Drexl, Andreas; Mundschenk, Martin

Long-term staffing based on qualification profiles

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 592

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 592

Long-term staffing based on qualification profiles

Andreas Drexl, Martin Mundschenk

Working Paper, May 2005

Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität, Kiel, Germany,
andreas.drexl@bwl.uni-kiel.de, martin@mundschenk.de

Abstract

Manpower still is one of the most expensive resources, in spite of increasing automation. While employee scheduling and rostering has been the topic of extensive research over the past decades, usually it is assumed that the demand for staff is either given or can be obtained without difficulty. In this research we close the gap between practical needs and available models and methods. In particular, we provide an integer programming model for long-term staffing decisions. The model is based on qualification profiles, the number of which grows exponentially in terms of the number of processes considered. In order to compute tight lower bounds we provide a column generation technique. The subproblem is a shortest path problem in a network where the arcs have multiple weights. Upper bounds, that is, feasible solutions are calculated my means of local search. We present computational results for randomly generated instances and empirical results for examples from practice. From these results it is evident that the bounds are tight and that substantial cost savings can be achieved.

Keywords: Work force, planning/staffing, qualification profile, column generation, local search, computational/empirical evaluation

1 Introduction

Effectiveness in the use of manpower resources is often the crucial advantage in a company's long-term success over its competitors (see MacBeath 1966). Even though employee scheduling has been addressed by personnel managers, operations researchers and computer scientists for more than 40 years (see Burke et al. 2004), the rostering literature assumes that the demand for staff is either given or can be obtained without difficulty (see Ernst et al. 2004). Work force planning is the highest level of personnel planning and determines the restrictions for lower levels like scheduling (see Wijngaard 1983). Most of the literature spent on this issue deals with special branches, especially the service sector, where demand for shifts and schedules is highly constrained (see the variety of models, algorithms and applications in, e.g, Pinedo 2005). On the other hand, short-term scheduling procedures affect the level of staffing that should be provided (see Abernathy 1973). The literature which deals with such interdependencies provides multi-stage integrated models for staffing and rostering. Unfortunately, however, only the size of the work force needed is considered while the skills required by the individual jobholder are usually neglected (see Abernathy 1973, Burns and Carter 1985).

Generally speaking, in the domain of manpower decision research, three major areas can be distinguished: staffing, scheduling and reallocation (see Warner 1976). Usually the time horizon of staffing covers several months to years, while scheduling/rostering faces several weeks. Finally, the time horizon of reallocation is hours to days.

The staffing can again be decomposed into two stages: determination of temporal and total manpower requirements (see Tien and Kamiyama 1982). These belong to the set of strategic or long-term decisions made by the management. Scheduling or rostering is the process of constructing work timetables for the staff so that an organization can satisfy the demand for its goods or services (see Ernst et al. 2004) as well as legal limitations and the convenience of the individual. The last stage is to reallocate the staff on shift to the jobs to be processed.

Wijngaard (1983) points out two extreme cases of organizations. One where each employee can do all kinds of jobs, where vacancies can be filled directly by recruitment and where

firing is easily possible. The other extreme is the case of an organization with very specific functions and employees with low mobility which makes it difficult to fill vacancies directly by recruitment. In the first case all of the work force must be qualified at the same (high) level causing the overall cost of employment to be high. In the latter case, the work force is much more heterogeneous so that low level activities can be processed by low qualified and cheaper jobholders, causing the overall costs of employment to be at a lower level. In this case the appropriate planning process is much more complex.

The optimal manpower structure for a specific company is determined by its economic environment. In a dynamic environment, the company's work force has to be flexible. In a static environment, the individual jobholder can be specialized to handle only certain activities.

In this paper we propose a general model which can be applied to a wide area of manufacturing and service organizations to determine simultaneously the size and the qualification of the work force. We develop a mathematical model to determine the number of personnel and the required individual skills, in order to meet predicted requirements with the goal of minimizing the total cost of employment.

The outline of the work is as follows: The problem setting is detailled in section 2. In section 3 we formulate the long-term staffing problem as an optimization model. Section 4 is dedicated to the development of algorithms for the computation of lower and upper bounds. The aim of section 5 is to evaluate the model and the methods developed by computational experiments and by empirical results. In section 6 we discuss some variants and extensions. Finally, section 7 gives a summary and some hints on future research.

2 Problem description

The basic model for manpower planning was developed jointly with a German printing company with focus on the industrial workers of the printing branch. Nevertheless the model might be transfered to different industries without much modification.

Consider a goods producing company, divided into a number of departments. The employees work on a one shift schedule, five days a week. Within one department, the jobholders are qualified all on a similar level, handling heterogeneous tasks. There are no specialists and there are no assistants. This common scenario leads to the case, that jobholders may get overstrained by handling uncommon, infrequent tasks, resulting in longer processing times. On the other hand, jobholders have to "waste" their time with assistant tasks like filing, packing or doing errands. Now imagine the aggregated time of all assistant tasks is enough to justify the employment of a full time assistant. The accruing tasks could be reallocated, so that one regular employee can be replaced with a cheaper one, doing only assistant tasks. Furthermore the remaining regular jobholders do not have to have the same high level of qualification. It might pay to have a few specialists, handling uncommon hard tasks, when they occur but performing the same tasks as the regular jobholders otherwise. To adjust the qualification of the work force to the requirements broken down to a disaggregated level within a workflow, can reduce the overall costs of the work force and even improve its performance.

The models' focus lies on the determination of the number and qualification of jobholders who are handling operational tasks on a day to day basis. The determination of the number and

qualification of managers is not considered within this model, because their tasks are more complicated and their wages are arranged freely with the employer. The payment of jobholders is divided into a number of salary levels, depending on the qualification of the jobholder.

In a manufacturing-to-order company, a job is started by the order of a customer. Each job consists of a number of processes, from incoming order to delivery. These processes are handled in various tasks within different cost centers by jobholders. There are tasks requiring a high level of qualification and there are ones that can be handled by jobholders with low qualification. Since higher qualified jobholders go with higher costs for the company, the cost minimal set and number of qualification profiles has to be determined.

Because of heterogeneous orders, processing times of a process vary from job to job. To obtain the overall time demand of each task within the considered time horizon, the aggregated processing time of every process for all jobs has to be calculated.

To determine the future demand of work force for a time horizon of for example one year, the demand of cumulated processing time for each process has to be predicted by using past data and by estimating marked demand for the next period. This data delivers the basis for our model to determine the work force, so that the time demand of a process is being covered by the time supply of a certain amount of jobholders, well qualified to handle all processes. Furthermore, there must be a minimum number of jobholders to be able to handle a certain process for backup reasons (e.g. vacation or disease) or to handle peak times which can result from short term reallocation.

The different processes, a job has to run trough until its delivery, are being handled by jobholders belonging to different occupational groups. Tasks like job accounting or acceptance of order are being handled by mercantile employees, while typesetting and graphical editing are handled by type setters. Within each occupational group there are grades of qualification going hand in hand with the money a jobholder earns. A mercantile employee who is only charged with filing documents is less qualified than someone in charge to purchase raw materials. The latter though is also able to handle tasks of lower qualification.

