):

$$l_{jk}^{min} = \max\{l \in L \mid r_{jkl}^{min} > 0\} \qquad \forall j \in J, k \in K \tag{5.15}$$

Similar, we define the maximal level of skill k possessed by employee m as:

$$l_{mk}^{max} = \max = \{l \in L \mid q_{mkl} > 0\} \qquad \forall m \in M, k \in K \tag{5.16}$$

Then, the job-skill-coefficient ε_{mjk} is given as

$$\varepsilon_{mjk} = \begin{cases} \min\left\{1, \frac{l_{mk}^{max}}{l_{jk}^{min}}\right\}, & \text{if } r_{jk1}^{min} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.17}$$

Note that, if skill k is not needed for performing job j ($r_{jk1}^{min} = 0$), ε_{mjk} also takes value 0 even though employee m can be qualified in domain k ($l_{mk}^{max} > 0$). Furthermore, the level of an employee's expertise exceeding the required level does not result in a reduction of job processing time, which is why the min-function in (5.17) bounds ε_{mjk} to value 1. Generally speaking, the contribution ratio cannot exceed 100%. As example, considering employee 2 in Figure 5.2, we have $\varepsilon_{1j1} = 1$, $\varepsilon_{1j2} = 0$, $\varepsilon_{1j3} = 0.6\overline{6}$, $\varepsilon_{1j4} = 0$.

Using ε_{mjk}, we can now derive an employee-job-contribution Q_{mj}, which is the best employee m can contribute to job j over all skills. As each employee can be trained in different domains and, hence, can be deployed for performing different skills, Q_{mj} is defined as the maximum of the job-skill-coefficients:

$$Q_{mj} = \max_{k \in K}\{\varepsilon_{mjk}\} \qquad \forall j \in J, m \in M \tag{5.18}$$

For the example above, we obtain $Q_{1j} = \max\{1, 1, 1, 0\} = 1$, $Q_{2j} = \max\{1, 0, 0.6\overline{6}, 0\} = 1$, $Q_{3j} = \max\{1, 1, 0.3\overline{3}, 0\} = 1$, $Q_{4j} = \max\{0, 0, 1, 0\} = 1$, $Q_{5j} = \max\{0, 0, 0.3\overline{3}, 0\} = 0.3\overline{3}$, $Q_{6j} = \max\{0, 0, 0, 0\} = 0$.

To this end, the set of redundant employees has to be identified. Employees are called redundant if they cannot contribute to the job progress though $Q_{mj} > 0$. For instance, employees 4 and 5 can perform operations only in domain $k = 3$ but only one employee is required in this domain for the example job j. We thus define for each job j the sets of employees M^{jc} with contribution in *identical* skill domains. Here, c refers to the index of such employee set and C^j denotes the index set for job j such that $c \in C^j$. Considering a pair of employees m and m', we say that $m, m' \in M^{jc}$, if

$$\begin{array}{lllll} \varepsilon_{mjk} > 0 & \text{for all} & k \in K & \text{where} & \varepsilon_{m'jk} > 0 \quad \text{and} \\ \varepsilon_{mjk} = 0 & \text{for all} & k \in K & \text{where} & \varepsilon_{m'jk} = 0 \end{array}$$

For instance, comparing ε_{mjk} for employees 1 and 3, we see that both employees can perform operations in domains 1, 2 and 3 $(\varepsilon_{1j1}, \varepsilon_{1j2}, \varepsilon_{1j3}, \varepsilon_{3j1}, \varepsilon_{3j2}, \varepsilon_{3j3} > 0)$ while $\varepsilon_{1j4} = \varepsilon_{3j4} = 0$. Similar, employees 4 and 5 can contribute only in domain 3 as $\varepsilon_{4j3}, \varepsilon_{5j3} > 0$. Whereas for domains 1, 2 and 4, we have $\varepsilon_{4j1} = \varepsilon_{4j2} = \varepsilon_{4j4} = \varepsilon_{5j1} = \varepsilon_{5j2} = \varepsilon_{5j4} = 0$. From this, $C^j = \{1, 2\}$ with $M^{j1} = \{1, 3\}$ and $M^{j2} = \{4, 5\}$. Next, we denote by τ_{jc} the number of employees actually required from set M^{jc} for performing job j. Since employees $m \in M^{jc}$ are proficient in the same domains, it is sufficient to examine skill-coefficients of any single employee from the respective set, say employee $m^{'}$, to derive τ_{jc}:

$$\tau_{jc} = \sum_{k \in \mathcal{K}} r_{jk1} \qquad j \in J, c \in C^j, \mathcal{K} = \{k \in K \mid \varepsilon_{m'jk} > 0\} \tag{5.19}$$

To compute τ_{jc} for set $c = 1$, we sum up the service requirements of job j at level $l = 1$ for skill domains $k = 1$, $k = 2$ and $k = 3$ as employees 1 and 3 are proficient in these three domains. Here, we get $\tau_{j1} = 3 + 2 + 1 = 6$. Since employees 4 and 5 in set $c = 2$ are experienced only in domain $k = 3$, $\tau_{j2} = r_{j31} = 1$. Let σ_{jtc} be a new decision variable that denotes the number of redundant employees in team t for set c. Suppose job j is performed by team t. Then, σ_{jtc} is calculated as

$$\sigma_{jtc} = \max \left\{ 0, \sum_{m \in M^{jc}} Q_{mj} \cdot x_{mt} - \tau_{jc} \right\} \qquad \forall j \in J, c \in C^j, t \in T \tag{5.20}$$

To now derive the effective team size, we have to reduce the aggregated employee-job-contribution of team t by the number of redundant employees in sets $c \in C^j$:

$$\sum_{m \in M} Q_{mj} \cdot x_{mt} - \sum_{c \in C^j} \sigma_{jtc} \qquad \forall j \in J, t \in T \tag{5.21}$$

To model the sequential processing time, we introduce a continuous variable δ_j that measures the additional time needed for performing job j if the team size is below Max_j. To this end, we define parameter ς_j that specifies the time increase per one employee that has to perform job operations sequentially. The number of such employees corresponds to

$$Max_j - \left(\sum_{m\in M} Q_{mj} \cdot x_{mt} - \sum_{c\in C^j} \sigma_{jtc}\right) \qquad \forall j \in J, t \in T \tag{5.22}$$

Using the introduced notation, STRSP with sequential use of skill is formulated as follows:

maximize:

$$\alpha \cdot \sum_{t\in T} \sum_{i\in J^0} \sum_{j\in J} z_{tij} - \beta \cdot \sum_{m\in M} \sum_{t\in T} x_{mt} - \gamma \cdot \sum_{t\in T} \sum_{j\in J} f_{tj} - \theta \cdot \left(\sum_{j\in J} \delta_j + \sum_{j\in J} \sum_{t\in T} \sum_{c\in C^j} \sigma_{jtc}\right) \tag{5.23}$$

subject to:

