Chapter 4

Optimization models and complexity analysis

In this chapter, we will present optimization models for the three problems that were outlined in the previous chapter. Furthermore, we will judge the complexity of the problems. In Section 4.1, we discuss the suitability of an integrated approach and of an alternative hierarchical planning approach for the three problems, i.e., for the project selection problem, the workforce assignment problem, and the utilization leveling problem. The hierarchical approach comprises three stages—one for each problem. We conclude that the hierarchical approach is preferable and present the corresponding optimization models for the three problems in Sections 4.2, 4.3, and 4.4, respectively. For our key problem of assigning workers to projects and allocating project workload, we will discuss limitations of our modeling approach in detail and point out potential remedies in Section 4.3. Additionally, we will present an integrated, monolithic optimization model in Section 4.5. The optimization models are mathematically precise statements of the problems. Each model features one objective function, various sets of constraints, and different sets of decision variables. The constraints define the solution space and allow to check if a solution, which is defined by the values of the decision variables, is feasible. The objective function allows to compare two solutions and to decide which of these two solutions is better. Finally, we will elaborate on the complexity of our three problems of the hierarchical planning approach in Section 4.6. The optimization models together with the evaluation of their complexity serve as a basis for the solution methods that will be presented in the next chapter.

4.1 An integrated approach vs. a hierarchical planning approach

In this section, we will briefly discuss two alternative approaches to tackle the three problems that were outlined in Sections 3.2–3.4. The first approach tries to simultaneously solve the three problems by formulating an integrated, monolithic optimization model. The second approach formulates separate optimization models, orders these models hierarchically, and solves one at a time. The separate models are only partially integrated. We will argue that the second approach is preferable for our problems and outline a three-stage hierarchical planning approach, which is partially integrated.

The first approach for our three problems, a monolithic model, integrates all decision variables into a single model. To integrate the three objectives of maximizing portfolio

M. Walter, *Multi-Project Management with a Multi-Skilled Workforce,* Produktion und Logistik, DOI 10.1007/978-3-658-08036-5_4, © Springer Fachmedien Wiesbaden 2015

benefit, minimizing average team size, and leveling working times into this single model, alternative roads can be selected (cf. Ehrgott, 2005; Neumann and Morlock, 2002, pp. 135–142; Domschke and Drexl, 2007, pp. 55-59). Two roads that are often selected are as follows. The first alternative associates a weight with each objective and considers the weighted sum of the objectives in a one-dimensional objective function. The second alternative considers a vector whose components are one-dimensional objective functions. For such a vector, a set of pareto-optimal solutions can be determined. The decision maker can select a solution out of this set of pareto-optimal solutions. For his selection, he must trade off the objectives against each other. For example, the decision maker could select the solution that offers the best leveled working times with an average team size of at most six workers per team and a portfolio benefit of at least 100.

In general, an integrated approach has two main disadvantages, which also become important in our case. First, an integrated approach tries to generate a detailed master plan and must, hence, process an enormous amount of data and information. In our case, a monolithic model would become very complex due to the high number of decision variables (cf. Günther, 1989, pp. 9–10). Second, a solution for the monolithic model fixes even those decision variables that lie in the distant future, although the situation in the distant future tends to be uncertain.

To overcome these two disadvantages of the first approach, the second approach structures the planning process hierarchically. The planning process is divided into several stages. The monolithic model is split into smaller problems, each problem is allocated to one stage of the planning process. The solution to a problem of a higher stage defines or constrains the solution space for problems at subordinated stages (cf. Schneeweiß, 1992, p. 13). An illustrative example of a hierarchical planning approach for a staffing problem is given by Grunow et al. (2004).

The hierarchy of planning problems often follows the importance of the corresponding decisions or the time horizon for which these decisions are made. The time horizon of a decision and its importance tend to be closely related: A long-term decision is usually a very important decision, i.e., a strategic decision, whereas short-term decisions, which affect only the near future, tend to have less impact on a firm. Accordingly, long-term or strategic planning, mid-term or tactical planning, and short-term or operational planning are distinguished.

Compared to an integrated approach, a hierarchical planning approach has advantages and drawbacks. On the one hand, a hierarchical planning approach is computationally better tractable and enables to postpone decisions of subordinate stages to times when uncertainty about data is resolved. On the other hand, a hierarchical approach requires the decision maker to rank the problems according to their importance before solutions to the problems will be generated. Here, the decision maker cannot trade off conflicting objectives as well as with the integrated approach. For instance, if the portfolio benefit is maximized first and the average team size is minimized afterwards, it may not be possible at the second stage to find a solution that offers an average team size of at most six workers per team.

For our three problems, however, a ranking of objectives in order of their importance stands out clearly. The decision about the project portfolio is the most important one for the firm, because selecting those projects which generate the highest total benefit is what matters most. For a given portfolio of projects the firm may wish an efficient

execution of projects. An efficient execution is facilitated by small project teams and by not scattering workers across a great number of projects. Therefore, minimizing the average size of project teams is the second most important objective. Finally, the aim of leveling hours worked by department members can be pursued when the other two goals have been achieved.

A ranking of objectives according to the time horizon of the corresponding decisions leads to the same order as a ranking in order of importance. The decision about the project portfolio affects the complete planning horizon $\{1, \ldots, T\}$ unless the set $\mathcal{P}^{\text{ongoing}} \cup \mathcal{P}^{\text{must}} \cup \tilde{\mathcal{P}}$ of projects can be partitioned into two non-empty subsets $\hat{\mathcal{P}}^1$ and $\hat{\mathcal{P}}^2$ such that there exists a period $t \in \mathcal{T} \setminus \{T\}$ with $t \geq t_p^{\text{finish}}$ for all $p \in \hat{\mathcal{P}}^1$ and $t < t_p^{\text{start}}$ for all $p \in \hat{\mathcal{P}}^2$ that divides the planning horizon into two separate planning horizons $\{1, \ldots, t\}$ and $\{t+1, \ldots, T\}$. The decision of assigning workers to projects affects also the complete planning horizon if the set $\mathcal{P}$ of projects cannot be partitioned into two sets $\mathcal{P}^1$ and $\mathcal{P}^2$, as just explained. Though, while the decision about the portfolio is almost irreversible, it is relatively easy and inexpensive to change the composition of the team for a project at later points in time, even after the start of the project. Hence, we can conclude that the time span that is affected by decisions about project teams is actually shorter than the time span that is affected by the decision about the project portfolio.

The time span that is affected by the problem of leveling hours worked by allocating departmental workload is shorter than the time spans affected by the decisions made for selection and team formation. The leveling problem must be solved for each department in each period $t \in \mathcal{T}$, hence, it is a short-term decision problem. At the time of project selection, the utilization of workers in a later period t is uncertain. Thus, it is reasonable to solve the leveling problem when utilization can be estimated more exactly, maybe two weeks before period t starts. It is not reasonable to solve this problem at the time when projects are selected, e.g., 10 months before period t starts.

In regard to the advantages of the hierarchical approach in general, and in regard to its suitability for our problems, we propose the following three-stage planning process. At the first stage, we solve the problem of selecting a project portfolio. At the second stage, we seek for the minimum number of assignments of workers to those projects that were selected at the first stage. Finally, we level the hours worked by department members in each period $t \in \mathcal{T}$ immediately before period t begins.

Although we have separated the solution process for our three problems into three stages, the problems are not fully separated but partially integrated. Here, partial integration means that we take into account subordinate problems when a problem of a superior stage is solved. When we solve the problem of project selection at the first stage, we take into account the availabilities of the workers who must accomplish the projects. We ensure that the requirements of the selected projects comply with the workers' availabilities. Furthermore, we take into account that sufficient availability remains for accomplishing departmental workloads. Also at the second stage, we take departmental workloads into account. For instance, at the time of project selection, the management of the firm may know that department d is busy with preparing a report every first month of a quarter and that department d' must prepare an exhibition appearance in period $t = 10$. These pieces of information can be integrated into the decision about the project portfolio.

In the subsequent sections we will present optimization models for the problem on each stage of our three-stage planning approach.

4.2 A model for the project selection problem

For a neat presentation of the model for project selection, we will introduce one further identifier. Recall that the set $\hat{\mathcal{P}}_k^{\text{suit}}$ is a subset of the union of the sets $\mathcal{P}^{\text{ongoing}}$, $\mathcal{P}^{\text{must}}$, and $\tilde{\mathcal{P}}$ and contains those projects p that are suitable for worker k. Let $\hat{\mathcal{P}}_k^{\text{suit}}(t) \subseteq \hat{\mathcal{P}}_k^{\text{suit}}$, $t \in \mathcal{T}$, denote the set of projects that are suitable for worker k and that are executed in period t. In other words, $\hat{\mathcal{P}}_k^{\text{suit}}(t)$ contains those projects p of the set $\hat{\mathcal{P}}_k^{\text{suit}}$ for which $t_p^{\text{start}} \leq t \leq t_p^{\text{finish}}$ holds.

Now, our problem of selecting projects can be modeled by (4.1)–(4.7). Model (4.1)–(4.7) is a mixed-integer linear programming (MIP) model with binary decision variables z_p, $p \in \tilde{\mathcal{P}}$, that indicate whether project p is selected or not, and with non-negative continuous decision variables $\hat{y}_{kpst}$, $k \in \mathcal{K}$, $p \in \hat{\mathcal{P}}_k^{\text{suit}}$, $s \in \mathcal{S}_{kp}^{\text{match}}$, $t \in \mathcal{T}_p$, that represent the workload that worker k performs for project p and skill s in period t.

$$\text{Max.} \quad \sum_{p \in \tilde{\mathcal{P}}} b_p z_p \tag{4.1}$$

$$\text{s. t.} \quad \sum_{k \in \mathcal{K}_s} (l_{ks} \hat{y}_{kpst}) = r_{pst} z_p \qquad p \in \mathcal{P}^{\text{ongoing}} \cup \mathcal{P}^{\text{must}} \cup \tilde{\mathcal{P}},\ s \in \mathcal{S}_p,\ t \in \mathcal{T}_p \tag{4.2}$$

$$\sum_{p \in \hat{\mathcal{P}}_k^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{kp}^{\text{match}}} \hat{y}_{kpst} \leq R_{kt} \qquad k \in \mathcal{K},\ t \in \mathcal{T} \tag{4.3}$$

$$\sum_{k \in \mathcal{K}_d} \left(R_{kt} - \sum_{p \in \hat{\mathcal{P}}_k^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{kp}^{\text{match}}} \hat{y}_{kpst} \right) \geq r d_{dt} \qquad d \in \mathcal{D},\ t \in \mathcal{T} \tag{4.4}$$

$$z_p = 1 \qquad p \in \mathcal{P}^{\text{ongoing}} \cup \mathcal{P}^{\text{must}} \tag{4.5}$$

$$z_p \in \{0, 1\} \qquad p \in \tilde{\mathcal{P}} \tag{4.6}$$

$$\hat{y}_{kpst} \geq 0 \qquad k \in \mathcal{K},\ p \in \hat{\mathcal{P}}_k^{\text{suit}},\ s \in \mathcal{S}_{kp}^{\text{match}},\ t \in \mathcal{T}_p \tag{4.7}$$

Objective function (4.1) maximizes the benefit of the project portfolio. Constraint set (4.2) ensures that each requirement r_{pst} of project p is satisfied if project p is selected. The requirement r_{pst} for skill s in period t is satisfied by contributions of workers who master skill s. The coefficient l_{ks} takes into account that the workers $k \in \mathcal{K}_s$ master skill s at different levels.

Constraint set (4.3) guarantees that a worker k does not spend more time for projects in a period t than the time he is available in this period. Constraint set (4.4) assures that the workers of every department have enough remaining availability to accomplish the departmental workload in every period. On the left-hand side of Constraint (4.4), we calculate for each worker $k \in \mathcal{K}_d$ of department d the time that remains of his initial availability R_{kt} in period t when contributions to projects are considered. The total remaining time of all workers of department d must be large enough to cover the departmental workload.

Constraint set (4.5) fixes the variables z_p, $p \in \mathcal{P}^{\text{ongoing}} \cup \mathcal{P}^{\text{must}}$, to 1, because we have to

include these projects in the portfolio. Constraint sets (4.6) and (4.7) state the domains of the actual decision variables.

By variables $\hat{y}_{kpst}$ in conjunction with Constraints (4.2)–(4.4), we model the allocation of workload and the availabilities for each worker explicitly. This is necessary because we consider cases where at least two workers master more than one skill each and where for at least one of these skills different levels are distinguished. Otherwise, i.e., especially if only homogeneous skill levels were considered, it would be possible to aggregate the capacity of the workers and to spare all variables $\hat{y}_{kpst}$. Though, such an aggregation would lead to a model of exponential size in the number of skills S (cf. Grunow et al., 2004, Section 3.3).

Grunow et al. (2004, Section 3.3) and Grunow et al. (2002, Section 4.2) show how capacities of multi-skilled resources can be aggregated if skill levels are homogenous. In the instances of their problems, the number of skills must have been relatively small so that model size did not became critical. We will give an example that shows how their aggregation could be applied in our case if skill levels were not differentiated. In this example, we also show why this aggregation is not applicable in the case of heterogeneous skill levels.

Example 4.1 Assume an instance A with $K = S = 2$, $\mathcal{T} = \{t\}$ and several projects that can be selected. Let $\mathcal{S}_{k_1} = \{s_1\}$ and $\mathcal{S}_{k_2} = \{s_1, s_2\}$ with $l_{ks} = 1$, $k \in \mathcal{K}$, $s \in \mathcal{S}_k$, and let $R_{kt} = 10$, $k \in \mathcal{K}$. Furthermore, assume that there is no departmental workload at all.[1] Let us denote the demand of the projects for skill $s \in \mathcal{S}$ that must be satisfied in period t by r_{st} where $r_{st} := \sum_{p \in \mathcal{P}^{\text{ongoing}} \cup \mathcal{P}^{\text{must}} \cup \tilde{\mathcal{P}} \mid t \in \mathcal{T}_p} r_{pst} z_p$. For instance A, Constraint sets (4.2) and (4.3) can then be replaced by the following constraints, which ensure that the skill requirements of all selected projects are satisfied and that the availabilities of all workers are regarded: $r_{s_1 t} \leq 20$, $r_{s_2 t} \leq 10$, and $r_{s_1 t} + r_{s_2 t} \leq 20$. In general, we would require that the following Constraint set must hold, where $\mathcal{S}'$ represents any non-empty subset of $\mathcal{S}$:

$$\sum_{s \in \mathcal{S}'} r_{st} \leq \sum_{k \mid \mathcal{S}_k \cap \mathcal{S}' \neq \emptyset} R_{kt} \qquad \mathcal{S}' \subseteq \mathcal{S},\, \mathcal{S}' \neq \emptyset,\, t \in \mathcal{T} \tag{4.8}$$

Since the number of non-empty subsets of $\mathcal{S}$ is equal to $2^{|\mathcal{S}|} - 1$, Constraint set (4.8) comprises an exponential number of constraints what makes this aggregation unattractive when there is a large number of skills.