Let us take a look at the example in table 1 with two occupational groups A and B where each of them can be charged with a high and a low qualified task i to handle the related process. A high qualified employee always has the ability to handle low qualified tasks within the same occupational group. A qualification profile contains the binary information if a worker is qualified to handle a specific process or not. Overall in case of m processes 2^m qualification profiles do exist including the case that no qualification is available at all. Someone who is only charged to handle task $i = 1$, equivalent to the qualification profile (1,0,0,0), earns a wage of 6 monetary units. Someone who is only charged to handle task $i = 2$ (0,1,0,0) earns a wage of 10. But someone who is charged to handle task $i = 1$ and $i = 2$ (1,1,0,0) also earns a wage of 10. Task $i = 2$ is superior in terms of qualification to task $i = 1$. Table 2 gives an overview of all the different qualification profiles of this example and the associated wages.

If in this model someone is charged to perform tasks from more than one occupational group, he is payed the sum of the wages of the tasks. A jobholder who is, e.g., charged to handle the processes $i = 1$ and $i = 4$ (1,0,0,1) would cost 17. Even though that doesn't seem to make sense in the first place, it penalizes certain qualification profiles and prevents the model to generate profiles where for example type setters are charged with the purchase of raw materials. Nevertheless, such profiles are not excluded explicitely but they will not be generated by the model (see section 5.2).

Occupational group	task i	qualification	wage
A	1	low	6
	2	high	10
B	3	low	5
	4	high	11

Table 1: Qualification level and wages

Qualification profile	wage
(1,1,1,0)	15
(0,1,1,0)	15
(1,0,1,0)	11
(0,0,1,0)	5
(1,1,0,0)	10
(0,1,0,0)	10
(1,0,0,0)	6
(1,1,1,1)	21
(0,1,1,1)	21
(1,0,1,1)	17
(0,0,1,1)	11
(1,1,0,1)	21
(0,1,0,1)	21
(1,0,0,1)	17
(0,0,0,1)	11

Table 2: Qualification profiles and wages

3 Optimization model

In the following first we consider the simplified case that all jobholders have the same annual working time, that is, only full-time workers do exist. The more general case of full- and part-time workers is considered in section 6.1.

Using the parameters

n : number of jobs j
m : number of processes i
p_{ji} : processing time of process i for job j
q : number of qualification profiles h
c_h : cost of employee with qualification profile h
b : annual working time of a jobholder
S_i : minimum number of jobholders required to handle process i

$$v_{ih} = \begin{cases} 1 & \text{if employee with qualification profile } h \text{ can handle process } i \\ 0 & \text{otherwise} \end{cases}$$

and decision variables

x_h : number of full-time employees with qualification profile h

we get the following optimization model:

$$\min \quad \sum_{h=1}^{q} c_h x_h \tag{1}$$

$$\text{s.t.} \quad \sum_{h=1}^{q} v_{ih} \frac{b}{\sum_{j=1}^{m} v_{jh}} x_h \geq \sum_{j=1}^{n} p_{ji} \qquad i = 1, \ldots, m \tag{2}$$

$$\sum_{h=1}^{q} v_{ih} x_h \geq S_i \qquad i = 1, \ldots, m \tag{3}$$

$$x_h \geq 0 \text{ and integer} \qquad h = 1, \ldots, q \tag{4}$$

Objective function (1) minimizes the total annual cost of the work force. Constraints (2) assure that the available working time for each process i meets at least the demand of working time of process i for all jobs. Of course, p_{ji} and b have to be measued in the same time units. Constraint (3) assures that there is a minimum number S_i of jobholders being able to handle process i for backup reasons as outlined above. Note that high values for S_i make the manpower on-hand more flexible, since more jobholders will be qualified to handle process i. Low values lead to low flexibility of the jobholders but also to lower overall costs.

The model (1) to (4), also called integer master problem, has an exponential number of columns, hence, there is no chance to solve it directly. In order to cope with this fact, we use column generation (see Gilmore and Gomory 1960) in order to solve the linar programming relaxation to optimality. This gives us a lower bound. The outcome of column generation then is used in order to compute feasible solutions, i.e., upper bounds for the optimal objective function value as well.

4 Algorithms

First we describe in section 4.1 how to compute a lower bound for the optimal objective function value. Here, the structure of the subproblem is of special interest; see section 4.2. In section 4.3 we show how to tighten the lower bound. Section 4.4 details how to come up with feasible solutions, that is, upper bounds. The algorithms are illustrated by means of an example in section 4.5.

4.1 Lower bound

In order to compute a lower bound of the integer staffing problem introduced above, we replace the integrality requirements (4) through $x_h \geq 1$ for all h. Starting with an initial set of columns, say q, we have to cheque whether a column does exist which lowers the objective function (1). Hence, we look at the dual (5) to (8) of the linear programming relaxation of the integer master problem.

$$\max \quad \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ji}\pi_i + \sum_{i=1}^{m} S_i\tau_i \tag{5}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} v_{ih}\frac{b}{\sum_{j=1}^{m} v_{jh}}\pi_i + \sum_{i=1}^{m} v_{ih}\tau_i \leq c_h \qquad h = 1,\ldots,q \tag{6}$$

$$\pi_i \geq 0 \qquad i = 1,\ldots,m \tag{7}$$

$$\tau_i \geq 0 \qquad i = 1,\ldots,m \tag{8}$$

Using the dual variables $\pi_i \geq 0$ associated with constraint (2) and $\tau_i \geq 0$ associated with constraint (3) we try to identify a column $q+1$ for which

$$c_h < \sum_{i=1}^{m} v_{i,q+1}\frac{b}{\sum_{j=1}^{m} v_{j,q+1}}\pi_i + \sum_{i=1}^{m} v_{i,q+1}\tau_i \tag{9}$$

is valid (i.e. a dual constraint (6) is violated). If such a column exists it has to be added to the primal (that is, it prices out attractively). If no such column exists we can stop.

In order to cheque this condition we have to solve the following optimization problem:

$$\max\left\{\sum_{i=1}^{m} v_{i,q+1}\frac{b}{\sum_{j=1}^{m} v_{j,q+1}}\pi_i + \sum_{i=1}^{m} v_{i,q+1}\tau_i : (v_{i,q+1}) \text{ is a qualification profile}\right\} \tag{10}$$

In the following section we will show that this optimization problem can be solved in polynomial time by means of iterative shortest path computations.

4.2 Subproblem

The set of qualification profiles can be transferred into a complete, directed network $G = (V, E, w)$ with node set V, arc set E, and arc weights w, respectively. Every node stands for the qualification to handle a specific process. Moreover, a source and a sink node have to be added and, hence, overall we have $m+2$ nodes. The nodes are labeled uniquely such that node 0 is the source node, node $m+1$ is the sink node and nodes 1 to m correspond to the m processes. The arc e_{ki} between node k and node i represents the option of combining processes within one qualification profile. The weight w_{ki} of arc e_{ki} is the extra payment needed. Figure 1 illustrates the structure of the network for the example introduced above.