(5.4) – (5.9), (5.11) – (5.14) and

$$\delta_j \geqslant \varsigma_j \cdot \left(Max_j - \left(\sum_{m\in M} Q_{mj} \cdot x_{mt} - \sum_{c\in C^j} \sigma_{jtc}\right)\right) - \mathcal{M} \cdot \left(1 - \sum_{i\in J^0} z_{tij}\right) \qquad \forall j \in J, t \in T \tag{5.24}$$

$$\delta_j \geqslant 0 \qquad \forall j \in J \tag{5.25}$$

$$\sigma_{jtc} \geqslant \sum_{m\in M^{jc}} Q_{mj} \cdot x_{mt} - \tau_{jc} - \mathcal{M} \cdot \left(1 - \sum_{i\in J^0} z_{tij}\right) \qquad \forall j \in J, c \in C^j, t \in T \tag{5.26}$$

$$\sigma_{jtc} \geqslant 0 \qquad \forall j \in J, c \in C^j, t \in T \tag{5.27}$$

$$s_{tj} + p_j + \delta_j \leqslant f_{tj} + \mathcal{M} \cdot \left(1 - \sum_{i\in J^0} z_{tij}\right) \qquad \forall j \in J, t \in T \tag{5.28}$$

Objective (5.23) extends objective (5.3) by additionally minimizing the total increase in job processing time weighted by θ. Constraints (5.24)-(5.25) compute the extension of the processing time for each single job j under team t that serves this job. Constraints (5.26)-(5.27) represent a general reformulation of (5.20). Specifically, the number of redundant employees with an identical job-skill-contribution is given for each job-team pair. Constraints (5.28) define the completion time of job j if performed by team t including the additional time δ_j.

5.3 Adaptive Large Neighborhood Search

The optimization models presented in Section 5.2 can assess the impact of a sequential skill setting on the achieved service level only partially due to the limited instance size that can be solved by a commercial MIP solver like CPLEX. To demonstrate the effect on larger problem instances, we propose an adaption of ALNS that was successfully applied to solve the STRSP with simultaneous use of skill by Cordeau et al. (2010) and Kovacs et al. (2012). Compared to their method, we propose new schemes to design the teams, two additional destroy operators, and a restart mechanism based on an iterative search heuristic. To this end, we adapt existing destroy and repair operators to the sequential skill setting. In the following subsections, we describe each component of ALNS in more detail.

5.3.1 Initial Solution

An initial solution is obtained by a two-phase approach. During the first phase, a constructive heuristic is used to initialize the team configuration, see Section 5.3.1.1. Trying to find the compromise between the size and the number of created teams, the heuristic oscillates between three schemes designed to create teams. The *minimum scheme* assigns employees based merely on job qualification requirements. Hence, it guarantees that the team size is at least as large as Min_j. The *maximum scheme* attempts to extend the teams up to Max_j. The *average scheme* represents an intermediate stage between the two other schemes and creates teams of size $\frac{Min_j+Max_j}{2}$. For the ease of notation, we denote the schemes by **MinS**, **MaxS** and **AvS** correspondingly. For each scheme, routes for the teams are constructed using an iterative insertion procedure, see

Section 5.3.1.2. Finally, the solution with the best value of the objective function is selected.

5.3.1.1 Construction of Teams

The procedure starts by initializing a sequence in which jobs that have not been assigned so far are examined. We denote these jobs by J^u. The jobs are sorted by a *job complexity* factor ϑ_j in descending order. The job complexity incorporates not only the job's skill requirements but also the maximum number of employees Max_j required by the job:

$$\vartheta_j = \xi \cdot Max_j + \mu \cdot \sum_{k \in K} \sum_{l \in L} r_{jkl} \qquad \forall j \in J^u \tag{5.29}$$

Thereby, the weights ξ and μ are used to express the priority of both components, where we assume $\xi > \mu$. The rational behind this is explained as follows. More challenging jobs require a larger number of more qualified employees. Teams of a larger size are, in turn, not only qualified for a larger number of jobs but can perform the assigned jobs in a shorter period of time due to a large number of simultaneously performed operations.

Iterating over the created job sequence, the algorithm generates three team variants for each candidate job until there are no further uninspected jobs or no free employees are available. Initially, all employees in set M are considered as „free“. To construct the **MinS**-team, available employees $m \in M$ are incrementally assigned to the team until all service requirements are covered. Thereby, we first select employees with a lower value of their *skill gap factor*. The *skill gap factor* cumulates the sum of a job's skill requirements that are still unmet after employee m has reinforced the team. Formally, we have

$$\varepsilon_{jm} = \sum_{k \in K} \sum_{l \in L} max \left\{ 0, r_{jkl} - \sum_{t \in T} \sum_{m' \in M \setminus \{m\}} q_{m'kl} \cdot x_{m't} - q_{mkl} \right\} \qquad \forall j \in J, m \in M \tag{5.30}$$

Giving the preference to better qualified employees, we increase the number of jobs that can be potentially performed by the team in the later optimization process.

To generate the **MaxS** or **AvS**-team, we reinforce the **MinS**-team by adding further employees until the team size attains Max_j or $\frac{Min_j + Max_j}{2}$ respectively. Note that this time, we sort the employees in set M by the *aggregated skill level* in ascending order to guarantee that enough qualified employees remain available at the end of the procedure to construct further teams. The *aggregated skill level* is defined as follows:

$$\epsilon_m = \sum_{k \in K} \sum_{l \in L} q_{mkl} \qquad \forall m \in M \tag{5.31}$$

Once a team is constructed, employee set M is updated and the algorithm proceeds to the next job for which a team is to be built yet.

5.3.1.2 Construction of Routes

Once the teams are formed, the iterative search heuristic is applied to create the routes. The algorithm starts by initializing the job sets J_t^{qual} that the created teams $t \in T$ can potentially perform. Formally,

$$J_t^{qual} = \left\{ j \in J^u \mid \sum_{m \in M} q_{mkl} \cdot x_{mt} \geqslant r_{jkl} \ \forall k \in K, l \in L \right\} \qquad \forall t \in T \tag{5.32}$$

Next, the teams are considered one by one in sequential manner. Thereby, we start with the team that is qualified for the smallest number of jobs. For this team, a set of temporary routes is initialized in order to diversify the search. The number of temporary routes corresponds to the number of *seed jobs* J_t^{seed} that can serve as starting point of the route. To identify set J_t^{seed}, we derive the minimal earliest starting time s_{min}^l among all jobs $j \in J_t^{qual}$ for which team t is qualified. Further, we define an additional parameter λ specifying the maximal positive deviation from s_{min}^l. We set the bound λ to control the number of seed jobs and, thus, the solution quality and the computational effort. The parameter tuning is described in Appendix A. Formally, J_t^{seed} is defined as follows:

$$J_t^{seed} = \{ j \in J_t^{qual} \mid a_j \leqslant s_{min}^l + \lambda \} \qquad \forall t \in T \tag{5.33}$$

