Haase, Knut; Latteier, Jörg; Schirmer, Andreas

The course scheduling problem at Lufthansa Technical Training

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 441

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

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 441

The Course Scheduling Problem at Lufthansa Technical Training

Haase, Latteier, Schirmer

April 5, 1997

Knut Haase, Jörg Latteier, Andreas Schirmer
Institut für Betriebswirtschaftslehre, Lehrstuhl für Produktion und Logistik
Christian-Albrechts-Universität zu Kiel, Olshausenstr. 40, D-24118 Kiel, Germany

Phone, Fax +49-431-880-15 31
E-Mail haase@bwl.uni-kiel.de, schirmer@bwl.uni-kiel.de

URL http://www.wiso.uni-kiel.de/bwlinstitute/Prod
ftp://www.wiso.uni-kiel.de/pub/operations-research

Contents

Abstract: Lufthansa Technical Training GmbH (LTT) performs training courses for Lufthansa AG as well as for several other international airlines. Courses of about 670 different types are offered of which several hundred take place each year. The course scheduling problem faced by LTT is to develop a yearly schedule which maximizes the profit margin incurred while meeting a variety of complex precedence, temporal, and resource-related constraints. A "good" operational schedule should also meet a number of additional subordinate objectives. We formalize the problem and describe a heuristic scheme along with several priority rules, as well as a local search method to determine well-suited weights for weighted composite rules. The operational planning situation of 1996 served as our major test instance; additional test instances were constructed by modifying this data. Several computational experiments were carried out to evaluate the performance of the algorithms. It turned out that the best so-found schedule is substantially better in terms of the profit margin incurred than the solution manually constructed by LTT.

Keywords: COURSE SCHEDULING; AVIATION; TRAINING; HEURISTICS; LOCAL SEARCH

1. Introduction

Lufthansa AG, the German national flag carrier, currently has about 40,000 employees and is one of the largest employers in the country. Almost every fourth of them is engaged in the technical area that is responsible for the checks, maintenance, and overhauls necessary to maintain a high standard of safety. The training of its technical staff is entrusted to Lufthansa Technical Training GmbH (LTT). Apart from Lufthansa Technik AG, the technical departments of numerous other airlines are customers of LTT. About 670 different theoretical and practical course types are offered, lasting from a few days up to one year and covering areas such as aircraft maintenance, overhauls, or inspector training. To satisfy all (or at least most) customer requests, each year several hundred courses of different types are held in offices in Frankfurt, Hamburg, and Berlin.

The manual construction of an operational course schedule for the LTT is an extraordinarily tedious and time-consuming task that each year monopolizes two employees for several weeks and results in a plan taking several square meters of paper. Due to the long time this process takes, it is virtually impossible to apply different planning strategies and assess the resulting schedules. This, however, would be desirable since usually not all requested courses take place, mostly due to scarce instructor capacities. It was conjectured that the ability to come up with several alternatives would allow to select better schedules than those manually constructed.

In Section 2 we briefly survey the open literature on related problem fields. We formalize the course scheduling problem (CSP) as faced by LTT in terms of a mathematical model in Section 3. Section 4 presents an algorithmic construction scheme together with different priority rules. Section 5 provides computational results analyzing the effect of the priority rules, focussing on the actual operational LTT data of 1996. Section 6 concludes the paper with some final remarks.

2. Literature Review

Within the last 15 years, a variety of articles has been published on problems located within the area of educational scheduling. Virtually all articles gravitate around two problem fields (Figure 1), viz. school timetabling (de Gans 1981; Abramson 1991; Cangalovic, Schreuder 1991, 1992; Hertz 1992; Alvarez-Valdes et al. 1994; Costa 1994) and academic course scheduling (Tripathy 1984; Aubin, Ferland 1989; Kang, White 1992; Sampson et al. 1995). One might conjecture that the CSP and the above problem settings are rather closely related. However, the majority of problems covered in the literature differ in some fundamental aspects from the CSP:

- Most timetabling or academic course scheduling problems are adapted to non-profit organizations such as schools, colleges, or universities. None of these organizations is seeking to maximize the profit margin arising from a schedule.