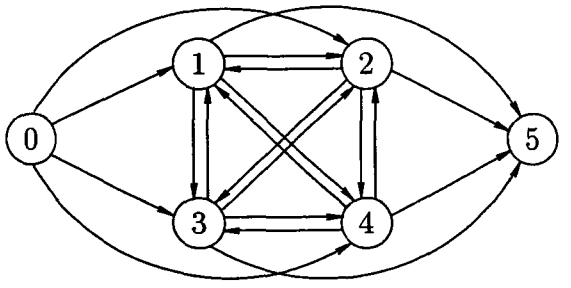

Figure 1: Network topology

$$\begin{pmatrix} 6 & 10 & 5 & 11 & - \\ - & 4 & 5 & 11 & 0 \\ 0 & - & 5 & 11 & 0 \\ 6 & 10 & - & 6 & 0 \\ 6 & 10 & & - & 0 \end{pmatrix}$$

Table 3: Arc weights $(w_{ki})_{k=0,\ldots,4}^{i=1,\ldots,5}$

The set of arcs E consists of four types of arcs $E = E^1 \cup E^2 \cup E^3 \cup E^4$ which are defined as follows (see table 3 also):

- Arcs E^1 from the source node 0 to every node i. The weight w_{0i} of these arcs is the wage for the qualification to handle process i.
- Arcs E^2 from every node i to the sink node $m+1$. The weight of these arcs is $w_{i,m+1} = 0$.
- Arcs E^3 connecting node k with node i within the same professional group. The weight w_{ki} of these arcs equals 0, if the qualification to handle process k is higher than the one to handle processs i. Otherwise the weight w_{ki} is the difference of the wage for the qualification to handle process i and the wage for the qualification to handle process k.
- Arcs E^4 connecting node k with node i not belonging to the same professional group. The weight w_{ki} of these arcs is the wage for the qualification to handle process i.

The total wage is equal to the shortest path from the source to the sink via all nodes that are supposed to be within a certain qualification profile. All in all there are $m \cdot (m-1) + 2m$ arcs. The costs c_h of an employee qualified to handle, e.g., processes 1 and 2, that is the profile (1,1,0,0), is determined by the shortest path from the source to the sink via nodes 1 and 2:

$$c_h = w_{0,1} + w_{1,2} + w_{2,5} = 6 + 4 + 0 = 10$$

The network is designed that way, that there are multiple paths from the source to the sink. By setting up rules for the network topology, half of the arcs connecting the process nodes can be eliminated.

Rule 1 *(labeling of nodes within occupational groups)*
Within one occupational group, the nodes have to be ordered by the level of qualification. This means that node $i+1$ is associated to a process $i+1$ that needs a higher level of qualification than process i.

Rule 2 *(labeling of occupational groups)*
The occupational groups can be in arbitrary order as long as the processes associated to these groups satisfy rule 1.

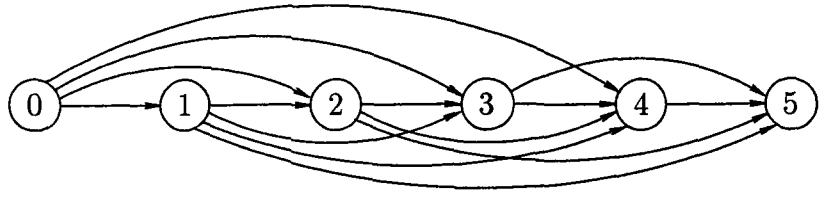

Figure 2: Reduced network

After applying these rules, only the upper triangle of the matrix (w_{ki}) is needed, reducing the network G to $G' = (V, E', w')$ with $E' \subset E$ as shown in Figure 2. For notational simplicity we will omit the prime, i.e. keep the notation $G = (V, E, w)$ for the reduced network also.

In order to compute a lower bound, first of all we have to define an initial set of colums, such that a feasible solution can be achieved. In our case this can easily be done by using the identity matrix.

Then the network is set up to generate column $q+1$ for the master problem. To this end the weights w_{ki} of the arcs are updated using the dual variables π_i and τ_i of the master problem according to

$$\bar{w}_{ki} = w_{ki} - \frac{b}{\sum_{j=1}^{m} v_{j,q+1}}\pi_i - \tau_i.$$

Since the weights $\bar{w}_{ki}$ of the network are a function of $\sum_{j=1}^{m} v_{j,q+1}$ we now have to deal with multiple arc weights. With every node included in the path from the source to the sink, the weights of all arcs change. Referring to figure 2, the weight of the arcs of a path from the source to the sink via one node only, would be $\bar{w}_{ki} = w_{ki} - \frac{b}{1}\pi_i - \tau_i$. The weight of the arcs of a path including two nodes would be $\bar{w}_{ki} = w_{ki} - \frac{b}{2}\pi_i - \tau_i$ and so on. But since the number of nodes included in the shortest path from the source to the sink in G is not known beforehand, we have to deal with the case that each arc has m weights. Equivalently we may consider m networks $G^z = (V, E, \bar{w}^z), z = 1, \ldots, m$, where the weight of each arc is calculated for a predefined number of nodes, that is,

$$\bar{w}_{ki}^z = w_{ki} - \frac{b}{z}\pi_i - \tau_i. \tag{11}$$

For each of these $z = 1, \ldots, m$ networks Q^z, the shortest path from the source to the sink, containing a maximum of z nodes, has to be determined by using the Bellman-Ford algorithm (see, e.g., Minieka 1978), a modification of the Dijkstra algorithm. This algorithm computes the shortest path between a pair of nodes covering at most z nodes. Note that we do not count the source and the sink node (which are contained in the path anyway) and, hence, a maximum of z nodes is equivalent to a maximum of $z+1$ arcs. If the number of nodes contained in the shortest path from the source to the sink in network G^z is smaller than z, then the network is discarded, because the weights of the arcs are not consistent with the number of nodes, contained in that path. More precisely, only paths according to the following definition have to be considered.

Definition 1 *(valid path)*
Consider the network G^z with weights $\bar{w}_{ki}^z = w_{ki} - \frac{b}{z}\pi_i - \tau_i$. Then a source-to-sink path in the network is valid only, if it covers $z+1$ arcs.

The following procedure, denoted as `BestPath`, has to be performed in order to determine the overall shortest path.

set $bestpath = \infty$
for $z = 1, \ldots, m$ do
  calculate weights $\bar{w}_{ki}^z = w_{ki} - \frac{b}{z}\pi_i - \tau_i$ for all arcs of the network
  compute shortest path, denoted as $path$, from source to sink covering at most $z+1$ arcs
  if number of arcs in $path$ is less than $z+1$ then stop endif
  if length of $path <$ $bestpath$ then
    $bestpath = path$
    store nodes included in $path$ in $(v_{i,q+1})$ with
    $v_{i,q+1} = 1$, if node i is contained in $path$ $(0, \text{otherwise})$
  endif
endfor

The costs c_h of a new profile are calculated using the weights of the initial graph $G = (V, E, w)$. If inequality (9) is valid a new column defined by $(v_{i,q+1})$ has to be added to the master problem. Otherwise the pricing criterion is met and the column generation process terminates with LB_1 as valid lower bound.

The proof, that the algorithm works correctly, that is, calculates the overall shortest path, is given below.