Starting from each seed job in J_t^{seed}, we define one temporary route by incrementally inserting unprocessed jobs if job j's time window complies with the current working time of the team f_{ti}, i.e. if $f_{ti} + d_{ij} \leqslant b_j\}$. Here, index i refers to the last job served by team t. Among all jobs that fulfill this criterion, the job with the least increment to the routing cost is inserted into the next feasible tour position. The selected job is then removed from sets J_t^{qual} of all teams $t \in T$. Formally, the routing cost of team t are given as

$$Cost_t^r = f_t + d_{ij} + p_j + \delta_j - w_j \cdot W. \tag{5.34}$$

Here, binary parameter w_j incorporates the ability to extend the tour. More precisely, w_j takes value 1 if at least one further job can be inserted into the tour after job j which reduces the routing cost by some value W, $w_j = 0$ otherwise.

The process is continued until no more jobs are can be inserted. The quality of a temporary route of team t is assessed by the objective function:

$$obj = \alpha \cdot \sum_{i \in J^0} \sum_{j \in J} z_{tij} - \beta \cdot \sum_{j \in J} f_{tj} \tag{5.35}$$

If a newly generated temporary route yields a higher objective function value obj than a previously determined one, the new route is stored as a current best solution.

5.3.2 Destroy Operators

We now describe four destroy operators used in the algorithm. We introduce a time- and a complexity-related operator. The random and employee related destroy have been proposed by Cordeau et al. (2010) and are adapted here to the sequential skill setting. Whenever applicable, the number of destroyed jobs N is defined as $N = n \cdot |J^s|$, where n denotes the destroy ratio and J^s refers to the number of currently scheduled jobs. The entire search is divided into time segments that correspond to a certain number of iterations. Initially, n is set to a prescribed minimal value n_{min}. At the end of each time segment, we increase n by 0.1. When n reaches its maximal value n_{max}, the algorithm restarts from n_{min}. Note that the bounds are set differently for small and large problem instances. More precisely, we gradually increase n in the interval [0.6, 0.9] for small sized instances while for large sized instances $n \in [0.3, 0.6]$. The parameter tuning is described in Appendix A.

5.3.2.1 Time Related Destroy

The time related operator selects jobs that are most closely related to each other in terms of their start time, see Ropke and Pisinger (2006). Thereby, a first job is selected randomly. The main idea behind this operator is that it would be easier to relocate jobs within the same time period than within different ones. Two jobs are considered as time related if $s^* - t^* \leqslant s_{tj} \leqslant s^* + t^*$, where s^* denotes the start time of the candidate job in the current solution and t^* denotes a maximal deviation from s^*.

5.3.2.2 Maxj Related Destroy

This operator first picks out $0.5 \cdot N$ jobs with highest Max_j values. As these jobs are challenging in terms of both, the required processing time and the number of required employees, removing these jobs from a solution creates large time gaps in the routes and frees a large number of redundant employees. This, in turn, creates more freedom for the repair heuristic to perform an exchange of jobs. However, as it might be difficult to exchange only the most difficult jobs, further $0.5 \cdot N$ jobs are selected randomly.

5.3.2.3 Random Destroy

The random destroy is based on the removal heuristic proposed by Cordeau et al. (2010) and Kovacs et al. (2012). The destroy operator randomly selects N scheduled jobs and removes them from the assigned routes.

5.3.2.4 Employee Related Destroy

Having removed jobs by one of the three destroy operators from Section 5.3.2.1 to 5.3.2.3, all teams are examined in order to identify redundant employees that can be removed without turning the current job-team assignment infeasible. These employees are used to form new teams or to reinforce the current teams, such that jobs that have not been assigned so far can be processed. The idea to use redundant employees has already been considered in the context of STRSP by Cordeau et al. (2010), Kovacs et al. (2012) and Anoshkina and Meisel (2020). However, the feasibility analysis is now constrained by two perspectives: skill feasibility and processing time feasibility. Skill feasibility guarantees that the aggregated skills of the remaining members in team t are still sufficient to perform all assigned jobs. Formally, $\sum_{m \in M \setminus \{m'\}} q_{mkl} \cdot x_{mt} \geqslant r_{jkl} \; \forall k \in K, l \in L$, where m' denotes the candidate employee checked for removal. Time feasibility refers to the changes in processing times due to changes of the team size. Therefore, when removing employee m', we update the δ_j values and the start times for all jobs assigned to the team. A removal is successful if the new start times still comply with time window constraints. If more than one employee could be removed, we first select the employee with a higher aggregated skill level ϵ_m.

5.3.3 Repair Operators

Following Kovacs et al. (2012), we implement three types of insertion heuristics to repair the partly destroyed solution. In the following, we briefly recall the main ideas behind each repair approach and explain how the sequential skill aspect is embedded into the repair process.

5.3.3.1 Greedy Insertion

The operator calculates the insertion cost for all unprocessed jobs $j \in J^u$ and all feasible insertion positions $p \in P$ within the routes of all teams. In each iteration, the algorithm selects the job with the least cost insertion position and inserts this job at its best position. The process continues until there are no more unprocessed jobs or no more feasible insertion positions. We define insertion cost $\triangle f(j,p)$ as

$$\triangle f(j,p) = \sum_{j' \in J_t} f_{tj'} + \varrho_j \cdot \mathcal{Q} \tag{5.36}$$

where J_t denotes the jobs assigned to team t in the current iteration including job j and ϱ_j whether job j blocks the tour. The tour is blocked if no further jobs can be scheduled after the insertion of job j. In this case, $\varrho_j = 1$ and $\triangle f(j,p)$ is increased by a sufficiently large penalty $\mathcal{Q}$. If at least one further job can be assigned at $p-1$ or at the last tour position, $\varrho_j = 0$.

5.3.3.2 Regret Insertion

The regret operator is a more sophisticated repair operator that incorporates a look-ahead. At each decision point, the algorithm selects the candidate job with the maximal regret value that corresponds to the difference in cost between the first best $\triangle f(j,p)^1$ and the k-th insertion position $\triangle f(j,p)^k$. In our experiments, we use $k = 2$ and $k = 3$. Generally speaking, the regret represents the opportunity cost that have to be paid if the best assignment becomes infeasible due to other insertion decision. Formally, the best insertion position p^* is defined as

$$p^* = arg\ \max_{p \in P}(\triangle f(j,p)^k - \triangle f(j,p)^1) \tag{5.37}$$

5.3.3.3 Sequential Insertion

The sequential insertion operator repairs one route at a time. Starting from the first route, the algorithm iteratively inserts jobs at their minimum cost position until no further unprocessed jobs are available or the routes of all teams have been examined. As the routes are examined one by one, the sequence in which teams are iterated constitutes the most important factor that impacts the solution quality. For some instances, it may be more suitable to rank the teams in the decreasing order of the number of jobs that are still assigned to them after destroy. The reverse order may be more suitable for some other instances. We therefore consider both orders in the sequential insertion.