Now, let us turn to the case of heterogeneous skill levels. Consider an instance B which is identical to instance A except that now $\mathcal{S}_{k_1} = \mathcal{S}_{k_2} = \{s_1, s_2\}$ with $l_{k_1 s_1} = 0.5$ and $l_{k_1 s_2} = l_{k_2 s_1} = l_{k_2 s_2} = 1$ holds. Here, the capacity of worker k_1 depends on the skill that she performs. We will consider two possibilities to adjust Constraints (4.8) to this situation. As we will see, both possibilities do not work. One possibility is to replace the right-hand side of Constraint set (4.8) by $\sum_{k \mid \mathcal{S}_k \cap \mathcal{S}' \neq \emptyset} (R_{kt} \cdot \max_{s \in \mathcal{S}_k \cap \mathcal{S}'} l_{ks})$. Then, we obtain the constraints $r_{s_1 t} \leq 15$, $r_{s_2 t} \leq 20$, and $r_{s_1 t} + r_{s_2 t} \leq 20$. According to these constraints, a project portfolio with $r_{s_1 t} = 15$ and $r_{s_2 t} = 1$ is a feasible solution of the selection problem, but there exists no corresponding feasible solution for the variables $\hat{y}_{kpst}$, i.e., there exists no feasible disaggregation. A second possibility is to replace the right-hand side of Constraint set (4.8) by $\sum_{k \mid \mathcal{S}_k \cap \mathcal{S}' \neq \emptyset} (R_{kt} \cdot \min_{s \in \mathcal{S}_k \cap \mathcal{S}'} l_{ks})$. This replacement yields the constraints $r_{s_1 t} \leq 15$, $r_{s_2 t} \leq 20$, and $r_{s_1 t} + r_{s_2 t} \leq 15$. These constraints render the

[1] This assumption is no loss of generality because we could interpret the requirements of each department as a project, as explained in Subsection 4.3.2 on page 64.

solution $r_{s_1t} = r_{s_2t} = 10$ infeasible, although there exists a disaggregation leading to feasible values for the variables $\hat{y}_{kpst}$. □

As Example 4.1 indicates, capacities of workers cannot be aggregated when skill levels are heterogeneous because the capacity of a worker depends on the skill or, to be more precise, on the skill mix which he performs. However, this skill mix is not known in advance, i.e., not before the model is solved. Hence, we must explicitly model the allocation of workload and the availabilities for each worker.

4.3 Models for the workforce assignment problem and their limitations

In this section, we consider models for the problem of assigning workers to projects and allocating project workload to workers. In Subsection 4.3.1, we will present two MIP models for this problem. The two models are alternative fomulations of the workforce assignment problem. Since this problem is in the focus of this thesis, we point out limitations of the models in Subsection 4.3.2 and outline potential extensions that mitigate these limitations.

4.3.1 Two alternative models for the workforce assignment problem

In this subsection, we will present two MIP models for the problem of assigning workers to projects and allocating project workload to workers. The first model is termed *standard model*, it can be intuitively derived from the problem definition in Section 3.3. The second model is named *network model*, because it implies for each period $t \in \mathcal{T}$ a network flow model. In such a network flow model, working time is assumed to flow from workers to projects and departments in order to cover their requirements.

To obtain a sparse formulation of both MIP models, we introduce one further identifier, analogously to the previous section. Remember that the set $\mathcal{P}_k^{\text{suit}} \subseteq \mathcal{P}$ includes those projects p that are suitable for worker k due to matching skills. Let $\mathcal{P}_k^{\text{suit}}(t) \subseteq \mathcal{P}_k^{\text{suit}}$, $t \in \mathcal{T}$, denote the set of projects that are suitable for worker k and that are carried out in period t. Put another way, $\mathcal{P}_k^{\text{suit}}(t)$ contains those projects p of the set $\mathcal{P}_k^{\text{suit}}$ for which $t_p^{\text{start}} \leq t \leq t_p^{\text{finish}}$ holds.

The standard model is given by (4.9)–(4.16). The decision variables of the standard model are the binary variables x_{kp}, $k \in \mathcal{K}$, $p \in \mathcal{P}_k^{\text{suit}}$, which indicate whether worker k is assigned to project p or not, and the non-negative continuous variables y_{kpst}, $k \in \mathcal{K}$, $p \in \mathcal{P}_k^{\text{suit}}$, $s \in \mathcal{S}_{kp}^{\text{match}}$, $t \in \mathcal{T}_p$, which record the workload that worker k accomplishes for project p and skill s in period t.

$$\text{Min.} \quad \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k^{\text{suit}}} x_{kp} \tag{4.9}$$

$$\text{s. t.} \quad \sum_{k \in \mathcal{K}_s} (l_{ks} y_{kpst}) = r_{pst} \qquad p \in \mathcal{P},\, s \in \mathcal{S}_p,\, t \in \mathcal{T}_p \tag{4.10}$$

$$\sum_{s \in \mathcal{S}_{kp}^{\text{match}}} y_{kpst} \leq R_{kt} x_{kp} \qquad p \in \mathcal{P},\, k \in \mathcal{K}_p^{\text{suit}} \setminus \mathcal{K}_p^{\text{assigned}},\, t \in \mathcal{T}_p \tag{4.11}$$

$$\sum_{p \in \mathcal{P}_k^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{kp}^{\text{match}}} y_{kpst} \leq R_{kt} \qquad k \in \mathcal{K},\, t \in \mathcal{T} \tag{4.12}$$

$$\sum_{k \in \mathcal{K}_d} \left(R_{kt} - \sum_{p \in \mathcal{P}_k^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{kp}^{\text{match}}} y_{kpst} \right) \geq rd_{dt} \qquad d \in \mathcal{D},\, t \in \mathcal{T} \tag{4.13}$$

$$x_{kp} = 1 \qquad p \in \mathcal{P}^{\text{ongoing}},\, k \in \mathcal{K}_p^{\text{assigned}} \tag{4.14}$$

$$x_{kp} \in \{0,1\} \qquad p \in \mathcal{P},\, k \in \mathcal{K}_p^{\text{suit}} \setminus \mathcal{K}_p^{\text{assigned}} \tag{4.15}$$

$$y_{kpst} \geq 0 \qquad k \in \mathcal{K},\, p \in \mathcal{P}_k^{\text{suit}},\, s \in \mathcal{S}_{kp}^{\text{match}},\, t \in \mathcal{T}_p \tag{4.16}$$

Objective function (4.9) minimizes the total number of assignments of workers to projects and, thus, the average team size. The total number of assignments includes the assignments that have already been made for ongoing projects. Constraints (4.10) ensure that the requirement r_{pst}, $p \in \mathcal{P}$, $s \in \mathcal{S}_p$, $t \in \mathcal{T}_p$, is satisfied by contributions from workers who master skill s. The coefficient l_{ks} takes into account that these workers $k \in \mathcal{K}_s$ master skill s at different levels.

Constraints (4.11) link the variables x_{kp} and y_{kpst}. These constraints guarantee that worker k can only contribute to project p if he is assigned to project p. A contribution $y_{kpst} > 0$ for any skill $s \in \mathcal{S}_{kp}^{\text{match}}$ in any period $t \in \mathcal{T}_p$ requires $x_{kp} = 1$. Simultaneously, Constraints (4.11) force $x_{kp} = 1$ if worker k contributes to project p for any skill $s \in \mathcal{S}_{kp}^{\text{match}}$ in any period $t \in \mathcal{T}_p$. Hence, if worker k contributes to project p, he is automatically assigned to project p.

Constraints (4.11) are so called "big-M constraints", which allow to model logical conditions (cf. Bosch and Trick, 2005, pp. 77–78; Williams, 1999, pp. 154–160). In a general form, the right-hand side of Constraints (4.11) would be written as Mx_{kp}, where M is a sufficiently large constant. We chose R_{kt} for M, as R_{kt} is an upper bound for $\sum_{s \in \mathcal{S}_{kp}^{\text{match}}} y_{kpst}$ and, hence, sufficiently large.

Constraints (4.12) take care that the working time which worker k spends for projects in period t does not exceed his availability R_{kt}. Constraints (4.13) ensure for each department $d \in \mathcal{D}$ that the remaining availabilities of all workers of department d are large enough in every period $t \in \mathcal{T}$ to accomplish the departmental workload rd_{dt}.

For each ongoing project $p \in \mathcal{P}^{\text{ongoing}}$, a team $\mathcal{K}_p^{\text{assigned}}$ of workers exists already. For each member k of this team, Constraint set (4.14) fixes the corresponding variable x_{kp} to 1. Constraint sets (4.15) and (4.16) state the domains of the actual decision variables.

Let us briefly discuss the big-M constraints (4.11). We could have replaced Constraints (4.11) by

$$y_{kpst} \leq R_{kt} x_{kp} \qquad p \in \mathcal{P},\, k \in \mathcal{K}_p^{\text{suit}} \setminus \mathcal{K}_p^{\text{assigned}},\, s \in \mathcal{S}_{kp}^{\text{match}},\, t \in \mathcal{T}_p \tag{4.17}$$

or by

$$\sum_{s \in \mathcal{S}_{kp}^{\text{match}}} \sum_{t \in \mathcal{T}_p} y_{kpst} \leq \left(\sum_{t \in \mathcal{T}_p} R_{kt} \right) x_{kp} \qquad p \in \mathcal{P},\, k \in \mathcal{K}_p^{\text{suit}} \setminus \mathcal{K}_p^{\text{assigned}}. \tag{4.18}$$

To assess the three alternatives (4.11), (4.17), and (4.18), we scrutinize the number of constraints resulting from each constraint set and the tightness of each constraint set for the linear programming (LP) relaxation of the corresponding MIP. The LP relaxation of the standard model is obtained when the binary variables $x_{kp} \in \{0,1\}$ are replaced by continuous variables $x_{kp} \in [0,1]$. This relaxation is commonly used to solve the model by branch-and-bound or branch-and-cut methods. The higher the number of constraints which define the feasible region of the LP relaxation, the more time is generally required for solving the LP relaxation. The tighter the relaxation, the closer does the feasible region of the relaxation come to the convex hull of feasible integer points of the MIP. The tightness of the relaxation is vitally important and more important than the number of constraints. In general, the drawback of additional constraints is outweighed by far if these constraints tighten the relaxation (cf. Williams, 1999, pp. 190–197).

The largest number of constraints exhibits Constraint set (4.17). It comprises $\overline{|\mathcal{S}^{\text{match}}|}$ times as much constraints as Constraint set (4.11), where $\overline{|\mathcal{S}^{\text{match}}|}$ denotes the average number of matching skills between a project $p \in \mathcal{P}$ and a worker $k \in \mathcal{K}_p^{\text{suit}} \setminus \mathcal{K}_p^{\text{assigned}}$. The smallest number of constraints are contained in Constraint set (4.18), where a constraint for project p covers the whole duration of project p. Constraint set (4.11) has $\overline{|\mathcal{T}^{\text{proj}}|}$ times as much constraints as Constraint set (4.18), where $\overline{|\mathcal{T}^{\text{proj}}|}$ denotes the average duration of a project $p \in \mathcal{P}$.

We will use the following example to gain insight in the tightness of the three constraint sets:

Example 4.2 Consider worker k and project p with $\mathcal{S}_{kp}^{\text{match}} = \{s_1, s_2\}$, $\mathcal{T}_p = \{t_1, t_2\}$, $R_{kt_1} = 50$, and $R_{kt_2} = 100$. Let $y_{kps_1t_1} = y_{kps_2t_1} = 25$ and $y_{kps_1t_2} = y_{kps_2t_2} = 0$ be a feasible solution for the MIP. Then $x_{kp} \geq 1$ satisfies (4.11), $x_{kp} \geq 0.5$ satisfies (4.17) and $x_{kp} \geq \frac{1}{3}$ satisfies (4.18) in the LP relaxation of the corresponding MIP. □

In Example 4.2, Constraint set (4.11) is the tightest out of the three alternatives. Indeed, Constraint set (4.11) is always at least as tight as Constraint sets (4.17) and (4.18) and tighter than (4.17) and (4.18) in general.

From our considerations we concluded that Constraint set (4.11) is the best choice. Numerical tests supported our conclusion. These numerical tests are presented in Section 7.3.[2]

The network model is an alternative way to represent the same problem as the standard model. Since the properties of network structures can often be exploited to design efficient solution methods, it seems worthwhile to pursue this alternative approach. The network model regards in each period $t \in \mathcal{T}$ each worker $k \in \mathcal{K}$ as a source of working time. This source is represented by a node in the network model that supplies an amount of R_{kt} hours of working time. Projects and departments are regarded as sinks, i.e., as nodes that ask for working time. Project p asks for r_{pst} hours of working time for each skill $s \in \mathcal{S}_p$ in period t. Department d demands rd_{dt} hours of working time in period t. In the network of period t, working time can flow from source nodes along arcs via intermediate nodes to the sinks which represent the demand nodes. In any period $t \in \mathcal{T}_p$, working time can flow from worker $k \in \mathcal{K}_p^{\text{suit}}$ to project p only if worker k is assigned to project p. Our aim is to determine the minimum number of assignments of workers to projects that allow a flow of working time which satisfies all demands of projects and departments in every period.

[2]For a tighter formulation of the big-M Constraints (4.11) and (4.17) see Subsection 6.1.1.

The underlying network of a period is sketched in Figure 4.1, which also depicts additional flow variables, which are required for the network model. For Figure 4.1, we assume that projects p_1 and p_2 are executed in period t and that workers k_1 and k_2 belong to department d_1. Furthermore, we assume that worker k_1 and k_2 master the skills s_1 and s_2, which are required by project p_1. Note that Figure 4.1 illustrates only a section of the total network of period t in order to clarify the concept. A demand for working time is represented by a negative supply.