- Both timetabling and academic course scheduling intend to find periodic schedules; these have to be repeated at regular intervals which are relatively small, e.g. one week. In contrast, a solution of the CSP represents a non-periodic schedule for a much longer planning horizon, e.g. one year.

- For many of the problems discussed even the associated feasibility problem is (strongly) **NP**-complete, owing to the combination of scarce resources and a fixed planning horizon within which all courses (lessons, lectures) have to be scheduled. For the CSP in contrast, there exists a trivial feasible solution for each instance, viz. to schedule no course at all, since requested courses may be rejected.

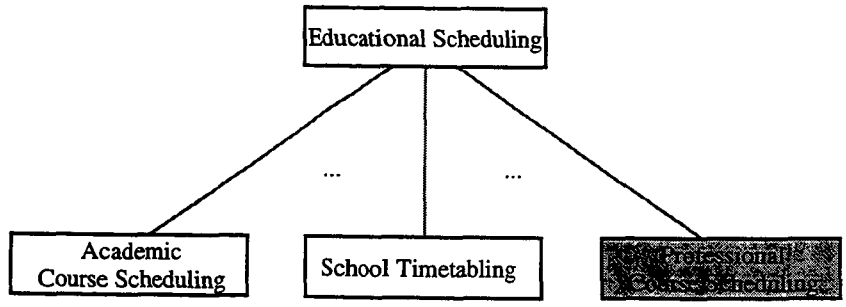

Figure 1: Professional Course Scheduling Is One Discipline of Educational Scheduling

An exception to these findings is provided by the work of Eglese, Rand (1987) who address the scheduling of conference seminars. The problem they consider is to find an assignment of periods and participants to seminars which covers all participants and meets a number of constraints essentially representing scarce room resources. However, since all seminars have a

uniform duration of one period each seminar is scheduled as a whole. Each seminar can only be held by one speaker rather than by a group of alternative speakers. Finally, no precedence or temporal constraints are part of the problem. Hence, also the approach of Eglese, Rand is not applicable to the CSP. Other than that, only very little research has been documented on related scheduling problems.

Summarizing, we may infer that the CSP substantially differs from problems in seemingly related fields such as timetabling or academic course scheduling (for more details cf. Haase et al. (1997)). We therefore consider the CSP to be in a separate problem field which we term professional course scheduling (Figure 1).

3. Model

3.1. Symbols and Definitions

The model formulation we propose here is founded on concepts from project scheduling; an alternative formulation in terms of a knapsack model is given in Haase, Latteier (1996). Each course is interpreted as a project consisting of a number of component jobs (basically corresponding to lessons) which are to be scheduled subject to specific temporal and resource constraints. Hence, the CSP can be regarded as a multi-project scheduling problem. Still, we will see that there are some strings attached to this problem, which make it differ from standard resource-constrained project scheduling problems. The CSP can be couched as follows.

Courses: N courses of CT different course types are to be scheduled within a fixed planning horizon of T periods; T comprises all periods available for training, i.e. all lessons of each workable day, taking into account weekends and holidays. Each course comprises a specific number of jobs, each representing one training lesson (Figure 2) and thus having a length of one period, and a fictitious job to be explained below. W.l.o.g. we assume that all jobs of a course n are consecutively indexed and have a lower index than any job belonging to a course $n' > n$. The total number of jobs is J. Let denote f_n (l_n) the first (last) job of course n.