5.3.3.4 Skill-Time-Based Estimation of Insertion Position

Since job processing times and team composition are interrelated, $\triangle f(j,p)$ cannot be accurately identified until the team composition becomes known. To address this issue, we create a fictive team for each job-team combination. The procedure is as follows. Recall that the set of redundant employees is updated after each destroy process, see Section 5.3.2.4. We use this set to extend the team first to its **MinS** configuration and then to its **MaxS** configuration. Subsequently, temporary value of parameter δ_j are computed for all considered teams and jobs $j \in J^u$. As the completion times of already assigned jobs can be affected by a new team composition, we have to reset δ_j also for jobs $j \in J_t$. Afterwards, value $\triangle f(j,p)$ are defined and the best insertion position is identified. To this end, the corresponding destroyed team is replaced by its fictive counterpart. The list of redundant employees is updated correspondingly, see Section 5.3.2.4. Note that as the number of redundant employees is varied, we have to recompute the insertion cost of all positions after each insertion operation.

As a numerical example consider the process illustrated in Figure 5.3. Suppose there is one insertion position for job j in each of the routes of team 1 and team 2. Three redundant employees 4, 5 and 6 are available after the destroy process. To estimate the insertion cost, two fictive teams are created. To meet the job's qualification requirement, team 1 is reinforced by employees 4, 5 and 6. Due to the limited number of redundant employees only **MinS** configuration can be constructed for this team but not **MaxS**. By definition (5.24), $\delta_j = \varsigma_j \cdot (5-4) = \varsigma_j$. For team 2, **MinS** is first composed by adding employee 4. Next, the effective team size is extended to **MaxS** ($Max_j = 5$) by also adding employee 6. From this, $\delta_j = \varsigma_j \cdot (5-5) = 0$. Let the insertion position of team 2's route be the best one. Then, the set of redundant employees is updated as follows. It

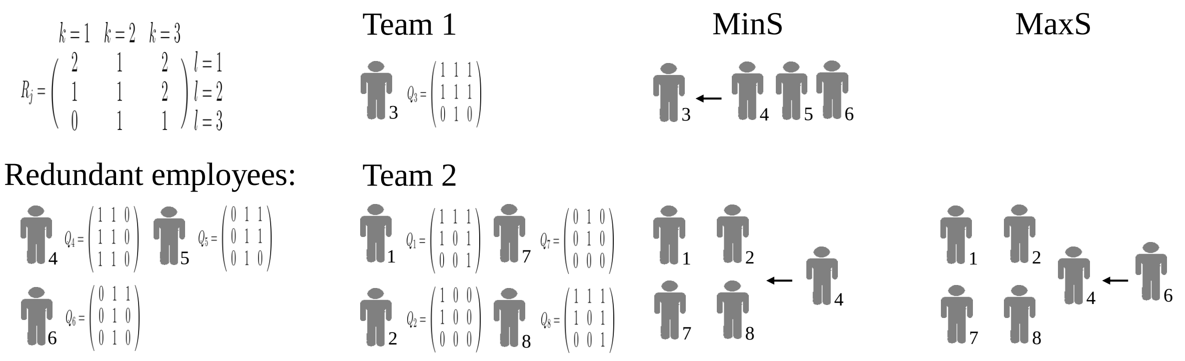

$k=1$ $k=2$ $k=3$
$R_j = \begin{pmatrix} 2 & 1 & 2 \\ 1 & 1 & 2 \\ 0 & 1 & 1 \end{pmatrix}$ $l=1$ $l=2$ $l=3$
Redundant employees:
4 $Q_4 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$
5 $Q_5 = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$
6 $Q_6 = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$
Team 1
3 $Q_3 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$
Team 2
1 $Q_1 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$
7 $Q_7 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$
2 $Q_2 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$
8 $Q_8 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$
MinS
3 4 5 6
1 2 7 8 4
MaxS
1 2 7 8 4 6

Figure 5.3: Repair process

can be seen that employees 7, 4 and 2 can be removed from the team without violating skill constraints. Note that the removal of employee 4 or employees 4 and 2 reduces the effective team size to 4 or 3 and, thus, extends the job completion time by ς_j or $2 \cdot \varsigma_j$ respectively. However, if time window constraints are still satisfied, all three employees are considered redundant.

5.3.4 Restart Mechanism

The main objective of the restart procedure is to diversify the search and prevent an early stagnation. If the solution has not been improved for a certain number of iterations I^R, we destroy the complete schedule and restart by constructing a new initial solution. Recall that initial solutions are generated by an iterative search method. From this, the skill requirement of the previously examined job affects the skill level as well as the number of employees available for the subsequent team construction process. For instance, the sorting of jobs by complexity applied in Section 5.3.1.1 forms a schedule with a structure, where the team size and the number of allocated jobs decreases while moving from the top to the bottom of the team list. To generate a new solution, we have to modify the order in which the jobs are examined. Therefore, the restart mechanism randomly shifts a number of jobs in the list, which gives more priority to less challenging jobs when creating the team. The level of diversification, i.e. the number of shifted jobs, is sampled uniformly from the set $\{0.3 \cdot |J|, 0.5 \cdot |J|, 0.6 \cdot |J|\}$. Eventually, the routes are reconstructed by the iterative insertion heuristic described in Section 5.3.1.2 and the solution is updated according to acceptance criteria defined in Section 5.3.4.1.

5.3.4.1 Acceptance Criteria

The proposed ALNS framework includes 15 pairs of destroy and repair operators. To select a particular pair in each iteration of the ALNS, we follow the approach suggested by Ropke and Pisinger (2006) and Kovacs et al. (2012) that is based on a roulette wheel principle. To decide about the acceptance of the generated solution, we apply an acceptance probability that follows the principal of simulated annealing as proposed in Ropke and Pisinger (2006).

5.4 Computational Study

5.4.1 Generation of Test Instances

This section describes the test instances that are used to evaluate the performance of the proposed solution approaches. In order to capture a wide range of problem structures, we created 20 data sets. Each data set is composed of 10 problem instances of identical size. The size is determined by the number of jobs, employees, skill domains and the competence levels. The number of jobs and employees vary from 4 to 500 and from 2 to 100, correspondingly. Instances with up to 10 jobs are referred to as small, with up to 40 as medium and with 80 jobs and more as large instances. The number of skill domains and competence level is set to $|K| = 3$ and $|L| = 3$ for all instances in each data set. In total, 200 problem instances are examined in the experiments.