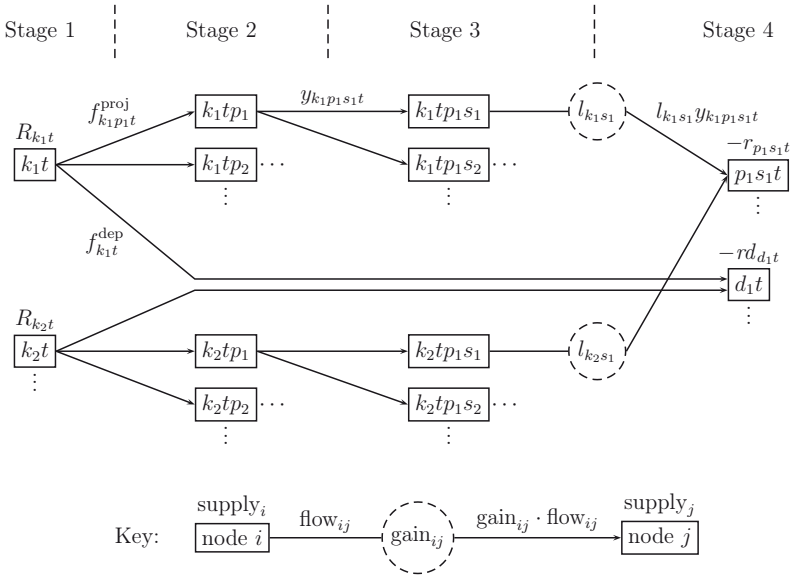

Figure 4.1: Section of the underlying network flow model for period t

Like the network of period t in Figure 4.1, the network of each period $t \in \mathcal{T}$ comprises four stages, with supply nodes at the first stage and demand nodes at the fourth stage. At the first stage of the network, the source nodes represent the workers $k \in \mathcal{K}$, who supply a flow of R_{kt} hours of working time. This flow from worker k is divided into flows f^{proj}_{kpt} to projects $p \in \mathcal{P}^{\text{suit}}_k(t)$ at the second stage and into a flow f^{dep}_{kt} to the department to which worker k belongs.

The flow f^{proj}_{kpt} to project p is split up into flows y_{kpst}, $s \in \mathcal{S}^{\text{match}}_{kp}$, to project demand nodes on the final stage. Before the flow y_{kpst} reaches the sink node that represents the project requirement r_{pst}, the flow is multiplied by a gain of l_{ks}. This gain weights the time that worker k spends for skill s of project p with his skill level l_{ks}. Since $l_{ks} \neq 1$ is possible, we obtain a network with gains (cf. Ahuja et al., 1993, pp. 566–568; Bertsekas, 1998, pp. 360–365). Problems on networks with gains are termed *generalized network problems*.

The network model requires two types of additional variables, which have already been introduced in Figure 4.1. First, variable $f^{\text{proj}}_{kpt} \in \mathbb{R}_{\geq 0}$, $k \in \mathcal{K}$, $p \in \mathcal{P}^{\text{suit}}_k$, $t \in \mathcal{T}_p$, records the flow between stage 1 and 2 from worker k to project p in period t. Second, the variable $f^{\text{dep}}_{kt} \in \mathbb{R}_{\geq 0}$, $k \in \mathcal{K}$, $t \in \mathcal{T}$, represents the flow between stage 1 and 4 from worker k to his department d in period t.

For each worker $k \in \mathcal{K}$ and all projects $p \in \mathcal{P}^{\text{suit}}_k$, the flow variables f^{proj}_{kpt} of all periods $t \in \mathcal{T}_p$ and hence the networks of all these periods are coupled by the binary decision variable x_{kp}. Each variable f^{proj}_{kpt}, $t \in \mathcal{T}_p$, must equal 0 if worker k is not assigned to project p, i.e., if $x_{kp} = 0$.

Although additional variables are required for the network model, it is worthwhile to consider this model, because underlying network structures can often be exploited by specialized network algorithms. These algorithms facilitate an efficient solution of the underlying problems (cf. Ahuja et al., 1993, pp. 402–403, for example).

The network model is given by (4.19)–(4.29). The decision variables of the network model are the binary variables x_{kp}, $k \in \mathcal{K}$, $p \in \mathcal{P}^{\text{suit}}_k$, which indicate whether worker k is assigned to project p or not, and the non-negative continuous variables y_{kpst}, $k \in \mathcal{K}$, $p \in \mathcal{P}^{\text{suit}}_k$, $s \in \mathcal{S}^{\text{match}}_{kp}$, $t \in \mathcal{T}_p$, which record the workload that worker k performs for project p and skill s in period t. Auxiliary variables are the non-negative continuous variables f^{proj}_{kpt}, $k \in \mathcal{K}$, $p \in \mathcal{P}^{\text{suit}}_k$, $t \in \mathcal{T}_p$, which represent the flow from worker k to project p in period t, and the non-negative continuous variables f^{dep}_{kt}, $k \in \mathcal{K}$, $t \in \mathcal{T}$, which represent the flow from worker k to his department in period t.

$$\text{Min.} \quad \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}^{\text{suit}}_k} x_{kp} \tag{4.19}$$

$$\text{s. t.} \quad R_{kt} = f^{\text{dep}}_{kt} + \sum_{p \in \mathcal{P}^{\text{suit}}_k(t)} f^{\text{proj}}_{kpt} \qquad k \in \mathcal{K},\, t \in \mathcal{T} \tag{4.20}$$

$$f^{\text{proj}}_{kpt} = \sum_{s \in \mathcal{S}^{\text{match}}_{kp}} y_{kpst} \qquad k \in \mathcal{K},\, p \in \mathcal{P}^{\text{suit}}_k,\, t \in \mathcal{T}_p \tag{4.21}$$

$$\sum_{k \in \mathcal{K}_s} (l_{ks} y_{kpst}) = r_{pst} \qquad p \in \mathcal{P},\, s \in \mathcal{S}_p,\, t \in \mathcal{T}_p \tag{4.22}$$

$$\sum_{k \in \mathcal{K}_d} f^{\text{dep}}_{kt} \geq r d_{dt} \qquad d \in \mathcal{D},\, t \in \mathcal{T} \tag{4.23}$$

$$f^{\text{proj}}_{kpt} \leq R_{kt} x_{kp} \qquad p \in \mathcal{P},\, k \in \mathcal{K}^{\text{suit}}_p \setminus \mathcal{K}^{\text{assigned}}_p,\, t \in \mathcal{T}_p \tag{4.24}$$

$$x_{kp} = 1 \qquad p \in \mathcal{P}^{\text{ongoing}},\, k \in \mathcal{K}^{\text{assigned}}_p \tag{4.25}$$

$$x_{kp} \in \{0, 1\} \qquad p \in \mathcal{P},\, k \in \mathcal{K}^{\text{suit}}_p \setminus \mathcal{K}^{\text{assigned}}_p \tag{4.26}$$

$$f^{\text{proj}}_{kpt} \geq 0 \qquad k \in \mathcal{K},\, p \in \mathcal{P}^{\text{suit}}_k,\, t \in \mathcal{T}_p \tag{4.27}$$

$$f^{\text{dep}}_{kt} \geq 0 \qquad k \in \mathcal{K},\, t \in \mathcal{T} \tag{4.28}$$

$$y_{kpst} \geq 0 \qquad k \in \mathcal{K},\, p \in \mathcal{P}^{\text{suit}}_k,\, s \in \mathcal{S}^{\text{match}}_{kp},\, t \in \mathcal{T}_p \tag{4.29}$$

Objective function (4.19) minimizes the total number of assignments and, hence, the average team size. The total number of assignments includes the assignments that have already been made for ongoing projects.

Constraints (4.20)–(4.22) are flow conservation constraints. Constraints (4.20) demand that the flow of working time from worker k to his department and to projects in period t must equal the supply R_{kt} of working time in period t. Constraints (4.21) ensure that the working time which worker k spends for all skills $s \in \mathcal{S}_{kp}^{\text{match}}$ of project p in period t equals the flow from worker k to project p in period t. Constraints (4.22) assure that requirement r_{pst} of project p for working time concerning skill s in period t is covered by contributions from suitable workers k. Their contributions are weighted with their skill levels l_{ks}.

Constraints (4.23) guarantee that the flow of working time from those workers who belong to department $d \in \mathcal{D}$ to their department d in period t is sufficiently large to cover the requirement rd_{dt}. Constraint set (4.23) allows that an excessive supply of working time is absorbed by the departments. As a consequence, the variable f_{kt}^{dep} does not specify the departmental workload that worker k must accomplish in period t, but is only an upper bound on the departmental workload that must be accomplished by worker k in period t.

Constraints (4.24) link the variables x_{kp} and f_{kpt}^{proj}. These constraints guarantee that worker k can only contribute to project p if he is assigned to project p. A contribution $f_{kpt}^{\text{proj}} > 0$ in any period $t \in \mathcal{T}_p$ and, hence, a contribution $y_{kpst} > 0$ for any skill $s \in \mathcal{S}_{kp}^{\text{match}}$ requires $x_{kp} = 1$. Simultaneously, Constraints (4.24) force $x_{kp} = 1$ if worker k contributes to project p in any period $t \in \mathcal{T}_p$. Thus, if worker k contributes to project p, he is automatically assigned to project p.

For each ongoing project $p \in \mathcal{P}^{\text{ongoing}}$, a team $\mathcal{K}_p^{\text{assigned}}$ of workers exists already. For each member k of this team, Constraint set (4.25) fixes the corresponding variable x_{kp} to 1. Constraint sets (4.26)–(4.29) state the domains of the actual decision variables. The decision variables whose domains are defined in (4.27) and (4.28) can be considered as auxiliary variables, which are required to model the network flows in each period $t \in \mathcal{T}$.

Constraints (4.24) are big-M constraints. They are equivalent to the big-M Constraints (4.11) of the standard model. Constraint set (4.24) can also be replaced by Constraint sets (4.17) or (4.18).

4.3.2 Limitations of the assignment models and potential remedies and extensions

We have already discussed limitations of our approach to project selection in Section 3.2 and limitations of our approach to leveling hours worked by allocating departmental workload in Section 3.4. Now, we will elaborate on limitations of the two models that we introduced in the previous Subsection 4.3.1. Both models have identical scope and seek for a solution to the same problem. They search for an assignment of workers to projects and for an allocation of project workload to workers. Since this problem is in the focus of this thesis, we dedicate a separate subsection to limitations of our models for this problem. The limitations that we discuss are the neglect of overtime, the neglect of learning effects[3], the neglect of the role of project managers, and the neglect of worker compatibility.

In many firms, workers are allowed to work overtime. If a worker works overtime, his regular availability is exceeded. Usually, the amount of extra hours per period is

[3]The consideration of learning effects would result in dynamic skill levels.

limited. Firms prescribe extra hours to meet peak demands. Workers are compensated for extra hours either by monetary means or by days off in other periods. If days off are granted instead of monetary rewards, many employment contracts prescribe that the average hours worked per period must not exceed the average regular availability per period. This means that extra hours must be completely compensated in the course of a year or so.

Our models (4.9)–(4.16) and (4.19)–(4.29) do not consider overtime. They regard the regular availability R_{kt} of worker k in period t as a hard constraint, which must not be violated. Hence, our modeling approach does not meet the conditions that prevail in many firms.

In the project management literature, overtime is rarely considered in models. A model that takes overtime into account is the model of Heimerl and Kolisch (2010a). Heimerl and Kolisch (2010a) want to minimize costs for wages. They associate wage rates with extra hours that are higher than wage rates for regular hours. In their model, extra hours are explicitly registered by distinct variables.

The integration of overtime into our models is possible as well. If extra hours are compensated by additional payments and need not be balanced over the planning horizon but are limited for each period, then we would just have to increase the availability R_{kt}. For example, assume that the regular availability of worker k in period t is given by $R_{kt} = 160$ and that 20 extra hours are allowed for worker k in period t, then we would set $R_{kt} := 180$. Since we do not consider variable costs, nothing else must be done. If we want to take variable costs for overtime into account, additional variables are necessary to record the extra hours of each worker in each period (cf. Heimerl and Kolisch, 2010a).

If extra hours must be compensated by days off in the course of the planning horizon, our models require three changes. First, we have to increase R_{kt}, $k \in \mathcal{K}$, $t \in \mathcal{T}$, by the number of extra hours that are allowed per period. For example, if the regular availability of worker k is given by $R_{kt} = 160$ for each period $t \in \mathcal{T}$ and if 20 extra hours were allowed per period, we would set $R_{kt} := 180$ for each period $t \in \mathcal{T}$.

Second, a new set of constraints must be added to our models. These constraints must demand that the number of hours worked by worker $k \in \mathcal{K}$ during the whole planning horizon does not exceed the number of regular hours that the labor contract allows for the planning horizon. In our example, worker k would not be allowed to work more than $160 \cdot T$ hours during the planning horizon $\mathcal{T} = \{1, \ldots, T\}$. Constraint (4.30) imposes this limit on the total working time of worker k during the planning horizon.

$$\sum_{t \in \mathcal{T}} \sum_{p \in \mathcal{P}_k^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{kp}^{\text{match}}} y_{kpst} \leq 160 \cdot T \tag{4.30}$$

Third, the workload of each department must be interpreted as a project. The first two changes, which we presented for the standard model and the network model, are not in line with Constraints (4.13) and Constraints (4.23), respectively, which ensure that the remaining availability after project work is sufficiently large to cover departmental workloads in each period. The issue is that Constraint (4.30) does not register the time that worker k spends for departmental work. To fix this issue, the workload of each department $d \in \mathcal{D}$ has to be interpreted as a project. This interpretation results in D additional projects. The project that corresponds with department d has a requirement of rd_{dt} man-hours in period $t \in \mathcal{T}$. This workload can only be accomplished by members of

department d. Thus, department membership must be interpreted as a skill, resulting in D additional skills. Such an additional skill is mastered only by the members of the corresponding department. The skill level with which each member masters his departmental skill is equal to 1. Additional binary assignment variables that indicate whether worker k is assigned to the project that corresponds to his department or not, can be fixed to 1. These three changes of our models would allow to integrate overtime that is compensated by days off.