Proposition 2 *Consider a network G^z with arc weights $\bar{w}^z$. Then the weight $\bar{w}^z_{ki}$ of arc e_{ki} is monotonically increasing in z.*

Proof: Let $\bar{w}^z_{ki}$ denote the weight of arc e_{ki} in network G^z. Then the inequality

$$w_{ki} - \frac{b}{z}\pi_i - \tau_i = \bar{w}^z_{ki} \leq \bar{w}^{z+1}_{ki} = w_{ki} - \frac{b}{z+1}\pi_i - \tau_i \tag{12}$$

is valid. □

Theorem 3 *Consider networtk G^z in iteration z of the algorithm. It is sufficient to compute the shortest path containing at most $z+1$ arcs in the network G^z in order to find the overall shortest path in G.*

Proof: Let L^z_λ denote the length of the shortest source-to-sink path in the network G^z containing exactly λ nodes. Then the inequality

$$L^z_z \geq L^z_{z-1} \geq L^{z-1}_{z-1} \tag{13}$$

is valid because of the monotonicity property (12). Hence, if the shortest path in G^z is not a valid path according to definition 1, it cannot be the overall shortest path in G. □

Corollary 4 *The algorithm can be stopped prematurely, that is, before z equals m, if the number of arcs in path is less than $z+1$.*

Proof: Follows immediately from proposition 2 and theorem 3. □

4.3 Improved lower bound

The lower bound LB_1 computed by means of the column generation technique described in the previous section can be improved easily. Let (x^{LP}_h) denote the optimal fractional variables of the linear programming relaxation of the integer master problem (1) to (4). These variables represent the non-integer number of jobholders, needed for qualification profile h. Then

$$\Phi = \sum_{h=1}^{q} x^{\mathsf{LP}}_h \tag{14}$$

defines the sum of the non-integral amount of work force needed, to perform all processes within the considered period of time. Hence, the minimum work force size of the integer model must be at least $\lceil \Phi \rceil$. The costs of the difference $\lceil \Phi \rceil - \Phi$ can be evaluated with the minimum costs, possible to accrue for the missing work force. So the lower bound LB_1 can be tightened to

$$LB_2 = LB_1 + (\lceil \Phi \rceil - \Phi) \cdot \min_{h=1}^{q} c_h$$

where q is the number of columns generated and c_h is the cost of column h.

4.4 Upper bounds

Upon termination of column generation we have a valid lower bound and a fractional solution (x_h^{LP}). An integral, feasible solution (x_h^{IP}) and a valid upper bound UB_1 can easily be computed by means of solving the mixed-integer program which is defined through the columns generated. Of course, it might be too time consuming to solve this mixed-integer program to optimality. Fortunately, a feasible solution is sufficient and, hence, computation can be aborted after a certain amount of time.

The gap between UB_1 and LB_2 usually allows much improvement. This is because the columns generated to solve the relaxed master problem might not be suitable in order to achieve a good integer solution. We use the solution corresponding to the upper bound UB_1 as initial solution for local search, producing a second upper bound denoted as UB_2. Among the variety of available local search algorithms (see, e.g., Aarts and Lenstra 1997) we decided to use simulated annealing in order to improve the upper bound.

For each qualification profile h a number of x_h^{IP} jobholders have to be employed and the total number of manpower needed to execute all processes is $u = \sum_{h=1}^{q} x_h^{\mathsf{IP}}$. Now we construct a matrix $A = (a_{ij})_{i=1,\ldots,m}^{j=1,\ldots,u}$. For every $x_h^{\mathsf{IP}} > 0$, we add x_h^{IP} columns to A, each containing the binary entries of the corresponding qualification profile (v_{ih}). Apparently, matrix $A = (a_{ij})$, or a for short, represents a feasible solution of the integer staffing problem.

Simulated annealing (SA) is a generic probabilistic heuristic approach originally proposed in Kirkpatrick et al. (1983) and Kirkpatrick (1984) for global optimization. Usually, SA locates a "good" approximation of the global optimum of a given objective function F in a large search space. At each iteration, SA considers some neighbors of the current solution a, and probabilistically chooses either to accept a new solution a' or keeping a. The probabilities are chosen so that the problem ultimately tends to move to solutions with better objective function value. Typically this process is repeated until a solution which is "good enough" has been determined, or until a given time limit has been reached.

SA uses several basic concepts:

- *Neighborhood concept*

  At each iteration μ the neighborhood $N(a)$ of solution a is specified (usually problem-specific). $N(a)$ represents the subset of solutions a' which can be reached in one iteration emanating from a. Generally, it is not possible to store the neighborhood structure, because the set of feasible solutions has an exponential size.

- *Probabilistic acceptance of a new neighborhood solution*

  In each iteration μ solution a' is accepted with probability

$$P(a, a', T_\mu) = \min\left\{1, exp\left(-\frac{F(a') - F(a)}{T_\mu}\right)\right\}.$$

  This probability is decreased during the course of the algorithm. In other words, we can escape a local optimum, but the probability for doing so is low after a large number of iterations. It guarantees that there is a high probability to locate a global optimum while avoiding to be trapped in local optima.

- *Parameter (temperature) dependent acceptance probability*

  The probability of making the transition to the new solution depends on a global time-varying parameter T called the annealing temperature: $(T_\mu)_{\mu=1}^{\infty}$ is a sequence of positive control parameters with $\lim_{\mu\to\infty} T_\mu = 0$.

- *Cooling schedule*

  Generally, the sequence $(T_\mu)_{\mu=1}^{\infty}$ is created by a function g, i.e. $T_{\mu+1} = g(T_\mu)$ for all μ. The initial annealing temperature T_0 has to be defined in advance.

- *Termination criterion*

  One has the freedom to introduce different stopping criteria. Typically, SA is repeated until the system reaches a state which is "good enough", or until a given time limit has been reached. The annealing temperature decreases to (nearly) zero short before termination.

In order to apply SA to a particular problem, we must specify the search space, the neighborhood search moves, the acceptance probability function, the cooling schedule and the termination criterion. These choices can significantly affect the method's effectiveness. Unfortunately, there is no unique choice that will be good for all problems, and there is no general way to find the best choice for a given problem (see, e.g., van Laarhoven and Aarts 1987, Johnson et al. 1989, 1991).

The local search is performed by swapping entries of the matrix (a_{ij}) from 1 to 0 or vice versa, following a set of rules. That is, the search mechanism is to examine members of the swap neighborhood of the starting solution a, and then move to neighbor a' and so on. To be more precise, let A_i and A^j denote row i and column j of matrix A, respectively.

Rule 3 *(row rule)*
Given row A_i a move is defined by chosing an entry a_{ij} $(j = 1, \ldots, u)$ at random and flipping it from 1 to 0 or vice versa.

Rule 4 *(column rule)*
Given column A^j a move is defined by chosing an entry a_{ij} $(i = 1, \ldots, m)$ at random and flipping it from 1 to 0 or vice versa.

We have modified the general purpose SA algorithm described in Abramson et al. (1996) to match the needs of our special case. To be more precise we have implemented the following algorithm. First, we scan the rows $i = 1, \ldots, m$ of matrix A cyclically in this order until no neighbor has been accepted within a certain number of trials. Given row A_i rule 3 is applied a certain number of times. If a neighboring solution a' is feasible and improves the current best known upper bound or is accepted probabilistically we go to the next row. Afterwards, we scan the columns $j = 1, \ldots, u$ of matrix A cyclically in this order until no neighbor has been accepted within a certain number of trials. Given column A^j rule 4 is applied a certain number of times. Again, if a neighboring solution a' is feasible and improves the current best known upper bound or is accepted probabilistically we go to the next column.

According to preliminary computational tests not further documented here the initial temperature to start SA was set to $T_0 = 200$ and the cooling schedule is $T_\mu = 0,98^\mu \cdot T_0$. The algorithm terminates after one million swaps or if the solution gap is less than 2%.