The input parameters are generated following the procedure described in Anoshkina and Meisel (2020). To compute the extension of the processing time, we bound $\varsigma_j = 30$ for all jobs.

The experiments were conducted on an Intel(R) Core (TM) i7-7700 3.60 GHz with 32 GB of RAM. The MIP models were solved using CPLEX 12.8. The ALNS heuristic was implemented in Java 1.8.0. We set a runtime limit of 3600 seconds for both the exact and heuristic approach.

5.4.2 Results of CPLEX MIP Solver

In this section, the mathematical models are evaluated on small and medium-size instances. Table 5.1 reports aggregated computational results obtained from CPLEX. These results have been obtained by applying the following weighting for the coefficients of the objective function: $\alpha = 1$, $\beta = 0.0001$, $\gamma = 0.0001$ and $\theta = 0.0001$. The first column denotes the instance size. The next columns list average solution values obtained from 10 instances in the corresponding instance set for each skill setting. Here, we denote the settings by **SM** (simultaneous use of skill) and **SQ** (sequential use of skill). The first two columns in each block show the number of variables (column Var.) and the number of constraints (column Con.) in the optimization model. Column Z gives the number of jobs that are performed in the solutions. Columns M and T report the total number of assigned employees and the number of active teams in the route plans. Column *Delta* denotes for **SQ** the total extension of job processing times. We omit *Delta* for **SM** since the job processing times are fixed for this setting. Instead, in column Z_δ we report the number of jobs scheduled under **SM** that could also be performed if the sequential processing times would be applied. Column *CPU* gives the consumed runtime in seconds. Finally, column *GAP* reports the percentage gap of the generated solution with respect to the lower bound value LB reported by CPLEX. Formally, GAP = (Objective - LB) /LB.

Based on the results in Table 5.1, we observe that **SQ** consistently generates solutions with a lower number of performed jobs. This is an expected observation due to the extended processing times involved in **SQ**, which are reported in column *Delta*. For the same reason, we observe that total completion time values under **SQ** can be higher even for instances with a lower number of assigned jobs, e.g. see instances 4×4, 6×6 and

Table 5.1: CPLEX solution for small and medium sized instances

Instance	Simultaneous Use of Skill (**SM**)									Sequential Use of Skill (**SQ**)								
$\|J\| \times \|M\|$	Var.	Con.	Z	M	T	Z_δ	F	CPU	GAP	Var.	Con.	Z	M	T	$Delta$	F	CPU	GAP
4 × 2	158	74	1.0	0.9	0.6	0.9	292	0.01	0%	189	84	0.9	1.1	0.6	21	283	0.01	0%
4 × 4	312	156	3.2	3.3	1.2	2.7	939	0.03	0%	378	175	2.9	3.9	1.2	77	941	0.05	0%
6 × 3	381	198	2.8	2.5	1.1	2.2	728	0.04	0%	450	218	2.5	2.8	1.1	60	707	0.05	0%
6 × 6	756	414	5.1	5.7	2.1	4.4	1387	0.25	0%	936	471	4.8	6.0	1.8	152	1519	0.43	0%
8 × 4	732	412	4.5	3.5	1.3	3.1	1297	1.19	0%	859	448	4.0	4.0	1.3	75	1252	0.68	0%
8 × 8	1456	856	7.5	7.5	2.8	6.0	2143	58.39	0%	1793	965	6.9	8.0	2.3	135	2178	424.23	0%
10 × 5	1235	740	5.8	4.5	1.9	3.7	1678	27.99	0%	1453	805	5.1	5.0	1.7	87	1576	37.08	0%
10 × 10	2460	1530	9.7	9.4	3.7	7.5	2663	3071.87	1%	3027	1719	9.2	10.0	3.0	227	2999	3368.90	6%
15 × 7	3095	2065	9.8	6.9	2.8	6.8	2820	3271.00	27%	3618	2229	8.1	7.0	2.4	149	2467	3269.07	39%
15 × 15	6615	4545	15.0	14.8	5.5	11.7	4136	3600.00	0%	8131	5086	13.9	15.0	4.3	378	4576	3600.00	8%
20 × 10	6870	4930	14.4	9.8	3.8	9.4	4191	3600.00	28%	7981	5296	11.9	10.0	3.4	191	3669	3600.00	41%
20 × 20	13720	10060	19.6	19.9	7.9	15.1	5281	3600.00	2%	16777	11199	17.7	20.0	5.9	431	5566	3600.00	12%
30 × 15	19905	15570	20.7	14.7	6.2	13.9	5832	3600.00	31%	22885	16626	17.0	15.0	5.1	228	5108	3600.00	43%
40 × 20	43340	35660	24.3	18.8	7.2	15.1	6921	3600.00	39%	49433	37927	19.2	20.0	6.0	351	5850	3600.00	52%

8 × 8. Note that *Delta* represents an average over 10 instances and, thus, we observe values that are not necessarily multiples of the minimal preset value ς_j.

Looking at columns M and T in Table 5.1, we see a clear trend for **SQ** to generate larger teams by allocating more employees to a somewhat lower number of teams.

The average difference in the number of performed jobs under **SM** and **SQ** ranges between 0.1 and 0.7 for small sized instances. This can be partially explained by the above mentioned tendency of **SQ** to compensate the longer processing times by increasing team sizes. Indeed, looking at column Z_δ, we see that the actual number of performed jobs in a **SM** solution would be lower if sequential processing times would be applied here. For medium sized instances, we observe that a higher share of jobs (1.1 - 5.1 jobs) of **SM** solutions becomes infeasible. However, we also observe very high optimality gaps and drastically increasing computational time for both, **SM** and **SQ**. Due to these high GAP values, it is unclear if only the corresponding skill concept accounts for the difference in performance. Intuitively, we can expect higher differences with increasing problem size. This is confirmed by the results of the subsequent experiment.

5.4.3 Results of ALNS

In this section, we compare the effect of **SM** and **SQ** for the entire set of instances. The comparison is performed by applying the ALNS heuristic to each skill setting and each problem size. Since the team composition under **SM** has no impact on the job processing time, we use only the **MinS** to create fictive teams during the repair process. However, to achieve a fair comparison between both skill settings, we iterate over all three team configuration schemes in the initial solution and the restart mechanism no matter whether **SM** or **SQ** is applied. The parameter settings are also the same for both problems considered. To set the parameters, preliminary experiments have been conducted on subsets of instances, see Appendix A. The preliminary experiments show that the sequential search heuristic can produce high quality solutions that are very close to the optimal ones, see Appendix B. As the solution converges fast, we opt to perform I = 1.000 iterations. In order to randomize and accelerate the search, we employ the restart procedure described in Section 5.3.4 every I^R iterations. For completeness, we state all further parameters in Table 5.2.