A consideration of overtime in a model raises the question how worked extra hours should be taken into account in the objective function of the model. If a firm grants monetary compensation for extra hours, it seems natural to minimize the payments for overtime. Though it is difficult to integrate this cost objective and the objective of a minimum average team size into a single objective function, because both objectives are expressed in different units. Natural weights for the two objectives are not apparent. Hence, it would make more sense to consider both objectives separately and to determine a set of pareto-optimal solutions. If the firm compensates extra hours by days off, the same difficulties arise if the firm wants to minimize the number of extra hours in order to achieve leveled working times for its workforce. If the firm does not aim at minimizing extra hours, extra hours need not be taken into account in an objective function.

We do not consider overtime in our models for two reasons. First, extra hours are rather an ad hoc measure to cover unforeseen workload peaks. Although flexible working time agreements have become more common, especially labor unions and works councils urge the management of firms to stick to regular working times and are opposed to the planned use of extra hours. Therefore, overtime is often not deemed a suitable way to expand capacity or to form small teams. The second reason for neglecting overtime is owed the novelty of our approach. Since this approach is the first of its kind, we restrict our models to the most essential elements and parts of the underlying problem in order to get good insight into basic properties of the problem. Effects that appear when input data or solution methods are changed stand out more clearly in case of a plain and compact model.

Furthermore, our models assume that the skill levels l_{ks}, $k \in \mathcal{K}$, $s \in \mathcal{S}_k$, are static. We ignore that skill levels might change due to effects of learning and forgetting. Learning and forgetting a skill is closely linked to performing the skill. When skill s is performed by worker k, the learning effect increases the skill level l_{ks}. The increase can be derived from a nonlinear learning curve (cf. Wright, 1936; Yelle, 1979). Forgetting appears when skill s is not performed for some time and decreases the skill level (cf. Chen and Edgington, 2005, p. 287).

In the literature, models exist that incorporate learning and forgetting (cf. Wu and Sun, 2006; Gutjahr et al., 2008, 2010; and Heimerl and Kolisch, 2010b, for example). These models apply nonlinear expressions to integrate the concept of the learning curve. The resulting models feature dynamic skill levels. Some of the models aim at allocating workload such that targets for skill levels are met at the end of the planning horizon.

The integration of dynamic skill levels into our models would require significant changes, which lead to nonlinear models. Though, in our opinion, the merit of considering dynamic skills is small in our case. We argue that the merit is small for two reasons. First, it is difficult and cumbersome to derive learning rates for each worker and each skill, especially because for some skills it can be costly to measure skill levels

at short intervals to construct a learning curve. Estimations of learning rates tend to be error-prone. Secondly, the typical length of our planning horizon of one year is relatively short compared to the duration of project tasks, which are quite complex and can last several months. Thus, learning effects within some periods should be rather small, as the number of units of output is small.[4] That is why we recommend to use static skill levels and evaluate skill levels once a year. Then, models for the next planning horizon can be fed with updated skill levels. Periodically updates of skill levels are also used by Süer and Tummaluri (2008).

If a firm aims at allocating project workload such that skills of workers are developed to meet skill level targets at the end of the planning horizon, a model featuring dynamic skill levels is advantageous. Nevertheless, if the firm wants that worker k gathers experience in skill s, we could simply add a constraint to our models. Assume that worker k is said to perform at least 100 hours of work for skill s during the planning horizon, then Constraint (4.31) would guarantee this experience.

$$\sum_{t \in \mathcal{T}} \sum_{p \in \mathcal{P}_k^{\text{suit}}(t) \,|\, s \in \mathcal{S}_p} y_{kpst} \geq 100 \tag{4.31}$$

The third limitation concerns the role of the project manager. Usually, each project has one project manager. She is the head of the project team, coordinates the team members, and is responsible for a successful implementation of the project (Kerzner, 2013, pp. 14–15).

While our models do not take the role of project managers into account, Yoshimura et al. (2006) and Patanakul et al. (2007) published models that explicitly consider the role of the project manager.

For our approach, we can imagine at least two ways to assign exactly one project manager to each project. First, project managers might be assigned in advance, i.e., before the assignment model is solved that determines the project teams. Second, the task of assigning one project manager to each project could be integrated into our assignment models. We would define a set $\mathcal{K}_p^{\text{PM}} \subseteq \mathcal{K}_p^{\text{suit}}$, $p \in \mathcal{P}$, of workers that are qualified to act as project manager for project p. Then, Constraint (4.32) would ensure that at least one worker out of the set $\mathcal{K}_p^{\text{PM}}$ is selected for managing project p.

$$\sum_{k \in \mathcal{K}_p^{\text{PM}}} x_{kp} \geq 1 \tag{4.32}$$

If in a solution more than one worker out of the set $\mathcal{K}_p^{\text{PM}}$ was assigned to project p, one of them must be selected as project manager. If it is not deemed suitable to include more than one potential project manager into a project team, we could demand that the left-hand side of Constraint (4.32) must equal 1.

Project managers have to accomplish special tasks for their project, e.g., administrative tasks. To model these requirements we could associate a distinct skill s with the

[4] It may even be difficult to define an appropriate unit of output. Though, a definition of a unit of output is required to measure a learning rate. Such a definition is obvious for manufacturing firms (cf. Nembhard and Uzumeri, 2000). For a software development organization, Boh et al. (2007) defined completed modification requests and software releases as units of output, which were accumulated over a time span of 14 years.

qualification to head a project team. Then, a requirement r_{pst} for this special skill s could model the tasks that must be accomplished by the project manager of project p in period t.

Finally, we address the compatibility between workers. Compatibility between workers means how well two workers cooperate. Compatibility is a complex matter, as it is affected by various personality traits and the work situation (cf. Tett and Murphy, 2002).

Findings about the relationship between worker compatibility and team performance are ambiguous. Intuitively, one would expect compatibility to be positively correlated to performance, as found by Reddy and Byrnes (1972) in an experimental study. However, Hill (1975) found in an empirical study of teams within an IT department that rather incompatibility than compatibility is associated with performance and effectiveness. Hill (1975) suggested that the nature of a task may impact the relation between worker compatibility and effectiveness: The more cooperation the task requires, the more important is compatibility. Though, if "synergistic gains are not possible, incompatibility may lead to higher total accomplishment through the channeling of energy into individual efforts" (Hill, 1975, p. 218).

Our models do not take care of worker compatibility. We assume that workers are equally compatible to one another, i.e., that they are indifferent to the selection of their co-workers.

In the literature, models have been proposed that consider worker compatibility. For example, Kumar et al. (2013) have formulated a MIP model for a problem where tasks must be assigned to workers and where some tasks depend on other tasks. If task j depends on task i, the workers that are assigned to these tasks must cooperate. The more the workers are compatible with each other, the better is their cooperation and the smoother is the corresponding work flow. The objective of the model of Kumar et al. (2013) is to assign tasks to workers such that the total pairwise compatibility of workers who must cooperate is maximized.

We could adopt the approach of Kumar et al. (2013) to modeling compatibility and we could integrate compatibility into our models in two ways. First, we could integrate compatibility into the objective function by adding a term that measures total pairwise compatibility of an assignment of workers to project teams. Second, we could integrate compatibility into the constraints of our models if we wish that total pairwise compatibility within project teams does not fall below a certain level. Both ways imply terms that are nonlinear in the decision variables x_{kp} or y_{kpst}, respectively. Alternatively, additional binary variables would allow to stick to a linear model.

Measuring total pairwise compatibility of a solution to the assignment problem can be done more or less detailed. We could merely consider the x_{kp} variables to measure compatibility solely based on project team membership. Alternatively, we could consider the y_{kpst} variables to weight the compatibility of workers based on actual cooperation. If, for example, worker k contributes to project p only in period $t = 1$, and worker k' contributes to project p only in period $t = 2$, the compatibility between k and k' will not matter much. Although worker k and worker k' join the same project team, the y_{kpst} variables reveal that the need for face-to-face interaction between k and k' is presumably small.

Considering compatibility would require a large quantity of personal data. It might be difficult to obtain these data and to obtain correct data, because workers would have to

reveal the quality of their working relationship to colleagues and are likely to give biased judgements.

If we only want to avoid that two workers k and k' are assigned to the same project, because they are likely to impede project work due to interpersonal conflicts, Constraint set (4.33) can be added to our models. Constraints (4.33) ensure for each project $p \in \mathcal{P}_k^{\text{suit}} \cap \mathcal{P}_{k'}^{\text{suit}}$ that at most one worker of the pair (k, k') is assigned to project p.

$$x_{kp} + x_{k'p} \leq 1 \qquad p \in \mathcal{P}_k^{\text{suit}} \cap \mathcal{P}_{k'}^{\text{suit}} \tag{4.33}$$

If, on the other hand, worker k and worker k' are an inseparable team and neither of them can work without the other, Constraints (4.34) can be added to our models. For each project for which both k and k' are suitable, Constraints (4.34) guarantee that either both are assigned to this project or neither of them.

$$x_{kp} = x_{k'p} \qquad p \in \mathcal{P}_k^{\text{suit}} \cap \mathcal{P}_{k'}^{\text{suit}} \tag{4.34}$$

4.4 Two alternative models for the utilization leveling problem

At the last stage of our three-stage hierarchical planning approach, we want to level the hours worked for the members of each department $d \in \mathcal{D}$ in each period $t \in \mathcal{T}$. The hours worked by employee k in period t comprise the time that he spends for projects and the time that he devotes to his department. At the last stage, the time that worker k will spend for projects in each period $t \in \mathcal{T}$ is already known and given by $\sum_{p \in \mathcal{P}_k^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{kp}^{\text{match}}} y_{kpst}$. The time yd_{kt} that worker k must work for his department in a period t is yet to be determined such that the hours worked by worker k and his colleagues are leveled in this period.

In leveling problems, loads must be determined such that the loads are as equal as possible. Various objective functions for leveling problems have been proposed and considered in the literature, e.g., (1) minimizing the sum of squared loads (cf. Burgess and Killebrew, 1962), (2) minimizing the weighted sum of underloads and overloads (cf. Shanker and Tzen, 1985), (3) minimizing the maximum load (cf. Berrada and Stecke, 1986), (4) minimizing the absolute deviations between desired loads and planned loads (cf. Easa, 1989), (5) minimizing the difference between maximum and minimum load (cf. Guerrero et al., 1999), and (6) minimizing the total pairwise difference of loads or the average pairwise difference of loads (cf. Jang et al., 1996; and Kumar and Shanker, 2001, respectively). Objective functions (1), (2), (4) and (6) are suitable for our problem. We will consider two typical objective functions: a quadratic one, which follows (1), and a linear one, which follows (6).

A quadratic objective function for leveling problems sums the squares of the loads that are to be leveled. In our case loads are hours worked. The sum of loads has to be minimized. Loads that are disproportionately large are punished in the objective function by squaring.

Example 4.3 Assume that department d has two workers k_1 and k_2 whose project contributions in period t are 0. Let $rd_{dt} = 4$ and let $yd_{k_1t}^2 + yd_{k_2t}^2$ be the objective function,

which must be minimized. Then $yd_{k_1t} = yd_{k_2t} = 2$ is the optimal solution with an objective function value of 8. An allocation with $yd_{k_1t} = 1$ and $yd_{k_2t} = 3$, which is less balanced, would result in an objective function value of 10. □

For a quadratic objective function, our problem of allocating departmental workload is given by (4.35)–(4.38). The aim of model (4.35)–(4.38) is to level the total workload of workers who belong to department d in period t. This model has to be solved for each department $d \in \mathcal{D}$ in each period $t \in \mathcal{T}$, i.e., it has to be solved $D \cdot T$ times. The decision variables of the quadratic leveling model are the non-negative continuous variables yd_{kt}, $k \in \mathcal{K}_d$, which represent the departmental workload that is allocated to worker k in period t.

$$\text{Min.} \quad \sum_{k \in \mathcal{K}_d} \left(yd_{kt} + \sum_{p \in \mathcal{P}_k^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{kp}^{\text{match}}} y_{kpst} \right)^2 \tag{4.35}$$

$$\text{s. t.} \quad \sum_{k \in \mathcal{K}_d} yd_{kt} = rd_{dt} \tag{4.36}$$

$$yd_{kt} \leq R_{kt} - \sum_{p \in \mathcal{P}_k^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{kp}^{\text{match}}} y_{kpst} \qquad k \in \mathcal{K}_d \tag{4.37}$$

$$yd_{kt} \geq 0 \qquad k \in \mathcal{K}_d \tag{4.38}$$

Objective function (4.35) minimizes the sum of squared working times of workers from department d in period t. Constraints (4.36) ensure that the entire departmental workload is distributed among department members. Constraints (4.37) guarantee that the time that worker k spends for his department and for projects does not exceed his availability R_{kt}. Constraints (4.38) state the domains of the decision variables.

Model (4.35)–(4.38) can be transformed into an LP by linearizing the quadratic objective function (cf. Williams, 1999, pp. 136–142). Objective function (4.35) is separable, because it can be stated as a sum of terms dependent on a single variable. Each quadratic term yd_{kt}^2, $k \in \mathcal{K}_d$, of the objective function can be approximated by a piecewise linear function. Introducing such a piecewise linear function to the model requires additional variables to represent the line segments of the piecewise linear function: For n line segments, $n + 1$ continuous variables are required; for each line segment, two variables correspond to the endpoints of the interval for which the line segment approximates the quadratic function. Since objective function (4.35) is convex and we minimize this function, the sketched way of linearization, which leads to a linear program, works (cf. Williams, 1999, pp. 139-140). This means that the linearized model matches every feasible value of a variable yd_{kt} with the correct point on the correct line segment.

An optimal solution to the linearized model can deviate from an optimal solution to the original quadratic model. In general, the finer the approximation, i.e., the more line segments are used to describe a quadratic function, the closer the solution of the LP comes to the solution to the original problem.[5]

[5] If all load variables yd_{kt}, $k \in \mathcal{K}_d$, were restricted to integer values, an exact linearization would be possible, though, it would result in a MIP (cf. Rieck et al., 2012).

For our problem, it is also possible to apply a linear objective function, which minimizes the total absolute difference between the working times of all pairs (k, k') of workers, $k \in \mathcal{K}_d$, $k' \in \mathcal{K}_d \setminus \{k\}$. The absolute difference between working times of two workers k and k' can be calculated by the absolute value function. The absolute value function is not a linear function but can be linearized. This linearization is exact, in contrast to the linearization described for the quadratic model. Before we turn to the linearization of the absolute value function, let us consider an example that shows how the absolute value function evaluates different allocations of departmental workload. Our example takes up Example 4.3.