SA terminates with a as the best feasible solution found. The objective value associated with a is denoted as UB_2. According to Abramson et al. (1996), we also implemented a function to restore feasibility of a' for the case that a move induces infeasibility. Because of the computational burden involved, the overall performance of SA worsened in terms of computation time and we did not keep this implementation.

4.5 Illustrative example

Consider a case with 50 jobs and 20 processes each job has to go through upon delivery. The basic data for this instance are provided in table 4. Column one identifies the process number i, column two shows the aggregated time demand P_i to process all $n = 50$ jobs, column three gives the minimum number S_i of jobholders needed to handle process i, column four identifies the occupational group each process belongs to where in this example we assume that processes 1 to 7 are being handled by commercial clerks (group 1), processes 8 to 13 by creative workers (group 2) and processes 14 to 20 by production workers (group 3). Finally, column five provides the wage for each process. Note that the level of qualification to handle process i and the associated wage are positively correlated. Finally, in this example the annual working time of each jobholder is set to $b = 70\ 000$.

In case of $m = 20$ processes, $2^{20} - 1 = 1\ 048\ 575$ different qualification profiles do exist. Column generation delivers the columns to obtain the optimal solution of the linear programming relaxation of (1) to (4). Table 6 gives an overview of the column generation process. An initial set of 10 columns has been constructed as outlined above. Column one of the table identifies the latest column considered, column two shows the corresponding optimal objective function value (OFV), column three provides the profile associated to the most recent column, column four gives the cost of the profile, column five tells us how attractive the column is in terms of the weights $\bar{w}_{ki}$ calculated according to equation (11) and the last column shows the corresponding cost difference.

Overall we can see that only 58 columns have to be considered and that the optimal objective function value of the linear programming relaxation is $LB_1 = 1\ 0881.98$. The minimum number of jobholders needed to handle all processes is $\Phi = \sum_{h=1}^{58} x_h = 8.488229$. Hence, the minimum integer size of the work force is $\lceil \Phi \rceil = 9$ and LB_1 can be improved as follows:

$$LB_2 = LB_1 + (\lceil \Phi \rceil - \Phi) \cdot \min_{h=1}^{58} c_h = 10\ 881.98 + (9 - 8.48) \cdot 600 = 11\ 189.05$$

If we solve the reduced integer master problem with the 58 columns computed before, an upper bound $UB_1 = 14\ 910$ is obtained. This implies that the solution gap is $\frac{UB_1 - LB_2}{LB_2} = 0.33$. The upper bound UB_1 corresponds to the solution shown in table 7. Thus 12 jobholders have to be employed with total costs of 14 910.

Now we use the feasible solution corresponding to the upper bound UB_1 as initial solution for the local search algorithm. The number of manpower needed is $u = \sum_{h=1}^{58} x_h^{IP} = 12$. The matrix A with dimension (20×12) is given in table 5. Each column of A represents a

i	P_i	S_i	group	wage
1	30 149	1	1	600
2	29 365	2	1	600
3	30 841	1	1	600
4	28 494	1	1	800
5	29 801	1	1	900
6	28 937	3	1	1 000
7	28 305	1	1	1 200
8	29 601	1	2	840
9	30 517	1	2	840
10	29 091	2	2	1 080
11	28 744	3	2	1 200
12	29 994	1	2	1 200
13	31 052	1	2	1 320
14	28 039	1	3	780
15	29 864	2	3	1 040
16	29 939	1	3	1 040
17	33 677	1	3	1 040
18	29 326	1	3	1 040
19	30 014	2	3	1 300
20	28 426	2	3	1 430

Table 4: Illustrative example

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Table 5: Matrix $A = (a_{ij})_{i=1,\ldots,m}^{j=1,\ldots,u}$

#	OFV	profile	c_h	$\bar{c}_h$	$\bar{c}_h - c_h$
11	20 880.00	01100110011010101010	3 820.00	12 330.00	8 510.00
12	19 780.00	10010101111001100101	3 630.00	12 330.00	8 700.00
13	17 600.45	00000001000000000000	840.00	4 080.00	3 240.00
14	17 511.01	00000000000010000000	1 320.00	4 200.00	2 880.00
15	17 413.23	01101110101001011011	3 830.00	6 266.67	2 436.66
16	16 441.24	00000000000001000000	780.00	2 880.00	2 100.00
17	15 842.00	00000010000000000000	1 200.00	3 240.00	2 040.00
18	15 264.74	10010101111101011011	3 630.00	5 063.84	1 433.83
⋮	⋮	⋮	⋮	⋮	⋮
51	11 060.47	00000001111100000000	1 200.00	1 268.87	68.86
52	10 931.66	00000000111110000000	1 320.00	1 446.00	125.99
53	10 928.97	10110110000000000000	1 200.00	1 268.57	68.57
54	10 923.13	01000000000000000000	600.00	660.00	60.00
55	10 920.61	00000000000000111111	1 430.00	1 464.67	34.66
56	10 884.82	01100110000000000000	1 200.00	1 228.57	28.57
57	10 883.34	11000110000000000000	1 200.00	1 228.30	28.30
58	10 881.98	00001110000000000000	1 200.00	1 200.00	0.00

Table 6: Columns generated

x_h^{IP}	profile	cost
1	10101110000000000000	1 200
1	00000000000000001010	1 300
3	00000000000001111111	1 430
1	11010100000000000000	1 000
1	00000000001010000000	1 320
1	10011100000000000000	1 000
3	00000001111100000000	1 200
1	01100110000000000000	1 200
12		14 910

Table 7: Feasible solution – UB_1

#	profile	cost
1	11100000000000000000	600
2	00000000000000011111	1 430
3	00000000000000100111	1 430
4	00000000000001000000	780
5	01111100000000000000	1 000
6	00000000101110000000	1 320
7	00000001111100000000	1 200
8	10011100000000000000	1 000
9	00000000000000111000	1 040
10	00000001011010000000	1 320
11	00000110000000000000	1 200
		12 320

Table 8: Improved feasible solution – UB_2

qualification profile, that is, the entries $(a_{ij})_{i=1,\ldots,m}$ correspond to the parameter $(v_{ih})_{i=1,\ldots,m}$ computed during column generation. In particular we can see that the two rows 3 and 7 of table 7 imply the three identical profiles (columns) 3 to 5 and 9 to 11 of table 5. The cost vector associated with the $u = 12$ profiles is (1 200, 1 300, 1 430, 1 430, 1 430, 1 000, 1 320, 1 000, 1 200, 1 200, 1 200, 1 200).

After applying SA, we obtain an improved solution, shown in table 8. The improved upper bound now is $UB_2 = 12\ 320$ leading to a solution gap of $\frac{UB_2 - LB_2}{LB_2} = 0.101076$. The total number of manpower needed is reduced to 11. The total cost of the work force is 12 320.

5 Evaluation

Generally, two possible approaches can be found adopted in literature when having to come up with test instances for benchmarking purposes. First, practical cases. Their strength is their high practical relevance while the obvious drawback is the absence of any systematic structure allowing to infer any general properties. Thus, even if an algorithm performs good on some practice cases, it is not guaranteed that it will continue to do so on other instances as well. Second, artificial instances. Since they are generated randomly according to predefined specifications, their plus lies in the fact that fitting them to certain requirements such as given probability distributions poses no problems. However, they may reflect situations with little or no resemblance to any problem setting of practical interest. Hence, an algorithm performing well on several such artificial instances may or may not perform satisfactorily in practice.