Table 5.2: Parameter setting of ALNS

Parameter	I	I^R	$t*$	W	λ	ε	μ	Q
Value	1000	20	120	100	60	2	1	500

Table 5.3 shows the results obtained for each skill setting. From the reported Z values, we can clearly see that the service level achieved under **SQ** is lower in comparison to the corresponding **SM** values for all instances. The gap between both settings does not exceed 0.6 jobs for small sized instances. The difference widens further with an increase of the problem size. For medium sized instances, the gap ranges between 0.9 - 5.9 jobs. These observations are in line with results observed in Section 5.4.2.

Considering the quality of the solutions, we observe that ALNS and CPLEX perform similarly well for small instances. Specifically, the maximal average deviation does not exceed 0.1 job for instances 8×4 and 10×5. For medium sized instances ALNS clearly outperforms CPLEX. For large sized instances, ALNS obtains solutions of high quality within a few minutes of computation time. Here, we also observe a further decrease in **SQ** service levels from around 11.7 to 63.5 jobs. Moreover, values reported in column

Table 5.3: ALNS solution

Instance	Simultaneous Use of Skill (**SM**)						Sequential Use of Skill (**SQ**)					
$\lvert J\rvert \times \lvert M\rvert$	Z	M	T	Z_δ	F	CPU	Z	M	T	$Delta$	F	CPU
4 × 2	1.0	0.9	0.6	0.9	292	1.53	0.9	1.1	0.6	21.0	283	1.52
4 × 4	3.2	3.5	1.2	2.7	943	2.70	2.9	4.0	1.3	82.5	948	2.72
6 × 3	2.8	2.5	1.1	2.2	728	2.65	2.5	2.8	1.2	69.0	733	2.65
6 × 6	5.1	5.6	2.0	4.2	1417	2.90	4.8	5.9	1.9	174.0	1554	2.89
8 × 4	4.5	3.7	1.3	3.0	1318	2.80	3.9	4.0	1.3	117.0	1224	2.85
8 × 8	7.5	7.3	2.6	5.6	2244	3.13	6.9	7.8	2.3	165.0	2274	3.23
10 × 5	5.7	4.5	1.8	3.9	1695	2.91	5.1	4.9	1.7	105.0	1608	3.03
10 × 10	9.7	9.5	3.2	7.1	2838	3.51	9.2	10.0	3.2	307.5	3102	3.57
15 × 7	9.8	6.8	2.6	5.4	2977	3.39	8.5	6.8	2.6	273.0	2854	3.36
15 × 15	15.0	14.7	5.1	10.2	4394	4.48	14.0	14.9	4.3	469.5	4843	4.78
20 × 10	14.8	10.0	3.8	9.0	4476	3.52	12.7	10.0	3.6	327.0	4270	3.38
20 × 20	20.0	19.6	6.5	14.9	5743	4.51	19.1	20.0	6.0	681.9	6503	4.42
30 × 15	23.8	15.0	5.6	13.7	7129	4.65	19.9	15.0	5.3	519.4	6561	4.48
40 × 20	32.1	20.0	7.1	17.6	9572	6.25	26.2	20.0	6.9	639.0	8589	5.98
80 × 30	55.8	30.0	11.3	29.5	16513	15.88	44.1	30.0	10.1	945.0	14250	15.97
100 × 40	73.2	40.0	15.5	38.7	21883	31.82	59.0	40.0	14.4	1360.0	18933	31.71
200 × 50	115.0	50.0	20.8	57.0	33888	106.75	90.2	50.0	20.5	1674.0	28797	109.71
300 × 80	183.1	80.0	32.3	91.4	53774	428.48	143.0	80.0	28.8	2360.3	45171	462.16
400 × 80	207.8	80.0	33.3	97.3	60834	669.10	160.0	80.0	31.8	2713.5	50489	675.36
500 × 100	266.0	100.0	41.8	121.8	77784	1260.03	202.5	100.0	39.2	3525.5	64033	1365.72

Z_δ cast this difference in a more dramatic light. Here, we see that a significantly higher share of jobs become infeasible if sequential skill processing times would be applied to a **SM** solution. This share attains 50% or more for all large sized problems. This finding strongly implies the importance to integrate skill-based processing times into the operations management of multi-skilled teams.

Looking at columns M and T, we observe that **SQ** consistently creates larger teams for all instances, compared to **SM**. However, compared to CPLEX, the total number of assigned employees is slightly lower, which is due to the heuristic's tendency to disconnect as many as possible employees in the repair process. This, in turn, explains higher *Delta* values and total completion times.

Finally, the results illustrate that the CPU time tends to be larger with problem size. However, most problem instances are solved within in a fraction of the given time limit. Even for the largest instances, the CPU time lies significantly under the preset time limit of 3600 seconds. This indicates that the heuristic can be successfully applied for solving problems of even large size.

5.5 Conclusions and Future Research

Although many studies have investigated variants of STRSP, the relationship between team competences and job processing times has hardly been questioned so far. This paper presented a skill-based concept that incorporates the team composition into an estimation of processing times. Based on this concept, two linear mixed integer optimization models have been presented for simultaneous and sequential use of skill. Both skill settings have been evaluated by means of an ALNS framework. The framework uses existing operators and new operators designed specifically for the problem type considered here. Moreover, a new restart mechanism has been implemented to accelerate the search. The computational experiments show that the sequential skill concept can be successfully integrated into the planning. Moreover, neglecting the relationship between job processing times and team composition can have an extremely negative impact on the schedule feasibility. This holds especially for medium and large problem instances for which a considerable share of the workload cannot be performed within the predefined time windows if a solution would be generated based on simultaneous use of skill but the job processing time would actually depend on a sequential use of skill. Finally, comparing the results of ALNS with the solutions obtained from CPLEX, we have demonstrated that the algorithm can find good quality solution within reasonable time for any medium and large sized instances.

Despite the results achieved, there is still potential for future research. For instance, an important issue is the data uncertainty that can arise in job qualification requirements or travel times. From this, it could be interesting to consider the sequential use of skill within the context of robust optimization.

Appendix A. Parameter Tuning of ALNS

Varying λ, we analyze how the route construction algorithm performs under different numbers of seed jobs generated in each routing subproblem. From the results displayed in Figure 5.4, we see that λ has no impact on the solution quality of small sized instances. This can be explained by a small number of seed nodes and, thus, by a small number of available seed jobs. For medium and large sized instances, we observe a considerable increase in the number of performed jobs for $\lambda \in [0, 60]$ whereas λ strictly above 60 does not necessarily contribute to a further improvement. In view of these results, we set $\lambda = 60$.