Example 4.4 Assume that department d has two workers k_1 and k_2 whose project contributions in period t are 0. Let $rd_{dt} = 4$ and let $\left|yd_{k_1t} - yd_{k_2t}\right|$ be the objective function, which must be minimized. Then $yd_{k_1t} = yd_{k_2t} = 2$ is the optimal solution with an objective function value of 0. An allocation with $yd_{k_1t} = 1$ and $yd_{k_2t} = 3$, which is less balanced, would result in an objective function value of 2. □

Let us consider the absolute value function from Example 4.4 to demonstrate how an absolute value function can be linearized. For linearization of the function $|yd_{kt} - yd_{k't}|$, we have to introduce a variable $\Delta_{kk'} \in \mathbb{R}_{\geq 0}$. If we require $\Delta_{kk'} \geq yd_{kt} - yd_{k't}$ and $\Delta_{kk'} \geq yd_{k't} - yd_{kt}$, then minimizing $\Delta_{kk'}$ is equivalent to minimizing $|yd_{kt} - yd_{k't}|$.

Now, the linear leveling model can be formulated. It is given by model (4.39)–(4.44). The aim of model (4.39)–(4.44) is to level the total workload of workers who belong to department d in period t. This model has to be solved for each department $d \in \mathcal{D}$ in each period $t \in \mathcal{T}$, i.e., it has to be solved $D \cdot T$ times. The decision variables are the non-negative continuous variables yd_{kt}, $k \in \mathcal{K}_d$, $t \in \mathcal{T}$, which represent the departmental workload that is allocated to worker k in period t. Auxiliary variables are the non-negative continuous variables $\Delta_{kk'}$, $k \in \mathcal{K}_d \setminus \left\{k_{|\mathcal{K}_d|}\right\}$, $k' \in \mathcal{K}_d$, $k' > k$, which represent the absolute difference between the hours worked by workers k and k'. Here, $k_{|\mathcal{K}_d|}$ denotes that worker of department d whose index k is the largest among all department members.

$$\text{Min.} \quad \sum_{k \in \mathcal{K}_d \setminus \left\{k_{|\mathcal{K}_d|}\right\}} \; \sum_{k' \in \mathcal{K}_d \,|\, k' > k} \Delta_{kk'} \tag{4.39}$$

$$\text{s. t.} \quad \Delta_{kk'} \geq yd_{kt} + \sum_{p \in \mathcal{P}_k^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{kp}^{\text{match}}} y_{kpst} - \left(yd_{k't} + \sum_{p \in \mathcal{P}_{k'}^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{k'p}^{\text{match}}} y_{k'pst} \right) \quad \begin{array}{l} k \in \mathcal{K}_d \setminus \left\{k_{|\mathcal{K}_d|}\right\}, \\ k' \in \mathcal{K}_d,\ k' > k \end{array} \tag{4.40}$$

$$\Delta_{kk'} \geq yd_{k't} + \sum_{p \in \mathcal{P}_{k'}^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{k'p}^{\text{match}}} y_{k'pst} - \left(yd_{kt} + \sum_{p \in \mathcal{P}_k^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{kp}^{\text{match}}} y_{kpst} \right) \quad \begin{array}{l} k \in \mathcal{K}_d \setminus \left\{k_{|\mathcal{K}_d|}\right\}, \\ k' \in \mathcal{K}_d,\ k' > k \end{array} \tag{4.41}$$

$$\sum_{k \in \mathcal{K}_d} yd_{kt} = rd_{dt} \tag{4.42}$$

$$yd_{kt} \leq R_{kt} - \sum_{p \in \mathcal{P}_k^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{kp}^{\text{match}}} y_{kpst} \qquad k \in \mathcal{K}_d \tag{4.43}$$

$$yd_{kt} \geq 0 \qquad k \in \mathcal{K}_d \tag{4.44}$$

Objective function (4.39) minimizes the sum of pairwise absolute differences between working times of members of department d in period t. Constraints (4.40) assure that the difference in working times is registered for each pair (k, k'), $k > k'$, where worker k works more hours than worker k'. Constraints (4.41) register the difference in working times for each pair (k, k'), $k > k'$, where worker k works less than worker k'. Constraints (4.42) ensure that the entire departmental workload is distributed among department members. Constraints (4.43) guarantee that the time worker k spends for his department and for projects does not exceed his availability R_{kt}. Finally, Constraints (4.44) state the domains of the decision variables yd_{kt}. The domains of the auxiliary variables $\Delta_{kk'}$ are implicitly defined in Constraints (4.40) and (4.41).

An optimal solution for the quadratic model is also optimal for the linear model and vice versa. Hence, both models are equivalent. Though, from a computational point of view, the linear model seems preferable.

4.5 A monolithic model for all three problems

After we presented models for each stage of the hierarchical planning approach, we will show an integrated, monolithic model for our three problems. The monolithic model can serve as a reference point, which enables us to assess the efficiency of the hierarchical planning approach.

In Section 4.1 we outlined two roads to integrate multiple objectives into a monolithic model. The first road was to consider a weighted sum of the single objectives; the alternative was to optimize a vector of objective functions, where each single objective constituted a component of the vector. The monolithic model that we present here features an objective function that is a weighted sum of the objectives of our three problems.

Let w_1, w_2 and w_3 denote the weights of our three objectives. Weight w_1 corresponds to the objective of selecting the most beneficial project portfolio. Weight w_2 refers to the goal of minimizing the number of assignments of workers to selected projects. Finally, factor w_3 weights the impact of the aim to level working times of workers.

Note that the weight ratios w_1/w_2 and w_2/w_3 must be carefully chosen to obtain desired and sensible results. If the ratio w_1/w_2 is too low, the number of assignments might be minimized by selecting no project at all. In the integrated model, working times cannot only be leveled by allocating departmental workload, but also by allocating project workload. If the ratio w_2/w_3 is too low, the working time could be balanced optimally by allocating project workload to many workers leading to a high number of assignments. If the weight ratios are sufficiently high, the three objectives are lexicographically ordered as in the hierarchical approach.

The monolithic model is given by (4.45)–(4.57). Model (4.45)–(4.57) is a MIP model that comprises two types of binary decision variables and three types of non-negative continuous variables. Binary decision variables are the variables z_p, $p \in \tilde{\mathcal{P}}$, which indicate whether project p is selected or not, and the variables x_{kp}, $k \in \mathcal{K}$, $p \in \hat{\mathcal{P}}_k^{\text{suit}}$, which

indicate whether worker k is assigned to project p or not. The first type of non-negative continuous decision variables are the variables $\hat{y}_{kpst}$, $k \in \mathcal{K}$, $p \in \hat{\mathcal{P}}_k^{\text{suit}}$, $s \in \mathcal{S}_{kp}^{\text{match}}$, $t \in \mathcal{T}_p$, which represent the workload that worker k performs for project p and skill s in period t. The second type of non-negative continuous decision variables are the variables yd_{kt}, $k \in \mathcal{K}$, $t \in \mathcal{T}$, which represent the departmental workload that is allocated to worker k in period t. The third and last type of the non-negative continuous decision variables are the auxiliary variables $\Delta_{kk't}$, $k \in \mathcal{K}_d \setminus \left\{k_{|\mathcal{K}_d|}\right\}$, $k' \in \mathcal{K}_d$, $k' > k$, $d \in \mathcal{D}$, $t \in \mathcal{T}$, which represent the absolute difference in working times in period t between workers k and k' who belong to the same department d. Again, $k_{|\mathcal{K}_d|}$ denotes that worker of department $d \in \mathcal{D}$ whose index k is the largest among all members of department d.

With respect to the decision variables, there are three differences compared to the hierarchical models: First, one set of variables obtains an additional index. All variables used in the hierarchical models are also used in the monolithic model except for the variables $\Delta_{kk't}$. Because leveling cannot be done any longer for each period $t \in \mathcal{T}$ separately, we must add a time index to the variables $\Delta_{kk'}$, resulting in the variables $\Delta_{kk't}$. Both variables have the same meaning. They represent the absolute difference between the hours worked by workers k and k' in the considered period t. The second difference is that the number of the variables x_{kp} increased, because the number of projects that come into question for staffing increased from $|\mathcal{P}|$ to $\left|\mathcal{P}^{\text{ongoing}} \cup \mathcal{P}^{\text{must}} \cup \tilde{\mathcal{P}}\right|$. Finally, the variables $\hat{y}_{kpst}$ do not state the preliminary, but the final allocation of project workload.

$$\text{Min.} \quad -w_1 \sum_{p \in \tilde{\mathcal{P}}} b_p z_p + w_2 \sum_{k \in \mathcal{K}} \sum_{p \in \hat{\mathcal{P}}_k^{\text{suit}}} x_{kp} + w_3 \sum_{d \in \mathcal{D}} \sum_{k \in \mathcal{K}_d \setminus \left\{k_{|\mathcal{K}_d|}\right\}} \sum_{k' \in \mathcal{K}_d \,|\, k' > k} \sum_{t \in \mathcal{T}} \Delta_{kk't} \tag{4.45}$$

$$\text{s. t.} \quad \sum_{k \in \mathcal{K}_s} (l_{ks} \hat{y}_{kpst}) = r_{pst} z_p \qquad \begin{array}{l} p \in \mathcal{P}^{\text{ongoing}} \cup \mathcal{P}^{\text{must}} \cup \tilde{\mathcal{P}}, \\ s \in \mathcal{S}_p,\ t \in \mathcal{T}_p \end{array} \tag{4.46}$$

$$\sum_{s \in \mathcal{S}_{kp}^{\text{match}}} \hat{y}_{kpst} \leq R_{kt} x_{kp} \qquad \begin{array}{l} p \in \mathcal{P}^{\text{ongoing}} \cup \mathcal{P}^{\text{must}} \cup \tilde{\mathcal{P}}, \\ k \in \mathcal{K}_p^{\text{suit}} \setminus \mathcal{K}_p^{\text{assigned}},\ t \in \mathcal{T}_p \end{array} \tag{4.47}$$

$$\Delta_{kk't} \geq yd_{kt} + \sum_{p \in \hat{\mathcal{P}}_k^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{kp}^{\text{match}}} \hat{y}_{kpst} - \left(yd_{k't} + \sum_{p \in \hat{\mathcal{P}}_{k'}^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{k'p}^{\text{match}}} \hat{y}_{k'pst} \right) \qquad \begin{array}{l} k \in \mathcal{K}_d \setminus \left\{k_{|\mathcal{K}_d|}\right\},\ k' \in \mathcal{K}_d, \\ k' > k,\ d \in \mathcal{D},\ t \in \mathcal{T} \end{array} \tag{4.48}$$

$$\Delta_{kk't} \geq yd_{k't} + \sum_{p \in \hat{\mathcal{P}}_{k'}^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{k'p}^{\text{match}}} \hat{y}_{k'pst} - \left(yd_{kt} + \sum_{p \in \hat{\mathcal{P}}_k^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{kp}^{\text{match}}} \hat{y}_{kpst} \right) \qquad \begin{array}{l} k \in \mathcal{K}_d \setminus \left\{k_{|\mathcal{K}_d|}\right\},\ k' \in \mathcal{K}_d, \\ k' > k,\ d \in \mathcal{D},\ t \in \mathcal{T} \end{array} \tag{4.49}$$

$$\sum_{k \in \mathcal{K}_d} yd_{kt} = rd_{dt} \qquad d \in \mathcal{D},\ t \in \mathcal{T} \tag{4.50}$$

$$yd_{kt} + \sum_{p \in \hat{\mathcal{P}}_k^{\text{suit}}(t)} \sum_{s \in \mathcal{S}_{kp}^{\text{match}}} \hat{y}_{kpst} \leq R_{kt} \qquad k \in \mathcal{K}_d,\, t \in \mathcal{T} \tag{4.51}$$

$$z_p = 1 \qquad p \in \mathcal{P}^{\text{ongoing}} \cup \mathcal{P}^{\text{must}} \tag{4.52}$$

$$x_{kp} = 1 \qquad p \in \mathcal{P}^{\text{ongoing}}, k \in \mathcal{K}_p^{\text{assigned}} \tag{4.53}$$

$$z_p \in \{0,1\} \qquad p \in \tilde{\mathcal{P}} \tag{4.54}$$

$$x_{kp} \in \{0,1\} \qquad p \in \mathcal{P}^{\text{ongoing}} \cup \mathcal{P}^{\text{must}} \cup \tilde{\mathcal{P}},\; k \in \mathcal{K}_p^{\text{suit}} \setminus \mathcal{K}_p^{\text{assigned}} \tag{4.55}$$

$$\hat{y}_{kpst} \geq 0 \qquad k \in \mathcal{K},\, p \in \hat{\mathcal{P}}_k^{\text{suit}},\; s \in \mathcal{S}_{kp}^{\text{match}},\, t \in \mathcal{T}_p \tag{4.56}$$

$$yd_{kt} \geq 0 \qquad k \in \mathcal{K},\, t \in \mathcal{T} \tag{4.57}$$

Objective function (4.45) minimizes the weighted sum of our three objectives. Note that maximizing the weighted portfolio benefit is achieved by multiplying the weighted portfolio benefit by -1 and by minimizing the resulting term.

Constraint set (4.46) ensures that each requirement r_{pst} of project p is satisfied if project p is selected. Constraints (4.47) link the variables x_{kp} and $\hat{y}_{kpst}$. These constraints guarantee that worker k can only contribute to project p if he is assigned to project p. A contribution $\hat{y}_{kpst} > 0$ for any skill $s \in \mathcal{S}_{kp}^{\text{match}}$ in any period $t \in \mathcal{T}_p$ requires $x_{kp} = 1$. Simultaneously, Constraints (4.47) force $x_{kp} = 1$ if worker k contributes to project p for any skill $s \in \mathcal{S}_{kp}^{\text{match}}$ in any period $t \in \mathcal{T}_p$. Hence, if worker k contributes to project p, he is automatically assigned to project p.

Constraints (4.48) and (4.49) assure that the absolute difference in working times is registered for each pair (k, k') of workers within each department d in each period t. Constraints (4.50) ensure that the entire departmental workload is distributed among department members for each department d in each period t. Constraints (4.51) guarantee that the time that worker k spends for his department and for projects in period t does not exceed his availability R_{kt}.