Therefore, we decided to devise a combination of both approaches, thereby attempting to keep their strengths while avoiding their drawbacks. In order to do so we present in section 5.1 the results of a computational study based on artificial instances while section 5.2 is dedicated to a practice case.

5.1 Computational study

We used two different subsets of artificial instances:

- Subset 1: Instances for which the optimal solution is readily available. Such instances are used to evaluate the quality of the lower and the upper bounds.
- Subset 2: A set of instances, generated at random by means of an instance generator.

Subset 1: Instances with known optimal solution Instances belonging to subset 1 are constructed as follows: Let α denote the number of different occupational groups. Moreover, let ω denote the number of processes associated to each occupational group. Then the total number of processes equals $m = \alpha \cdot \omega$. Assume that the minimum number S_i of jobholders required for process i equals 1. Furthermore, assume that the aggregated processing time $P_i = \sum_{j=1}^{n} p_{ji}$ is equal to Φ (used for computing the lower bound LB_2). Finally, assume that the wages are monotonically increasing with the "group" index $h = 1, \ldots, \omega$.

The example given in table 9 with $\alpha = 2$, $\omega = 4$ and $P_i = 20$ for all i serves for illustrating purposes. If we assume that the annual working time is $b = 40$, then the optimal solution for this example is given in table 10 with $UB_1 = UB_2 = LB = 12$.

Occupational group	process	wage
1	1	1
	2	2
	3	3
	4	4
2	5	1
	6	2
	7	3
	8	4

Table 9: Instance with known optimal solution

Jobholder	profile	wage
1	11000000	2
2	00110000	4
3	00001100	2
4	00000011	4
		12

Table 10: Optimal solution

Subset 2: Instances generated at random Given the parameters m and n, processing times p_{ji} are drawn at random from the interval $[0, \ldots, 3]$. Measured in hours, this seems to be realistic from an empirical point of view. The parameter S_i is generated at random from the set $\{1, \ldots, S^{\max}\}$ with $S^{\max} = 4$.

We have implemented the models and algorithms in ANSI C. We used the open source LP/IP solver lp_solve (see [11]). The computations were performed on a PowerPC G4 with 1.67 GHz running Mac OSX.

Subset 1: Results In order to evaluate the quality of the lower and the upper bound the model was tested with various instances belonging to subset 1. The results are shown in table 11.

The first three columns of table 11 characterize each of the 11 instances tested in terms of the input used. Column four displays the optimal objective function value OPT. Column five says whether the solution corresponding to the lower bound already establishes an integral solution ($\text{int}_{\text{LB}} = \text{yes}$) or not ($\text{int}_{\text{LB}} = \text{no}$). Column six provides the upper bound UB_2. Finally, columns seven and eight display the upper and lower bound deviations $\text{GAP}_{\text{UB}} = \frac{UB_2 - \text{OPT}}{\text{OPT}}$ and $\text{GAP}_{\text{LB}} = \frac{\text{OPT} - LB_2}{LB_2}$, respectively. Based upon these results it is strongly conjectured that the lower bounds are tight.

α	ω	b	OPT	int_{LB}	UB_2	GAP_{UB}	GAP_{LB}
2	20	20 000	6 000	no	6 000	0.00	0.00
2	20	10 000	10 000	no	11 400	0.14	0.00
12	4	4 000	16 800	no	16 800	0.00	0.00
1	48	4 000	184 800	no	205 800	0.11	0.00
1	48	8 000	93 600	no	109 300	0.16	0.00
8	6	4 000	19 200	no	19 200	0.00	0.00
4	10	4 000	12 000	no	12 600	0.05	0.00
4	10	10 000	6 000	no	7 500	0.25	0.00
5	10	2 000	27 500	yes	27 500	0.00	0.00
5	10	4 000	15 000	no	16 700	0.11	0.00
5	10	10 000	7 500	no	8 500	0.13	0.00

Table 11: Results – subset 1 instances

Even though these instances can be solved at a glance, they are hard to solve for our simulated annealing algorithm, leading in some cases to broad gaps GAP_{UB}. With respect to the long-term nature of the problem and the inherent uncercainty involved, however, we think that the solution gap GAP_{UB} is acceptable. The hardness of instances tends to increase with increasing ratio $\frac{b}{P_i}$. That is, the more different processes a jobholder has to be qualified to handle, the harder it is to find the optimal solution.

Subset 2: Results We tested the model with different configurations of the parameters m and n on randomly generated instances. For each of the parameter configurations, ten instances were generated and solved and the average results were computed. Table 12 shows the results of 11 different parameter configurations.

m	n	# of jobholders	GAP	IMP_{UB}	T_{LB}	T_{UB}
12	3 000	21	0.094	0.013	0.007	16.199
15	2 000	27	0.040	0.008	0.014	21.498
15	3 000	26	0.105	0.014	0.014	30.061
15	4 000	25	0.091	0.016	0.009	26.662
20	2 000	37	0.055	0.007	0.044	37.494
20	4 000	35	0.096	0.018	0.041	79.461
20	6 000	36	0.078	0.246	0.040	159.655
30	2 000	60	0.062	0.350	0.249	159.695
30	3 000	57	0.100	0.558	0.214	268.343
40	3 000	77	0.106	2.002	0.941	623.487

Table 12: Average results – subset 2 instances

The first two columns give the number of processes m and the number of jobs n to be handled. With $b = 1\ 640\frac{\text{hours}}{\text{year}}$ the third column shows the average number of jobholders needed, to perform the accruing workload. The fourth column shows the average gap between UB_2 and LB_2 where $\text{GAP} = \frac{UB_2 - LB_2}{LB_2}$. In the next column one can see the average improvement between UB_1 and UB_2. It can be clearly seen, that UB_1, delivered by the mixed-integer programming solver, worsens with the size of the instance. In column six, the average CPU-time T_{LB} needed to compute the lower bound LB_2 is given. As can be seen in the last column, the average CPU-time T_{UB} to compute UB_2, takes almost all of the total computation time.

5.2 Empirical results

The approach was tested in cooperation with a medium-size printing company. The data was extracted out of a computer-based job tracking system (see [12]), providing the complete data from 1999 to 2004. We performed both a retrospective analysis comparing our results with past data and a prospective analysis by predicting future periods. In what follows we summarize the results obtained (for details see Mundschenk and Drexl 2005).

Comparison with past data In the considered company, there are 33 different cost centers, every job can run through until its completion. The jobholder scans the barcode attached to the job ticked coming with each job, to register the checkin time of that job in that cost center, into the job tracking system. The same procedure is done, when all tasks are performed within that cost center and the job has to be checked out. Since different kinds of processes are aggregated within each cost center, we cannot determine the requirements to the individual skills of the jobholders on the level of each single process. However, the data is still well enough to expect suitable results. An extract of the data is given in table 13.

year	n	m	annual workload (h)	cost (€)	# of jobholders	# of FTE
1999	3 667	33	85 723.88	3 276 664.95	132	91.43
2000	3 231	33	84 303.38	3 363 326.07	131	90.06
2001	3 052	33	85 064.68	2 884 708.38	110	89.71
2002	2 791	33	81 194.30	3 396 041.32	146	92.68
2003	2 318	33	70 852.68	2 853 577.81	119	80.97
2004	2 291	33	69 192.68	2 741 759.52	111	73.69

Table 13: Empirical data – 1999 to 2004

year	# of FTE	cost (€)	improvement	GAP
1999	58	2 214,625	32%	0.114
2000	56	2 152,561	36%	0.092
2001	59	2 143,392	26%	0.094
2002	56	2 055,859	39%	0.099
2003	51	1 828,938	36%	0.094
2004	50	1 775,936	35%	0.083

Table 14: Results – 1999 to 2004

The approach was applied to each of the annual data sets, extracted from the job tracking system, with the appropriate parameters. Aggregated results for the years 1999 to 2004 are displayed in table 14. By generating the appropriate qualification profiles, it can be seen, that the total annual workload (see table 13) could have been handled with about 60% of the work force. Thus applying the model lowers the total cost of the work force in the range of 26% – 39% where this percentage is defined as $\frac{\text{old cost}-\text{improved cost}}{\text{old cost}}$.