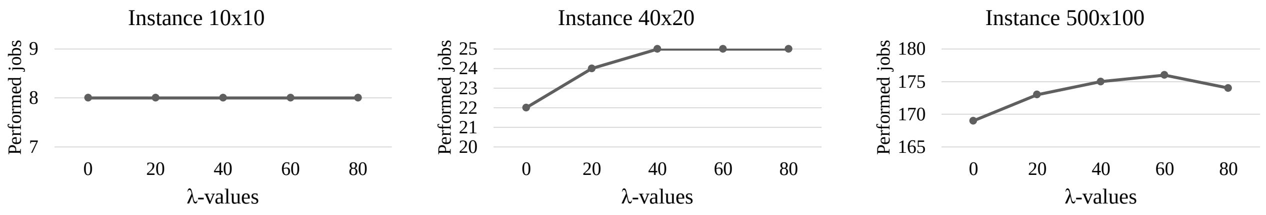


Figure 5.4: Influence of λ on the solution quality

Next, we demonstrate the impact of the destroy ratio n on the solution quality. To assess the effect, we test the heuristic performance with n ranging between 0.3 and 0.9. Table 5.4 reports average results of 10 instances obtained for two small, medium and large sized data sets. Generally, the larger n is, the more freedom the repair algorithm has for exchange and, thus, it is more probable to find a new best neighbor. Note that in the context of **SQ**, also higher n values unlikely lead to improvements if the team destroy does not create a sufficiently large pool of redundant employees, see Section 5.3.3.4. Since the number of scheduled jobs J^s for large sized instances is relatively high, small n values (e.g. $n = 0.3$) can already provide a sufficient number of disconnected jobs and, thus, employees available for reoptimization. However, a too large destroy ratio can slow down the repair process. Therefore, we observe that the solution quality deteriorates with an increase of n for medium and large sized instances.

Table 5.4: Computational results for varied destroy ratios

Instances	Destroy Ratio						
$\|J\| \times \|M\|$	0.3	0.4	0.5	0.6	0.7	0.8	0.9
8×4	3.6	3.7	3.6	3.7	3.8	3.8	3.9
10×10	8.4	8.6	8.6	8.6	9.0	8.7	8.7
20×10	12.0	12.0	12.1	12.2	11.6	11.9	11.6
40×20	25.4	25.6	25.2	25.0	25.6	25.3	24.5
80×30	43.0	43.0	42.6	41.6	43.0	40.3	42.4
100×40	56.2	55.6	55.2	55.9	54.7	54.9	53.7

The situation is rather the reverse for small sized instances. Due to a small number of available employees, a significantly larger destroy ratio is required to create an employee pool sufficient for exchange. Therefore, we notice that the higher n values yield better solution quality. In view of these results, we opt to gradually increase n in the interval [0.6, 0.9] for small sized instances, while for medium and large sized instances n is varied in the interval [0.3, 0.6].

Finally, we demonstrate how the solution quality responds to variations in the number of iterations. Specifically, we compare an ALNS heuristic with an ALNS framework that also embeds the iterative search. Table 5.5 reports the results obtained for one set of small, medium and large sized instances. The first column gives the instance size. The next five columns report the number of jobs (Z) achieved after 1.000, 5.000, 10.000 and 20.000 iterations. Considering the first row, we see that ALNS assigns 9 jobs already after 1.000 iterations if iterative search is invoked. Otherwise, ALNS has only 7 jobs after 1.000 iteration and starts to increase the number of performed jobs not until after 10.000 iterations. The obtained results show that the combination of ALNS with the iterative search is seen to converge significantly faster while the ALNS achieves the same results after approximately 20.000 iterations.

Table 5.5: Solution convergence

Instances	no iterative search				with iterative search
$\|J\| \times \|M\|$	I = 1.000	I = 5.000	I = 10.000	I = 20.000	I = 1.000
10×10	7	7	8	9	9
40×20	24	26	27	28	28
100×40	57	59	60	60	62

Appendix B. Iterative Search Heuristic

To demonstrate the effect of the iterative search method, we report in Table 5.6 the results obtained after 10 iterations for all set of small and medium sized instances. Looking at column Z, we observe that the heuristic achieves a lower service level compared to CPLEX (see Table 5.1) for both **SM** and **SQ**. However, the obtained results are very close to the corresponding results generated by CPLEX. The maximal deviation does not exceed 1 job and ranges between 0.2 and 0.9 for small sized instances and between 0.3 and 1 for medium sized instances. Whereas, for the two last instance sets, we observe even higher service levels of the iterative search compared to CPLEX.

Table 5.6: Iterative search

Instance	Simultaneous Use of Skill (**SM**)						Sequential Use of Skill (**SQ**)					
$\|J\| \times \|M\|$	Z	M	T	Z_δ	F	CPU	Z	M	T	$Delta$	F	CPU
4 × 2	1.0	1.2	0.8	0.9	292	0.26	0.9	1.2	0.7	21.0	283	0.26
4 × 4	3.2	4.0	1.2	2.7	944	0.43	2.9	4.0	1.3	79.5	947	0.42
6 × 3	2.8	3.0	1.3	2.2	734	0.42	2.3	3.0	1.2	72.0	656	0.42
6 × 6	4.8	6.0	2.1	4.1	1276	0.41	4.4	6.0	2.0	121.5	1338	0.42
8 × 4	4.3	4.0	1.5	3.1	1221	0.41	3.7	4.0	1.3	66.0	1110	0.42
8 × 8	7.2	8.0	2.6	5.7	2128	0.42	6.5	8.0	2.3	157.5	2041	0.42
10 × 5	5.6	5.0	1.9	3.9	1640	0.42	4.8	5.0	1.6	79.5	1463	0.42
10 × 10	9.4	10.0	3.6	7.3	2739	0.42	8.3	10.0	3.3	217.5	2598	0.42
15 × 7	9.0	7.0	2.7	6.1	2554	0.42	8.1	7.0	2.6	219.0	2646	0.43
15 × 15	14.5	15.0	5.6	11.1	4177	0.43	12.9	15.0	4.7	426.0	4248	0.43
20 × 10	13.7	10.0	4.0	8.9	4016	0.43	11.5	10.0	3.5	246.0	3647	0.43
20 × 20	19.8	20.0	7.3	14.2	5721	0.43	17.4	20.0	6.0	487.5	5651	0.43
30 × 15	22.0	15.0	5.6	13.0	6352	0.43	18.2	15.0	5.1	415.5	5745	0.44
40 × 20	30.5	20.0	7.4	17.7	8811	0.45	24.7	20.0	7.2	678.4	7908	0.45

Bibliography

Anoshkina, Y. and Meisel, F. (2019). Technician teaming and routing with service-, cost-and fairness-objectives. *Computers & Industrial Engineering*, 135:868–880.

Anoshkina, Y. and Meisel, F. (2020). Interday routing and scheduling of multi-skilled teams with consistency consideration and intraday rescheduling. *EURO Journal on Transportation and Logistics.*, 9:1–18.