Constraint set (4.52) fixes the variables z_p, $p \in \mathcal{P}^{\text{ongoing}} \cup \mathcal{P}^{\text{must}}$, to 1, because we have to include these projects in the portfolio. For each ongoing project $p \in \mathcal{P}^{\text{ongoing}}$, a team $\mathcal{K}_p^{\text{assigned}}$ of workers exists already. For each member k of this team, Constraint set (4.53) fixes the corresponding variable x_{kp} to 1. Eventually, Constraint sets (4.54)–(4.57) state the domains of the actual decision variables. The domains of the auxiliary variables $\Delta_{kk't}$, which are required to obtain a linear term for the leveling goal in the objective function, are implicitly defined in Constraints (4.48) and (4.49).

4.6 Complexity analysis

In this section, we will draw upon complexity theory and use its findings and methods to judge whether our three problems are computationally easy or hard to solve. If we can show that a problem can be classified as a hard problem according to complexity theory, there is almost no hope to find an exact algorithm that solves any instance of the problem to optimality in adequate time. Then, heuristic solution methods can be a resort. Hence, the results of this section can guide our search for solution methods. We

will first introduce the key concepts of complexity theory in Subsection 4.6.1, before we analyze each of our three problems in the following Subsections 4.6.2–4.6.4 separately. The results are summarized in Subsection 4.6.5.

4.6.1 Basic concepts of complexity theory

In this subsection, we will give an overview of basic concepts of complexity theory. Complexity theory deals with—among other things—the complexity of algorithms and the complexity of problems. Algorithms can be classified according to their running time. We will distinguish polynomial-time from exponential-time algorithms. Problems can be assigned to different complexity classes according to their hardness. We will explain the most common complexity classes and sketch how membership of a problem in these classes can be proved.

Complexity theory comprises two main branches (cf. Garey and Johnson, 1979; Wegener, 2003). The first branch addresses running times and memory requirements of algorithms. The running time of an algorithm is also called its time complexity. The second branch entails the hardness of problems. The hardness of problems is also called its complexity. As we will see, both branches are related. The roots of complexity theory lie in the areas of computer science, mathematics, and operations research and started to flourish in the late 1960s (cf. Ahuja et al., 1993, p. 788).

It was said that the first branch of complexity theory addresses running times and memory requirements of algorithms. Nowadays, often the running time of an algorithm is in the spotlight, while memory requirements of an algorithm are less important, because memory space has become abundant and cheap. Since fast solution processes can save money and since algorithms can be enormously complex, the running time of an algorithm remains an important issue, even though processor speed has drastically increased, while processor prices have not increased in the recent decades.

With respect to running times of algorithms, complexity theory seeks to determine the minimum, average, and maximum running time that is required by an algorithm to solve any instance of a problem for which the algorithm was developed. For meaningful statements, the time required is expressed in relation to the instance size (cf. Garey and Johnson, 1979, p. 5). To compare different algorithms for the same problem, usually the maximum time required is considered (cf. Ahuja et al., 1993, pp. 56–57; Wegener, 2003, pp. 25–27; Nemhauser and Wolsey, 1999, p. 119), i.e., comparisons are based on worst-case time complexity.

The maximum time that an algorithm requires to solve instances of a given size is represented by an upper bound O on the number of elementary computational operations that are executed by the algorithm to solve such an instance. The upper bound O is an asymptotic upper bound, which only holds when instance size approaches infinity (Nemhauser and Wolsey, 1999, p. 119; Wegener, 2003, pp. 25–27, cf.). It is assumed that every elementary operation takes one unit of time (cf. Wegener, 2003, p. 22; Schirmer, 1995, p. 3). The number of elementary operations is expressed as a function f of the instance size or, in other words, as a function of the amount of information necessary to represent the instance. The instance size is given by one or more parameters. For our problem, the number of workers K and the number of projects P are parameters that impact instance size.

If only one parameter n specifies the size of an instance, Ahuja et al. (1993, p. 59) define that "an algorithm is said to run in $O(f(n))$ time if for some numbers c and n_0, the time taken by the algorithm is at most $cf(n)$ for all $n \geq n_0$". As an example, consider the time complexities $O(n^2)$, $O(2^n)$ and $O(n!)$. For $O(n^2)$ the function $f(n)$ is a polynomial in n. An algorithm whose running time is bounded by a polynomial in the instance size is called *polynomial-time algorithm*. If for some algorithm the function $f(n)$ is not a polynomial in n, as it is the case with $O(2^n)$ and $O(n!)$, the algorithm is said to run in exponential time and is termed *exponential-time algorithm* (cf. Garey and Johnson, 1979, p. 6).

If more than one parameter describes the size of an instance, the definitions of the previous paragraph apply analogously. As an example, let K and P define the instance size of a problem. Running times of $O(K + P^2)$ or $O(KP)$ are called polynomial, whereas running times of $O(K^P)$ or $O(KP!)$ are called exponential.

The second branch of complexity theory addresses the hardness of problems and classifies problems as computationally easy or hard to solve by assigning them to different complexity classes. Before we outline the most important complexity classes for our work, we will briefly distinguish *decision problems* from *optimization problems* and consider what their difference implies for complexity analysis, because the division of problems into different complexity classes is primarily done for decision problems, whereas our three problems are optimization problems.

Decision problems are problems whose solution is either "yes" or "no" (cf. Garey and Johnson, 1979, p. 18). For example, let us consider an instance of the problem of allocating project workload to workers. The following questions state decision problems: Does a feasible solution exist for the given instance? Does a feasible solution with an objective function value of 5 or less exist for the given instance?

We are, however, concerned with optimization problems, which belong to the broader class of search problems. We want to answer questions such as the following one: "Which feasible allocation of project workload for a given instance requires the least number of assignments of workers to projects?"

Before we explain how the hardness of a decision problem can be related to the hardness of an optimization problem, we define two phrases that are used in the following:

(1) an optimization problem and its *corresponding* decision problem, and

(2) a decision problem and its *corresponding* optimization problem.

Let us define the phrases by examples in a rather informal way. For (1) assume that the optimization problem "Max. (Min.) b subject to some constraints" is given. Then, the corresponding decision problem is defined as "Is there a feasible solution for the optimization problem with $b \geq c$ ($b \leq c$)?". Vice versa, for (2) consider the following decision problem: "Is there a solution with $b \geq c$ ($b \leq c$) that observes all constraints of a given problem?" Then, the corresponding optimization problem is defined as "Max. (Min.) b subject to all constraints of the given problem". Based on our definition, the corresponding optimization problem of a decision problem is uniquely defined, while the corresponding decision problem of an optimization problem is uniquely defined except for the value of c. Since the value of c is not relevant for our purposes, we consider the corresponding decision problem of an optimization problem as uniquely defined.

Garey and Johnson (1979, pp. 109–117) and Wegener (2003, pp. 50–53) use the concept of Turing reducibility, also called polynomial reducibility (cf. Schirmer, 1995, pp. 18–20), to show how results for the complexity of a decision problem can be transferred to the corresponding optimization problem. Later, we will consider this concept in more detail. For now, we content ourselves with the fact that the concept of Turing reducibility allows to conclude that an optimization problem is hard if its corresponding decision problem is hard. Hence, we can examine the decision problems that correspond to our optimization problems in order to obtain insights into the complexity of the optimization problems.

A focus of complexity theory is on the class NP (*non-deterministic polynomial-time*), which contains all decision problems that can be solved by a non-deterministic algorithm in polynomial time. Such an algorithm is a theoretical type of algorithm that decomposes the process of solving a problem into two stages: a *guessing stage* and a *checking stage*. At the first stage, a solution to the problem is guessed. The solution is also called instance. This is the non-deterministic guessing stage. At the second stage, it is checked in polynomial time if the solution is feasible (yes-instance) or not (no-instance). This is the polynomial-time checking stage. The statement that a problem that is in NP can be solved in polynomial time by this theoretical type of algorithm refers only to the checking stage and only to the case where a yes-instance is checked (cf. Nemhauser and Wolsey, 1999, pp. 128–129). The decision problems that correspond to our three optimization problems belong to the class NP what we will prove for each problem in the following subsections.

A major contribution of complexity theory is the classification of decision problems that belong to the class NP into three different subclasses. These three main subclasses are the class P (*deterministic polynomial-time*) of "easy" problems, the class NPC (*NP-complete*) of "hard" problems, and the class NPI (*NP-intermediate*) of problems whose complexity is between the complexities of the classes P and NPC (cf. Garey and Johnson, 1979, pp. 154–161). Problems in class P can be solved by polynomial-time algorithms. For problems of class NPC only exponential-time algorithms are known, but until now it could not be ruled out that polynomial-time algorithms for these problems might be developed. If a polynomial-time algorithm was found for one problem out of NPC, every problem of the class NPC could be solved in polynomial time. This would imply $\mathsf{P} = \mathsf{NP}$. Though, a majority of researchers supposes that the conjecture $\mathsf{P} \neq \mathsf{NP}$ is true. A proof or a falsification of this conjecture, however, remains to be presented and is currently the presumably greatest challenge in the field of complexity theory.

Among the problems that belong to the class NPC, there is a special type of problems called *number problems* (cf. Garey and Johnson, 1979, pp. 90–106). Some of these number problems are computationally better tractable than other number problems and non-number problems. We will succinctly treat this special type of problems in order to be able to draw potential conclusions for our problems.

Number problems are problems where the largest number that appears within the instance data cannot be bounded by a polynomial in the instance size. As an example for a number problem, consider the knapsack problem, where a subset from n items has to be selected that has maximum total utility and a total weight not exceeding the knapsack capacity c. An instance comprises n weights, n utility values and the capacity value c. Without loss of generality, let c be the largest number of the instance. The instance size, i.e., the amount of information necessary to represent the instance is bounded by

$O(n \log_2 c)$ if numbers are encoded using the binary representation. Because the maximum number c cannot be bounded by $O(n \log_2 c)$, the knapsack problem is a number problem.

Another number problem, which also belongs to NPC, is the traveling salesman problem. In this problem, we seek a tour of minimum length through n cities. The distance between any two cities cannot be bounded by a polynomial in the instance size.

In contrast, the minimum cover problem[6], which also belongs to NPC, is not a number problem, i.e., it is a non-number problem. In the minimum cover problem, a set C is given consisting of sets C_i, $i = 1, \ldots, |C|$, where each set C_i is a subset of a finite set F. Additionally, a positive integer $b \leq |C|$ is given. The question is whether C contains a cover of F that has a size of at most b, i.e., whether there is a subset $C' \subseteq C$ with $|C'| \leq b$ such that every element of F is contained in at least one set $C_i \in C'$ (cf. Garey and Johnson, 1979, p. 222). The only and thus largest number that appears in an instance of the minimum cover problem is the integer b. The instance size is bounded by $O(|C||F| + \log_2 b)$. Since $b \leq |C|$ holds by definition, b is bounded by a polynomial in C and hence by a polynomial in the instance size. Consequently, the minimum cover problem is not a number problem.

As for all NP-complete problems, no algorithms for number problems have been found whose running time is bounded by a polynomial in the instance size. However, for some number problems, algorithms exist whose running time is bounded by a polynomial in the instance size and in the maximum number of the instance. These algorithms are called *pseudopolynomial-time algorithms*. Those number problems that can be solved by pseudopolynomial-time algorithms are better tractable than all other problems in NPC, be it number problems or not.

As an example for a number problem that can be solved in pseudopolynomial time, consider the knapsack problem. It can be solved by dynamic programming in $O(nc)$ time by filling in a table with nc cells (cf. Garey and Johnson, 1979, pp. 90–92; Martello and Toth, 1990, p. 7). This time complexity implies that instances of the knapsack problem can be solved in polynomial time with respect to instance size if the largest number c of these instances is rather small such that c can be bounded by a polynomial in the number of items n.

On the other side, there are number problems for which no pseudopolynomial-time algorithm is known, e.g., the traveling salesman problem. Even if all numbers within an instance of the traveling salesman problem are small, it takes exponential time to solve the instance. Even if all intercity distances are either 1 or 2, for example, no polynomial-time algorithm is known.

Due to the existence of these two types of number problems, the class NPC is further divided. All problems that belong to the class NPC are called NP-complete. Number problems for which no pseudopolynomial-time algorithm is known and all non-number problems in NPC are called NP-*complete in the strong sense* and belong to the subclass sNPC (*strongly* NP-*complete*) (cf. Garey and Johnson, 1979, p. 95). For instance, the traveling salesman problem and the minimum cover problem are strongly NP-complete, whereas the knapsack problem is only NP-complete.

The distinction of number problems from non-number problems is relevant for our work, because arbitrarily large numbers can appear in instances of our three problems. Consider the workforce assignment problem, for example. For the decision and opti-

[6]The minimum cover problem is also called set-covering problem.

mization version of this problem, instance size is bounded by $O(KT\log_2(\max_{k,t} R_{kt}) + KS\log_2(\max_{k,s} l_{ks}) + DT\log_2(\max_{d,t} rd_{dt}) + PST\log_2(\max_{p,s,t} r_{pst}))$. Thus, the worker availabilities R_{kt}, the departmental requirements rd_{dt}, and the project requirements r_{pst} are not bounded by a polynomial in the instance size. Recall that we bounded the skill levels l_{ks} by 2.

In the following three subsections we will prove that the decision problems that correspond to our three optimization problems belong to P, NPC or even sNPC. To prove membership in the class P for a decision problem Π, it is sufficient to state a polynomial-time algorithm which solves Π. For a detailed description of proving techniques with regard to membership in the classes NPC and sNPC see Garey and Johnson (1979, pp. 63–74 and 95–106). We will only sketch how membership in these two classes is proved by giving two formal proof definitions. Additionally, we briefly explain a proving technique called *restriction*, which facilitates the application of the rather theoretical and formal proof definitions.

To prove that a problem Π belongs to the class NPC, two steps are necessary. We first have to show that Π belongs to NP. Finally, a problem Π^{NPC} that is known to be in the class NPC must be polynomially transformed to Π such that a yes-instance (no-instance) of Π^{NPC} is transformed into a yes-instance (no-instance) of Π. If such a polynomial transformation exists and if an algorithm existed that would solve Π, this algorithm would also solve Π^{NPC}. Since no polynomial-time algorithm exists for Π^{NPC} if $\mathsf{P} \neq \mathsf{NP}$ is true, there exists no polynomial-time algorithm for Π and thus Π must belong to NPC.