Work force size needed in the future The data for 2005 and 2006 have been determined using linear regression. The results are displayed in table 15. Since the total workload will be further on declining, the total size of the work force can further be reduced. By adjusting the qualification profile of each jobholder, the total cost of the work force can be reduced to about 1.7 million €.

Year	# of FTE	cost (€)	GAP
2005	49	1 806 299	0.112
2006	49	1 731 793	0.107

Table 15: Work force size and associated costs for 2005 and 2006

6 Variants and extensions

In section 6.1 we will show that the approach can be extended to cover the issue of part-time workers. Moreover, we will show how to determine individual working times based on the results obtained in section 6.2.

6.1 Part-time workers

In the following we will show that part-time workers can easily be covered by our methodology. Assume that a set K of contracts do exist which differ in the annual working time (the only aspect which is of interest here) and let b^k, $k \in K$, denote the time according to contract k. Let C_k denote the number of profiles available with respect to working time b^k. Then the extended model reads as follows:

$$\min \quad \sum_{k \in K} \sum_{h \in C_k} c_h x_h \tag{15}$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{h \in C_k} v_{ih} \frac{b^k}{\sum_{j=1}^{m} v_{jh}} x_h \geq \sum_{j=1}^{n} p_{ji} \qquad i = 1, \ldots, m \tag{16}$$

$$\sum_{k \in K} \sum_{h \in C_k} v_{ih} x_h \geq S_i \qquad i = 1, \ldots, m \tag{17}$$

$$x_h \geq 0 \text{ and integer} \qquad \forall\, k \in K, \forall\, h \in C_k \tag{18}$$

If we compare (1) to (4) with (15) to (18) we can see that the only difference is that in the latter case we have to take care of all profiles available for all the different working times. Note that the number of profiles available does not depend on the annual working time b^k and, hence, we have $|C_1| = |C_2| = \ldots = |C_{|K|}|$.

As a consequence we need to setup a network for every possible annual working time $k \in K$ in order to compute the lower bound LB_1 of the extended model similar to what has been described in section 4.2. Accordingly, in every iteration of the column generation process, at most $|K|$ columns are generated and added to the master problem, if the pricing criterion

$$c_h < \sum_{i=1}^{m} v_{i,q+1} \frac{b^k}{\sum_{j=1}^{m} v_{j,q+1}} \pi_i + \sum_{i=1}^{m} v_{i,q+1} \tau_i$$

holds for $k \in K$. The upper bounds UB_1 and UB_2 can be computed without any modification.

In order to illustrate this extension, we pickup the example from section 4.5. In addition to the data used there, we introduce the possibility to hire part-time workers. Now a jobholder can either work $b^1 = 70\ 000$ or $b^2 = 35\ 000$ time units. Table 3 displays the data. Compared

				wage	
i	P_i	S_i	group	full-time	part-time
1	30 149	1	1	600	330
2	29 365	2	1	600	330
3	30 841	1	1	600	330
4	28 494	1	1	800	440
5	29 801	1	1	900	495
6	28 937	3	1	1 000	550
7	28 305	1	1	1 200	660
8	29 601	1	2	840	462
9	30 517	1	2	840	462
10	29 091	2	2	1 080	594
11	28 744	3	2	1 200	660
12	29 994	1	2	1 200	660
13	31 052	1	2	1 320	726
14	28 039	1	3	780	429
15	29 864	2	3	1 040	572
16	29 939	1	3	1 040	572
17	33 677	1	3	1 040	572
18	29 326	1	3	1 040	572
19	30 014	2	3	1 300	715
20	28 426	2	3	1 430	788

Figure 3: Extended model – instance

to table 4 wages for part-time workers are provided, too, which are assumed to be 55% of full-time wages.

To determine the lower bound LB_1, two networks $G1 = (V, E, w1)$ and $G2 = (V, E, w2)$ have to be defined. The weights $w1$ and $w2$ of the arcs of each network correspond to the wages of full- and of part-time jobholders. Column generation produces up to two columns per iteration. This is accomplished by solving the subproblem for each network separately. In our case we have a total of $2 \cdot (2^{20} - 1) = 2\ 097\ 150$ potential columns. This is because each of the $2^{20} - 1$ different qualification profiles can be evaluated with the cost for full- and part-time jobholders. Surprisingly, only 55 columns have to be generated, until the pricing criterion is reached for both networks. The lower bound LB_1 equals 9 440.513.

Similar to section 4.4 now we can determine the upper bound by solving the integer program and tighten it by performing local search. The upper bound determined for the extended model is $UB_2 = 10\ 871$. In table 16 the qualification profiles, the associated wages and the working time of the corresponding profile are given.

6.2 Individual working times

In the previous sections we have provided a means to compute those qualification profiles which are needed in order to satisfy long-term staffing requirements in a cost efficient way. Afterwards, of course, the question arises how much time each worker should spend on a specific process. This question can easily be addressed by linear programming.

#	profile	cost	working time
1	00000000000010000000	726	35 000
2	01000000000000000000	330	35 000
3	00000010000000000000	660	35 000
4	00000000000001000000	429	35 000
5	00000000111100000000	660	35 000
6	00000000000000000011	788	35 000
7	00000000000000000011	788	35 000
8	00000000000000111100	1 040	70 000
9	00000000000000111100	1 040	70 000
10	00000001100000000000	840	70 000
11	00011100000000000000	550	35 000
12	00011100000000000000	550	35 000
13	01011100000000000000	550	35 000
14	10100000000000000000	600	70 000
15	00000000011100000000	660	35 000
16	00000000011100000000	660	35 000
		10 871	

Table 16: Qualification profiles of full- and part-time jobholders

Using the parameters

m : number of processes i
P_i : demand of processing time for process i, that is, $P_i = \sum_{j=1}^{n} p_{ji}$
u : number of jobholders l
z_{il} : vector of qualification profile of jobholder l
b_l : annual working time of jobholder l

and decision variables

y_{il} : individual working time of jobholder l spent on process i

we get the following model:

$$\min \quad \sum_{i=1}^{m} \sum_{l=1}^{u} y_{il} \tag{19}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} z_{il} y_{il} \leq b_l \qquad l = 1, \ldots, u \tag{20}$$

$$\sum_{l=1}^{u} z_{il} y_{il} \geq P_i \qquad i = 1, \ldots, m \tag{21}$$

$$y_{il} \geq \begin{cases} \delta & \text{if } z_{il} = 1, \\ 0 & \text{otherwise} \end{cases} \qquad i = 1, \ldots, m, l = 1, \ldots, u \tag{22}$$

The value δ in constraint (22) has to be chosen out of the range $0 < \delta < b_l$. To set the lower bound of y_{il} to a positive value in case of $z_{il} = 1$ assures constraint (3) of the staffing model.