Attia, E. A., Duquenne, P., and Le Lann, J. M. (2014). Considering skills evolutions in multi-skilled workforce allocation with flexible working hours. *International Journal of Production Research.*, 52:4548–4573.

Cassera, M. A., Zheng, B., Martinec, D. V., Dunst, C. M., and Swanström, L. L. (2009). Surgical time independently affected by surgical team size. *The American Journal of Surgery*, 198:216–222.

Chen, R., Changyong, L., Dongxiao, G., and Huimin, Z. (2020). A competence-time-quality scheduling mode of multi-skilled staff for it project portfolio. *Computers & Industrial Engineering*, 139:1–17.

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

Chen, X., Thomas, B. W., and Hewitt, M. (2017). Multi-period technician scheduling with experience-based service times and stochastic customer. *Computers & Operations Research*, 82:1–17.

Cordeau, J.-F., Laporte, G., Pasin, F., and Ropke, S. (2010). Scheduling technicians and tasks in a telecommunication company. *Journal of Scheduling*, 13:393–409.

De Bruecker, P., Van Den Bergh, J., Beliën, J., and Demeulemeester, E. (2015). Workforce planning incorporating skills: State of the art. *Journal of Scheduling*, 13:393–409.

Estellon, B., Gardi, F., and Nouioua, K. (2009). High-performance local search for task scheduling with human resource allocation. *Lecture Notes in Computer Science*, 552:1–15.

Fırat, M., Briskorn, D., and Laugier, A. (2016). A branch-and-price algorithm for stable workforce assignments with hierarchical skills. *European Journal of Operational Research*, 251:676–685.

Firat, M. and Hurkens, C. (2012). An improved MIP-based approach for a multi-skill workforce scheduling problem. *Journal of Scheduling*, 15:363–380.

Gutjahr, W. J., Katzensteiner, S., Reiter, P., Stummer, C., and Denk, M. (2010). Multi-objective decision analysis for competence-oriented project portfolio selection. *European Journal of Operational Research*, 205:670–679.

Hashimoto, H., Boussier, S., Vasquez, M., and Wilbaut, C. (2011). A GRASP-based approach for technicians and interventions scheduling for telecommunications. *Annals of Operations Research*, 183:143–161.

Heimerl, C. and Kolish, R. (2010). Work assignment to and qualification of multi-skilled human resources under knowledge depreciation and company skill level targets. *International Journal of Production Research*, 48:3759–3781.

Hurkens, C. A. (2009). Incorporating the strength of MIP modeling in schedule construction. *RAIRO-Operations Research*, 43:409–420.

Karam, A., Attia, E. A., and Duquenne, P. (2017). A MILP model for an integrated project scheduling and multi-skilled workforce allocation with flexible working hours. *IFAC PapersOnline*, 50:13964–13969.

Khalfay, A., Crispin, A., and Crockett, K. (2017). Applying the intelligent decision heuristic to solve large scale technician and task scheduling problem. *International Conference on Intelligent Decision Technologies*, 72:71–81.

Kovacs, A. A., Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2012). Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of Scheduling*, 15:579–600.

Malachowski, B. and Korytkowski, P. (2016). Competence-based performance model of multi-skilled workers. *Computers & Industrial Engineering*, 91:165–177.

Paraskevopoulos, D. C., Laporte, G., Repoussis, P. P., and Tarantilis, C. D. (2016). Resource constrained routing and scheduling: Review and research prospect. *European Journal of Operational Research*, 263:737–754.

Qin, S., Liu, S., and Kuang, H. (2016). Piecewise linear model for multiskilled workforce scheduling problem considering learning effect and project quality. *Mathematical Problems in Engineering*, 4:1–11.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40:455–472.

Walter, M. and Zimermann, J. (2016). Minimizing average project team size given multi-skilled workers with heterogeneous skill levels. *Computers & Operations Research*, 70:163–179.

Zamorano, E. and Stolletz, R. (2017). Branch-and-price approaches for the multiperiod technician routing and scheduling problem. *European Journal of Operational Research*, 257:55–68.

Zha, G. and Zhang, L. (2014). Scheduling projects with multiskill learning effect. *The Science World Journal*, 2014:1–7.

Authors’ Contribution Statement

The contribution of each individual author to the Essays 1, 2, and 3 is defined in relation to Contributor Roles Taxonomy CRediT. Definition of roles can be found under `https://casrai.org/credit/`. The ranking indicates a relative contribution of each author with ① being the highest.

The breakdown of each authors’ contributions to the first essay „Technician Teaming and Routing with Service-, Cost- and Fairness-Objectives“ is as follows:

Category	Y. Anoshkina	F. Meisel
Conceptualization	①	①
Methodology	①	②
Software and Validation	①	
Investigation	①	②
Writing - Original Draft	①	②
Writing - Review Editing	①	②
Visualization	①	

31/03/2021	31/03/2021
Date	Date

Y. Anoshkina	F. Meisel

The breakdown of each authors' contributions to the second essay „Interday Routing and Scheduling of Multi-Skilled Teams with Consistency Consideration and Intraday Rescheduling“ is as follows:

Category	Y. Anoshkina	F. Meisel
Conceptualization	①	②
Methodology	①	②
Software and Validation	①	
Investigation	①	②
Writing - Original Draft	①	②
Writing - Review Editing	①	②
Visualization	①	

31/03/2021	31/03/2021
Date	Date
Y. Anoshkina	F. Meisel

The breakdown of each authors' contributions to the third essay „Robust Optimization Approaches for Routing and Scheduling of Multi-Skilled Teams under Uncertain Job Skill Requirements“ is as follows:

Category	Y. Anoshkina	F. Meisel	M. Goerigk
Conceptualization	①	③	②
Methodology	①	③	①
Software and Validation	①		
Investigation	①	③	②
Writing - Original Draft	①	③	②
Writing - Review Editing	①	③	②
Visualization	①		

31/03/2021	31/03/2021	31/03/2021
Date	Date	Date
Y. Anoshkina	F. Meisel	M. Goerigk

Erklärung zum selbständigen Verfassen der Arbeit

Ich erkläre hiermit, dass ich meine Doktorarbeit „Integration of Workforce Teaming and Routing“ selbstständig und ohne fremde Hilfe angefertigt habe und dass ich als Koautor maßgeblich zu den weiteren Fachartikeln beigetragen habe. Alle von anderen Autoren wörtlich übernommenen Stellen, wie auch die sich an die Gedanken anderer Autoren eng anlehnenden Ausführungen der aufgeführten Beiträge wurden besonders gekennzeichnet und die Quellen nach den mir angegebenen Richtlinien zitiert.

Kiel, 7/12/2020	
Ort, Datum	Unterschrift Y. Anoshkina