To prove that a number problem Π^{Num} belongs to the class sNPC, two alternative ways are possible. Both ways are closely related to the way just outlined for proving NP-completeness. The first alternative requires two steps. First, we have to restrict problem Π^{Num} to a problem Π_p^{Num} where all numbers are bounded by a polynomial p in the instance size. Second, we must show that this problem Π_p^{Num} is NP-complete. The second alternative requires to show that Π^{Num} belongs to the class NP and that a problem Π^{sNPC} that is known to be strongly NP-complete can be pseudopolynomially transformed to Π^{Num}.[7]

To apply the proof definitions, which essentially rely on a transformation of one problem into another, proving techniques have emerged. A common technique for proving that a problem Π is (strongly) NP-complete uses the principle of *restriction* (cf. Garey and Johnson, 1979, pp. 63–66). If a (strongly) NP-complete problem exhibits a one-to-one correspondence to the problem Π, it can be easily transformed to Π. Though, often it is difficult to find such a (strongly) NP-complete problem. Then, it might be possible to restrict Π to a special case for which a transformation from a known (strongly) NP-complete problem can easily be constructed. Restricting Π to a special case means that Π is restricted to a proper subset of all its instances. If it can be shown that the restricted problem is (strongly) NP-complete, so its generalization Π is, because there is no algorithm that solves *any* instance of Π in (pseudo)polynomial time. We will apply the principle of restriction in the following subsections.

The classes NP, P, NPC, and sNPC refer to decision problems only, but for optimization problems a comparable classification is common. If a decision problem is NP-complete, the corresponding optimization problem is called NP-*hard*, and if a decision problem is

[7] For the definition of a pseudopolynomial transformation see Garey and Johnson (1979, pp. 101–106).

strongly NP-complete, the corresponding optimization problem is called NP-*hard in the strong sense* or just *strongly* NP-*hard* (cf. Schirmer, 1995, p. 20 and p. 26). The term (strongly) NP-hard means that a (strongly) NP-hard optimization problem is at least as hard as any decision problem that is (strongly) NP-complete.

To show that a (strongly) NP-hard optimization problem is at least as hard as any (strongly) NP-complete decision problem, the aforementioned concept of Turing reducibility can be applied. A Turing reduction of a problem Π to a problem Π' is similar to a polynomial transformation except for the fact that a Turing reduction allows to solve Π by repeatedly calling a subroutine to solve different instances of Π' (cf. Garey and Johnson, 1979, p. 111; Schirmer, 1995, pp. 118–120). An optimization problem Π_{opt} is (strongly) NP-hard if there is an NP-complete problem Π^{NPC} (strongly NP-complete problem Π^{sNPC}) that Turing-reduces to Π_{opt}. Since any decision problem Turing-reduces to its corresponding optimization problem, an optimization problem is (strongly) NP-hard if the corresponding decision problem is (strongly) NP-complete. Hence, we can derive complexity results for our optimization problems from the complexity results that we obtain for their corresponding decision problems.

4.6.2 Complexity of the project selection problem

In this subsection, we prove that our project selection problem is NP-hard in the strong sense.

The project selection problem was described in Section 3.2 and modeled in Section 4.2. The corresponding decision problem asks whether there is a feasible portfolio with a total benefit of at least b, $b \in \mathbb{N} \setminus \{0\}$.

To simplify our presentation, we abbreviate our optimization problem as $\text{MPSWS}_{\text{opt}}$ (multi-project skilled workforce selection problem) and its corresponding decision problem as $\text{MPSWS}_{\text{dec}}$. Additionally, let the vector $\boldsymbol{z}$ represent values for all variables z_p, $p \in \mathcal{P}^{\text{ongoing}} \cup \mathcal{P}^{\text{must}} \cup \tilde{\mathcal{P}}$, and let the matrix $\hat{\boldsymbol{y}}$ represent values for all variables $\hat{y}_{kpst}$, $k \in \mathcal{K}$, $p \in \hat{\mathcal{P}}_k^{\text{suit}}$, $s \in \mathcal{S}_{kp}^{\text{match}}$, $t \in \mathcal{T}_p$.

Lemma 4.1 $\text{MPSWS}_{\text{dec}} \in \text{NP}$. □

Proof Let $(\boldsymbol{z}, \hat{\boldsymbol{y}})$ be a (guessed) solution for an arbitrary instance of $\text{MPSWS}_{\text{dec}}$. We can check in polynomial time if this solution is feasible and if the portfolio that is associated with $\boldsymbol{z}$ has a total benefit of at least b. The feasibility check requires that we test whether the solution satisfies Constraint sets (4.2)–(4.7). The number of constraints within these sets is bounded by a polynomial in the instance size. ■

Theorem 4.1 $\text{MPSWS}_{\text{dec}}$ *is strongly* NP-*complete.* □

Proof (Polynomial transformation from Minimum Cover)
Problem: Minimum Cover
Instance: Set C containing sets C_i, $i = 1, \ldots, |C|$, where each set C_i is a subset of a finite set F; positive integer $b \leq |C|$.
Question: Does C contain a cover of F that has a size of at most b, i.e., is there a subset $C' \subseteq C$ with $|C'| \leq b$ such that every element of F is contained in at least one set $C_i \in C'$?

MINIMUM COVER is NP-complete in the strong sense even if $|C_i| \leq 3$ holds for all $C_i \in C$ with $|C_i| = 3$ for at least one C_i (cf. Garey and Johnson, 1979, p. 222).

For this proof, we restrict $\text{MPSWS}_{\text{dec}}$ to the case where $T = 1$; $R_{kt} = 1$, $k \in \mathcal{K}$, $t = 1$; $rd_{dt} = 0$, $d \in \mathcal{D}$, $t = 1$; $\left|\mathcal{P}^{\text{ongoing}}\right| = \left|\mathcal{P}^{\text{must}}\right| = 0$ and $\left|\tilde{\mathcal{P}}\right| = K+1$, i.e., where $K+1$ projects can be selected.

We assume that the set of skills $\mathcal{S}$ is a union of two disjoint sets $\mathcal{S}^F$ and $\mathcal{S}^{\text{unique}}$ with $\left|\mathcal{S}^{\text{unique}}\right| = K$. Each unique skill $s \in \mathcal{S}^{\text{unique}}$ is mastered by exactly one worker and each worker $k \in \mathcal{K}$ masters one unique skill $s \in \mathcal{S}^{\text{unique}}$ and between one and three additional skills from $\mathcal{S}^F$. For each skill $s \in \mathcal{S}$ and each worker $k \in \mathcal{K}$, we presume $l_{ks} = 1$.

From the set $\tilde{\mathcal{P}}$ of projects, K projects are assumed to require only one skill, namely, a unique skill $s \in \mathcal{S}^{\text{unique}}$, but no two of these projects p, $p = 1, \ldots, K$, require the same unique skill, i.e., each unique skill $s \in \mathcal{S}^{\text{unique}}$ is required by exactly one project. For each project p, $p = 1, \ldots, K$, its requirement is given by $r_{pst} = 1$, $s \in \mathcal{S}_p$, $t = 1$, and the benefit is given by $b_p = 1$. The remaining project $p = K + 1$ requires all the skills $s \in \mathcal{S}^F$, but no skill from $\mathcal{S}^{\text{unique}}$. Let project $p = K + 1$ have requirements $0 < r_{pst} \leq \frac{1}{3}$, $s \in \mathcal{S}^F$, $t = 1$, and a benefit $b_p = K+1$. Note that our restriction of $\text{MPSWS}_{\text{dec}}$ leads to problem instances in which all numbers are bounded by a polynomial in the instance size.

To tie an instance of MINIMUM COVER to an instance of $\text{MPSWS}_{\text{dec}}$, we associate the set F with the set $\mathcal{S}^F$ of skills that are required by project $p = K + 1$. Let each subset C_i be associated with a worker k and let the elements in C_i correspond to the skills in $\mathcal{S}_k \cap \mathcal{S}^F$ that are mastered by the associated worker apart from his unique skill. Then there exists a cover of F that has size b or less if and only if a feasible portfolio can be selected with a total benefit of at least $K + 1 + (K - b)$.

Note that a feasible portfolio with a total benefit of $K + 1 + (K - b)$ necessitates that $K - b$ workers must spend their entire time to accomplish the project that requires their unique skill $s \in \mathcal{S}^{\text{unique}}$. This necessity leaves only b workers who can contribute to project $p = K + 1$. Together with Lemma 4.1 our proof is complete. ■

Corollary 4.1 $\text{MPSWS}_{\text{opt}}$ *is strongly* NP*-hard.* □

To be more precise, the proof has shown that $\text{MPSWS}_{\text{opt}}$ is strongly NP-hard if every worker masters at least one unique skill and if at least one worker masters four skills or more in total. By a second proof, we will show that not only instances where each worker masters a unique skill and some other skills are strongly NP-hard but also instances where each worker masters only one skill. In addition, the second proof points to a special case that can be solved in pseudopolynomial time. Our second proof for Theorem 4.1 reads as follows.

PROOF (Pseudopolynomial transformation from MULTIDIMENSIONAL KNAPSACK)
Problem: MULTIDIMENSIONAL KNAPSACK (cf. Kellerer et al., 2004, pp. 235–238)
Instance: Set J of n items; set I of m dimensions such as weight, volume and concentration; benefit p_j, $j \in J$; size w_{ij}, $i \in I$, $j \in J$, of item j with respect to dimension i; positive integer c_i, $i \in I$, that represents the capacity with respect to dimension i; positive integer b.
Question: Is there a subset $J' \subseteq J$ such that $\sum_{j \in J'} w_{ij} \leq c_i$, $i \in I$, and $\sum_{j \in J'} p_j \geq b$?

MULTIDIMENSIONAL KNAPSACK is NP-complete, because it comprises the NP-complete (one-dimensional) knapsack problem as a special case ($m = 1$). MULTIDIMENSIONAL KNAPSACK can be solved by dynamic programming in $O\left(n \left(\max_i c_i\right)^m\right)$ time

by filling a table with $n \times c_1 \times c_2 \times \cdots \times c_m$ cells (cf. Kellerer et al., 2004, pp. 248–252). Hence, if the number of dimensions m is bounded by a constant, MULTIDIMENSIONAL KNAPSACK can be solved in pseudopolynomial time, but in general, MULTIDIMENSIONAL KNAPSACK is strongly NP-complete (cf. Kaparis and Letchford, 2008, p. 91).

To prove NP-completeness of $\text{MPSWS}_{\text{dec}}$ by transformation from MULTIDIMENSIONAL KNAPSACK, we restrict $\text{MPSWS}_{\text{dec}}$ to the special case where $\mathcal{P}^{\text{ongoing}} = \mathcal{P}^{\text{must}} = \emptyset$; $T = 1$; $rd_{dt} = 0$, $d \in D$, $t = 1$; $\mathcal{S}_p = \mathcal{S}$, $p \in \tilde{\mathcal{P}}$; and $|\mathcal{S}_k| = 1$, $k \in \mathcal{K}$, i.e., where each worker masters only one skill. Additionally, we assume $l_{ks} = 1$, $k \in \mathcal{K}$, $s \in \mathcal{S}_k$.

Let each project $p \in \tilde{\mathcal{P}}$ be associated with an item j and let each skill s out of the finite set of skills be associated with a dimension i. Let for all projects $p \in \tilde{\mathcal{P}}$ each requirement r_{pst}, $s \in \mathcal{S}_p$, $t = 1$, be associated with the corresponding item size w_{ij} and let the knapsack capacities c_i, $i \in I$, correspond to the total workforce availability with regard to the associated skill s given by $\sum_{k \in \mathcal{K}_s} R_{kt}$, $t = 1$. Then, an instance of MULTIDIMENSIONAL KNAPSACK is a yes-instance if and only if the associated instance of $\text{MPSWS}_{\text{dec}}$ is a yes-instance. Together with Lemma 4.1 our proof is complete. ∎

Corollary 4.2 $\text{MPSWS}_{\text{opt}}$ *is strongly* NP*-hard even if each worker masters only one out of several skills. The special case of* $\text{MPSWS}_{\text{opt}}$ *that corresponds to the restricted decision problem outlined in the second proof of Theorem 4.1 can be solved in pseudopolynomial time for any fixed number of skills* $|\mathcal{S}|$. □

However, this special case of $\text{MPSWS}_{\text{opt}}$ where only mono-skilled workers with homogeneous skill levels are considered is far off those practical cases that we are interested in. The project selection problem that prevails in practice is NP-hard in the strong sense.

4.6.3 Complexity of the workforce assignment problem

In this subsection, we show that our optimization problem of assigning workers to projects and allocating project workload to workers is NP-hard in the strong sense.

The problem was described in Section 3.3 and modeled in Subsection 4.3.1. The corresponding decision problem asks whether there is a feasible allocation of project workload that results in no more than b assignments of workers to projects, $b \in \mathbb{N} \setminus \{0\}$.

To shorten our presentation, we abbreviate our optimization problem as $\text{MPSWA}_{\text{opt}}$ (multi-project skilled workforce assignment problem) and its corresponding decision problem as $\text{MPSWA}_{\text{dec}}$. Additionally, let the matrix $\boldsymbol{x}$ represent values for all variables x_{kp}, $k \in \mathcal{K}$, $p \in \mathcal{P}_k^{\text{suit}}$, and let the matrix $\boldsymbol{y}$ represent values for all variables y_{kpst}, $k \in \mathcal{K}$, $p \in \mathcal{P}_k^{\text{suit}}$, $s \in \mathcal{S}_{kp}^{\text{match}}$, $t \in \mathcal{T}_p$.