Table 17: Time y_{il} spent on process $i = 1, \ldots, m$ by jobholder $l = 1, \ldots, u$

i \ l	1	2	3	4	5	6	7	8	9	10	11
1	9 814	0	0	0	0	0	0	52 758	0	0	0
2	29 355	0	0	0	10	0	0	0	0	0	0
3	30 831	0	0	0	10	0	0	0	0	0	0
4	0	0	0	0	11 272	0	0	17 222	0	0	0
5	0	0	0	0	29 791	0	0	10	0	0	0
6	0	0	0	0	28 917	0	0	10	0	0	10
7	0	0	0	0	0	0	0	0	0	0	69 990
8	0	0	0	0	0	0	10	0	0	60 592	0
9	0	0	0	0	0	10	30 507	0	0	0	0
10	0	0	0	0	0	0	19 703	0	0	9 388	0
11	0	0	0	0	0	8 964	19 770	0	0	10	0
12	0	0	0	0	0	29 984	10	0	0	0	0
13	0	0	0	0	0	31 042	0	0	0	10	0
14	0	0	0	70 000	0	0	0	0	0	0	0
15	0	0	10	0	0	0	0	0	58 608	0	0
16	0	18 557	0	0	0	0	0	0	11 382	0	0
17	0	33 667	0	0	0	0	0	0	10	0	0
18	0	10	29 316	0	0	0	0	0	0	0	0
19	0	10	30 004	0	0	0	0	0	0	0	0
20	0	17 756	10 670	0	0	0	0	0	0	0	0

For illustrative purposes we apply this model to the instance considered in section 4.5 (see table 4). If we assume that a total amount of $b = 70\ 000$ time units is available for every jobholder per year we get the distribution of working time shown in table 17.

7 Summary and future work

The approach presented in this paper is suitable to provide a cost-efficient long-term configuration of the manpower required. It takes into account the needs to handle all processes, accruing in a manufacturing and service producing company. Furthermore, the model can easily be adjusted to face various hazards in personnel planning, resulting from the dynamics of a companies environment.

Among others, the model and the algorithms have been evaluated successfully using data from a printing company, revealing the potential to significantly lower the costs of the work force. All the data needed were easily available due to the fact that the model uses cost accounting data.

Apparently, starting with the manpower of an organization onhand, manpower planning is a dynamic process. As with organization plans, theoretical perfection can be assumed and an ideal requirement of manpower can be determined. Then the requirement can be approached from what exists, so that the plan develops as a progression of existing trends in manpower work ratios (see MacBeath 1966). Furthermore, staff development and career planning must be considered in the organization's structure. The individual jobholder needs to see the potential scope for his own career to develop, and obtain reasonable job satisfaction currently and in the future. The staffing model presented above only delivers the ideal requirements of manpower to meet the forecasted demand of a future production period. Though it totally neglects the issue of staff development, it can be used as part of an overall approach, which takes all issues of staff satisfaction and career planning into account.

There do exist plenty of opportunities for future work. Among others, the development of local search algorithms which produce very good solutions for a wide variety of instances, is of primary interest. Very large-scale neighborhood search techniques (see the survey Ahuja et al. 2002) could be a promising starting point here. Furthermore, the development of exact branch-and-price algorithms is a viable research avenue (see, e.g., Barnhart et al. 1998). Finally, to study the hierarchical integration of staffing and rostering models is an important research topic.

References

[1] AARTS, E.H.L., LENSTRA, J.K. eds. (1987), Local search in combinatorial optimization, Wiley, Chichester

[2] ABERNATHY, W.J. (1973), A three-stage manpower planning and scheduling model – a service sector example, Operations Research, Vol. 21, pp. 693–711

[3] ABRAMSON, D., DANG, H., KRISHNAMOORTHY, M. (1996), A comparison of two methods for soving 0-1 integer programs using a general purpose simulated annealing algorithm, Annals of Operations Research, Vol. 63, pp. 129–150

[4] AHUJA, R.K., ERGUN, Ö., ORLIN, J.B., PUNNEN, A.P. (2002), A survey of very large-scale neighborhood search techniques, Discrete Applied Mathematics, Vol. 123, pp. 75–102

[5] BARNHART, C., JOHNSON, E.L., NEMHAUSER, G.L., SAVELSBERGH, M.W.P., VANCE, P.H. (1998), Branch-and-price: column generation for huge integer programs, Operations Research, Vol. 46, pp. 316–329

[6] BURKE, E., COWLING, P., DE CAUSMAECKER, P., BERGHE, G.V. (2004), The state of the art of nurse rostering, Journal of Scheduling, Vol. 7, pp. 441–499

[7] BURNS, R., CARTER, M. (1985), Work force size and single shift schedules with variable demands, Management Science, Vol. 31, pp. 599–607

[8] EISELT, H.A., SANDBLOHM, C.-L. (2000), Integer programming and network models, Springer, Berlin

[9] ERNST, A., JIANG, H., KRISHNAMOORTHY, M., SHIER, D. (2004), Staff scheduling and rostering: a review of applications, methods and models, European Journal of Operational Research, Vol. 153, pp. 23–27

[10] GILMORE, P.C., GOMORY, R.E. (1960), A linear programming approach to the cutting-stock problem, Operations Research, Vol. 9, pp. 849–859

[11] `http://www.geocities.com/lpsolve/`

[12] `http://www.optimus2020.com/`

[13] JOHNSON, D., ARAGON, C., MCGEOCH, L., SCHEVON, C. (1989), Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning, Operations Research, Vol. 37, pp. 865–892

[14] JOHNSON, D., ARAGON, C., MCGEOCH, L., SCHEVON, C. (1991), Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning, Operations Research, Vol. 39, pp. 378–406

[15] KIRKPATRICK, S. (1984), Optimization by simulated snnealing – quantitative studies, Journal of Stat. Phys., Vol. 34, pp. 975–986

[16] KIRKPATRICK, S., GELATT, C., VECCHI, M. (1983), Optimization by simulated annealing, Science, Vol. 220, pp. 671–680

[17] VAN LAARHOVEN, P., AARTS, E.H.L. (1987), Simulated annealing: theory and applications, Reidel, Dordrecht/Holland

[18] Lohnabkommen für die Druckindustrie 2004 (in German)

[19] MACBETH, G. (1966), Organization and manpower planning, Business Publicatios Ltd., London

[20] MINIEKA, E. (1978), Optimization algorithms for networks and graphs, M. Dekker Inc., New York-Basel

[21] MUNDSCHENK, M., DREXL, A. (2005), Work force planning in the printing industry, Working Paper No. 593, University of Kiel, Germany

[22] PINEDO, M. (2005), Planning and scheduling in manufacturing and services, Springer, Berlin

[23] TIEN, J., KAMIYAMA, A. (1982), On manpower scheduling algorithms, Society of Industrial and Applied Mathematics, Vol. 24, pp. 275–287

[24] WARNER, M. (1976), Nurse staffing, scheduling and reallocation in the hospital, Hospital & Health Services Administration, Vol. 21:3, pp. 77–90

[25] WIJMGAARD, J. (1983), Aggregation in manpower planning, Management Science, Vol. 29, pp. 1427–1435