Lemma 4.2 $\text{MPSWA}_{\text{dec}} \in$ NP. □

PROOF Let $(\boldsymbol{x}, \boldsymbol{y})$ be a (guessed) solution for an arbitrary instance of $\text{MPSWA}_{\text{dec}}$. We can check in polynomial time if the solution is feasible and if the number of assignments of workers to projects that is associated with $\boldsymbol{x}$ does not exceed b. A check whether the solution is feasible requires to test if the solution satisfies Constraint sets (4.10)–(4.16). The number of constraints within these sets is bounded by a polynomial in the instance size. ∎

Theorem 4.2 $\text{MPSWA}_{\text{dec}}$ *is strongly* NP*-complete.* □

Proof (Polynomial transformation from Minimum Cover)
We restrict $\text{MPSWA}_{\text{dec}}$ to the special case where $P = 1$; $\mathcal{S}_p = \mathcal{S}$, $p = 1$; $T = 1$; $rd_{dt} = 0$, $d \in \mathcal{D}$, $t = 1$. Let us assume that $R_{kt} = \infty$, $k \in \mathcal{K}$, $t = 1$, i.e., that the workers' availabilities are unbounded.[8]

Now, let the set F of an instance of Minimum Cover (see page 79) be associated with the set $\mathcal{S}$ of skills. Let each subset C_i be associated with a worker k and let the elements of C_i correspond to the skills in $\mathcal{S}_k$ that are mastered by the associated worker. Then there is a cover of F that has size b or less if and only if there is a feasible allocation of the workload of project $p = 1$ with a team size of b or less. Together with Lemma 4.2 our proof is complete. ∎

Remark 4.1 If the unbounded version of the decision problem is strongly NP-complete, all the more is the bounded version. □

Remark 4.2 Minimum Cover can be solved in polynomial time by matching techniques if $|C_i| \leq 2$ holds for all $C_i \in C$. We will succinctly outline an efficient solution procedure that incorporates matching techniques, but first we will give some definitions on which the procedure is founded.

An undirected graph $G(N, E)$ is a set N of nodes and a set E of edges. Each edge $e \in E$ joins two nodes u and v. Both nodes u and v are said to be incident with e. A matching M in graph G is a subset of edges in E such that every node $v \in N$ is incident with at most one edge in M (cf. Burkard et al., 2009, p. 2; Jungnickel, 2005, p. 205). A matching M is a maximum matching if its cardinality $|M|$ is maximal, i.e., if there is no matching M' with $|M'| > |M|$.

To tackle an instance of Minimum Cover by a matching technique, we construct a graph G by establishing a node v for each element $f \in F$ that is given in the instance of Minimum Cover. For each subset $C_i \in C$ with $|C_i| = 2$, an edge e is established that joins those two nodes u and v that correspond to the elements $f \in C_i$.

To find a cover of F, a maximum matching M is determined for this graph G, e.g., by the algorithm of Edmonds (cf. Edmonds, 1965; Jungnickel, 2005, pp. 374–396). If the number of nodes $|N|$ is even and $|M| = \frac{|N|}{2}$, then the matching M represents a cover of F. Otherwise, M represents only a partial cover. If M represents only a partial cover of F, we extend this partial cover step by step, until all elements of F are covered. In each step, we either extend the partial cover by a subset C_i with $|C_i| = 2$ whose corresponding edge does not belong to M or by a subset C_i with $|C_i| = 1$ whose element f has not been covered yet.

The minimum number of subsets C_i necessary for a cover of F is given by the sum of the cardinality $|M|$ of the matching and the number of subsets C_i that are required to extend the partial cover to a full cover. This minimum number can be compared to b to answer the question whether there is a cover of size b or less. □

[8] Instead of unbounded worker availabilities we could alternatively choose the project requirements r_{pst} for the sole project $p = 1$ so small that every worker k could accomplish the workload for all skills $s \in \mathcal{S}_{kp}^{\text{match}}$ of his matching skills if he was assigned to the project. In consequence, the largest number that appears in an instance of the restricted version of $\text{MPSWA}_{\text{dec}}$ is bounded by a polynomial in the instance size.

Remark 4.2 is relevant for instances of our assignment problem where no worker masters more than two skills and where the project requirements are very small when compared to the availabilities of workers. For these instances, the solution procedure that was outlined in Remark 4.2 can be applied to find the minimum team size if only one project must be staffed. If more projects must be staffed, say $P \geq 2$ projects, then the solution procedure must be applied P times, once for each project.

For a clearer picture of the complexity of $\text{MPSWA}_{\text{dec}}$ with respect to its subproblems, we will provide a second proof showing that $\text{MPSWA}_{\text{dec}}$ is strongly NP-complete. Our second proof is not redundant in so far, as it shows that there are even instances of $\text{MPSWA}_{\text{dec}}$ that feature only one skill but cannot be solved in polynomial or pseudopolynomial time. Our first proof did only show that instances with $|\mathcal{S}_k| \geq 3$ for at least one worker k are intractable. The following second proof, which relies on transformation from 3-Partition, will reveal conditions where instances with $|\mathcal{S}| = 1$ are intractable. Our second proof does not render our first proof redundant, as the first proof shows that $\text{MPSWA}_{\text{dec}}$ is strongly NP-complete even if worker availabilities are unbounded. Hence, both proofs shed precious light on the frontier between hard and easy problems (cf. Garey and Johnson, 1979, pp. 80–90).

As a vehicle for the second proof, we use the network model, which was introduced in Subsection 4.3.1, to represent our assignment problem. Before we will present the proof, let us shortly turn towards two variants of a network design problem that are related to our assignment problem and that led us the way to our proof. The two variants are the *fixed-charge network flow problem* and the *minimum edge-cost flow problem*.

Both the fixed-charge network flow problem (cf. Kim and Pardalos, 1999; Cruz et al., 1998; Magnanti and Wong, 1984) and the minimum edge-cost flow problem (cf. Garey and Johnson, 1979, p. 214) seek for a minimum cost origin-destination flow on a network with bounded arc capacities where fixed arc costs are incurred whenever an arc transports a positive flow.

There are only slight differences between the two network design problems. The fixed-charge network flow problem considers not only fixed arc costs, but in addition also variable arc costs which linearly depend on the amount of flow shipped on the arc. Furthermore the flow variables are continuous variables, whereas the minimum edge-cost flow problem presumes integral flow variables.

Both network design problems are NP-complete in the strong sense. For the fixed-charge network flow problem, Guisewite and Pardalos (1990) prove this problem complexity by transformation from 3-Satisfiability (abbreviated 3SAT, cf. Garey and Johnson, 1979, p. 259). For the minimum edge-cost flow problem, strong NP-completeness is indicated by Garey and Johnson (1979, p. 214), who refer to a transformation from Exact Cover by 3-Sets (abbreviated X3C, cf. Garey and Johnson, 1979, p. 221). This transformation from Exact Cover by 3-Sets is shown by Benoist and Chauvet (2001, pp. 2–3), for example.

To recognize the relation of our assignment problem to network design, see that we could restrict our problem to a special case of the fixed-charge network flow problem. For our proof, however, we will exploit the relation to the minimum edge-cost flow problem. Our proof follows Benoist and Chauvet (2001, pp. 5–6), who show that the minimum edge-cost flow problem remains strongly NP-complete for a special case which they call

bipartite minimum edge-cost flow problem.[9] Their proof relies on transformation from 3-PARTITION. We will take on this transformation to prove that $\text{MPSWA}_{\text{dec}}$ is intractable even if it features only one skill.

Theorem 4.3 $\text{MPSWA}_{\text{dec}}$ *is strongly* NP*-complete even if restricted to one skill, i.e., even if* $|\mathcal{S}| = 1$. □

PROOF (Pseudopolynomial transformation from 3-PARTITION)
Problem: 3-PARTITION
Instance: Set C that contains $3m$ elements, $m \geq 3$, $m \in \mathbb{N}$; a positive integer B; a size $s(c) \in \mathbb{N} \setminus \{0\}$ for each $c \in C$ such that $\frac{B}{4} < s(c) < \frac{B}{2}$ for all $c \in C$ and $\sum_{c \in C} s(c) = mB$.
Question: Can C be partitioned in m disjoint subsets $C_1, C_2, \ldots, C_m$ such that $\sum_{c \in C_i} s(c) = B$ holds for $i = 1, \ldots, m$? (If so, every subset C_i must contain exactly three elements from C.)

3-PARTITION is NP-complete in the strong sense (cf. Garey and Johnson, 1979, p. 224).

To prepare a transformation, we picture an instance of 3-PARTITION by a bipartite graph $G(N, A)$ with node set N and arc set A. Graph G is depicted in Figure 4.2. The set N of nodes is a union of the disjoint node sets U and V. Arcs run only from nodes in U to nodes in V and arc capacity is not bounded. The graph G represents a directed network with a set of supply nodes, namely, the set U, and a set of demand nodes, namely, the set V. Concretely, for each element $c \in C$ a node u is established whose supply is equal to the size $s(c)$. The node set V comprises m nodes, which have an identical supply of $-B$, i.e., a demand of B units of flow. The arc set comprises the arcs $\langle u, v \rangle$, $u \in U$, $v \in V$.

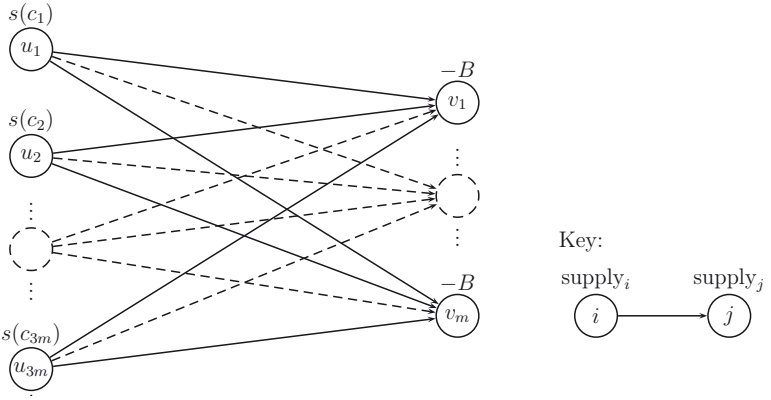

Figure 4.2: Network representation of an instance of 3-PARTITION for transformation to $\text{MPSWA}_{\text{dec}}$

[9] The bipartite minimum edge-cost flow problem was formulated by Benoist and Chauvet (2001) for a problem in construction industry. This problem is described in detail by Benoist (2007).

A solution to an instance of 3-PARTITION is represented by a flow from nodes $u \in U$ to nodes $v \in V$ that does not exceed the supply of nodes in U and that satisfies all demands of nodes in V and that does not require more than $3m$ arcs to transport the flow units.

For our proof, we restrict $\text{MPSWA}_{\text{dec}}$ to the special case where $T = 1$; $D = 1$; $P = m - 1$; $S = 1$; and $l_{ks} = 1$, $k \in \mathcal{K}$, $s = 1$. In this case, all workers belong to the same unique department and all projects require the same unique skill.

For a transformation from 3-PARTITION to $\text{MPSWA}_{\text{dec}}$, let each element c, i.e., each node u, correspond to a worker k and let the supply of node u, i.e., $s(c)$, be associated with the availability R_{kt}, $t = 1$, of the corresponding worker k. Furthermore, we associate each of the $m - 1$ projects with a demand node v and we associate the requirement r_{pst} of a project p for the unique skill s in period $t = 1$ with the demand of the corresponding node v. Finally, let the sole department $d = 1$ correspond to a node v and let its requirement r_{dt}, $t = 1$, correspond to the demand of the node v. This demand amounts to B units of flow.

Then, an instance of 3-PARTITION is a yes-instance if and only if there is an allocation of project workload such that the number of assignments of workers to projects does not exceed $3(m - 1)$. Note that $3(m - 1)$ assignments imply that each project is staffed with three workers, who spend their entire available time for this project, while three workers who are not assigned to any project spend their entire time to accomplish the departmental workload. Together with Lemma 4.2 our proof is complete. ∎

Remark 4.3 Since 3-PARTITION remains strongly NP-complete as long as $m \geq 3$, $\text{MPSWA}_{\text{dec}}$ is NP-complete in the strong sense if two projects must be staffed and a unique department has a positive requirement, or if at least three projects must be staffed. □

Remark 4.4 The special case where two projects must be staffed and no departmental workload arises, is NP-complete. This complexity result can be concluded from equivalent proofs of Guisewite and Pardalos (1990) and Benoist and Chauvet (2001). Both contributions show by transformation from SUBSET SUM (cf. Garey and Johnson, 1979, p. 223) that the corresponding fixed-charge network flow problem and the corresponding minimum edge-cost flow problem, respectively, are NP-complete if there are only two demand nodes. By their proofs, both works help to clarify the sight on the boundary line between NP-completeness and strong NP-completeness for the respective problem.

Note that the conclusion for our assignment problem does not hold if only one project must be staffed and a positive requirement of one department must be satisfied. Although this case results in two demand nodes, transformation from SUBSET SUM is not possible in this case. □

To sum up, we have the following three complexity results for the optimization version $\text{MPSWA}_{\text{opt}}$ of our assignment problem.

Corollary 4.3 $\text{MPSWA}_{\text{opt}}$ *can be solved in polynomial time if workers master at most two skills and worker availabilities are "unbounded".* □

Corollary 4.4 $\text{MPSWA}_{\text{opt}}$ *is* NP*-hard if two projects must be staffed and worker availabilities are bounded.* □

Corollary 4.5 $\text{MPSWA}_{\text{opt}}$ *is* NP*-hard in the strong sense if at least one worker masters three or more skills that are required by a project, or if at least three projects must be staffed and worker availabilities are bounded, or if two projects must be staffed and in the sole department some work must be accomplished and worker availabilities are bounded.* □

4.6.4 Complexity of the utilization leveling problem

In this subsection, we state that the utilization leveling problem is solvable in polynomial time.

Our optimization problem of allocating workload of a department such that the working times of workers who belong to the department are leveled as well as possible was described in Section 3.4 and modeled in Section 4.4. The corresponding decision problem asks whether there is a feasible allocation of departmental workload such that the sum of pairwise absolute differences in working times is not greater than b, $b \in \mathbb{N} \setminus \{0\}$.

Theorem 4.4 *The decision problem of our leveling problem is in* P*.* □

Proof The decision problem of our leveling problem can be solved in polynomial time. A polynomial-time algorithm that solves the optimization problem in $O(K^2)$ time is presented in Section 6.1.2. ■

Corollary 4.6 *The optimization version of our leveling problem can be solved in polynomial time.* □

4.6.5 Summary of results

Table 4.1 summarizes the main results of the complexity analysis that we conducted for our three problems.

Table 4.1: Main results of the complexity analysis for the three problems considered in this thesis

Problem	Complexity
Project selection	NP-hard in the strong sense
Workforce assignment	NP-hard in the strong sense
Utilization leveling	Solvable in polynomial time