Resources: Each course requires certain scarce resources such as instructors or rooms. Let denote R^{ρ} the set of training rooms and R^{γ} the set of customers having filed course type requests. The (renewable) capacity of any such resource $r \in R^{\rho} \cup R^{\gamma}$ in period t is given by c_{rt}. The interpretation of the (renewable) room capacity is straightforward; the (renewable) customer capacity states the number of customer staff allowed to be trained at the same time (participant quota). Further, demand for each course type c is specified by each customer $r \in R^{\gamma}$ in terms of the number of intended participants (some of these may be zero). Accordingly, let denote $R^{\gamma\gamma} := R^{\gamma} \times \{1,...,CT\}$ the set of customer-course type resources such that each $r \in$

$R^{\gamma\gamma}$ represents a specific customer with demand for a specific course type. The (nonrenewable) capacity c_r for each such r denotes the number of intended participants. Finally, R^{τ} denotes the set of all instructors. The (renewable) capacity of instructor $r \in R^{\tau}$ in period t is given by c_{rt} while its (nonrenewable) capacity within the complete planning horizon is denoted by c_r; the former (binary) capacity specifies whether an instructor is available for instruction in a period, whereas the latter provide capacity for the actual instructing lessons as well as preparation periods for these lessons.

Modes: Since most lessons can be held by different instructors and courses can include different participants, we represent alternative ways of performing courses by defining for each job j a number M_j of modes in which it can be processed. In addition, for each job j there is a set $S_j \subseteq \{1,...,T\}$ of start periods; a job may begin only in one of these periods. For each job j processed in mode m, the (renewable) per-period usage of a resource $r \in R^{\tau} \cup R^{\rho} \cup R^{\gamma}$ is specified by k^{θ}_{jmr} while the (nonrenewable) total consumption of a resource $r \in R^{\tau} \cup R^{\gamma\gamma}$ is specified by k^{ν}_{jmr}.

Participants: For each course n, a fictitious job serves to handle all requirements related to participants (this job is not depicted in Figure 2); w.l.o.g. we use f_n+1 to denote it. Its (nonpreemptable) duration equals the duration of course n which, owing to the number of weekends and holidays comprised, depends on the period $t \in S_j$ in which n begins. Each of its modes represents a feasible composition of participants from one or several customers where participants are assigned to only one course at a time.

Topics: Lessons on the same subject are combined to topics if they require the same instructor qualifications and training facilities. To represent these requirements, we introduce instructor and room groups; each such group is a set of resources which are equivalent in the sense that they can be used alternatively to meet the requirements of a topic. The set of all instructor groups is denoted by GT, the set of all room groups by GR. With each instructor group $g \in GT$ we associate a set of instructors $R_g \subseteq R^{\tau}$ and with each room group $g \in GR$ a set of rooms $R_g \subseteq R^{\rho}$.

Blocks: Topics in turn are aggregated to blocks (Figure 2). There are B blocks, each consisting of several consecutive jobs of the same course. Let denote $\bar{f}_b$ ($\bar{l}_b$) the first (last) job of block b and fb_n (lb_n) the first (last) block of course n. For educational and legal reasons, precedences exist between some blocks. For each course type, a so-called standard sequence of blocks is known, which meets all precedence requirements and performs well under practical considerations. Each block b is associated with a set of instructor groups $G^{\tau}_b \subseteq GT$ and a set of room groups $G^{\rho}_b \subseteq GR$, which consist of the groups associated with the topics of the block. Therefore, the set of instructors which may be involved in a block can be derived from

$$R^{\tau}_b = \bigcup_{g \in G^{\tau}_b} R_g \qquad (1 \le b \le B) \quad (1)$$

W.l.o.g. these groups can be assumed to be disjoint; so, each combination of block b and (instructor or room) group g implicitely defines a topic. The requirement r_{bg} of a block b for a group g specifies the workload of a topic, e.g. the number of instructor lessons required. Loosely speaking, each job j of a block b contributes a share of k^{θ}_{jmr} to r_{bg} if it is processed in mode m and if resource r belongs to group g. The requirement can be met by each instructor or room in the respective group. However, the number of different instructors in a group g which may be used for block b is limited by a maximum number m_{bg}; e.g. by setting $m_{bg} = 1$ this allows to guarantee that all lessons of a topic are held by the same instructor.

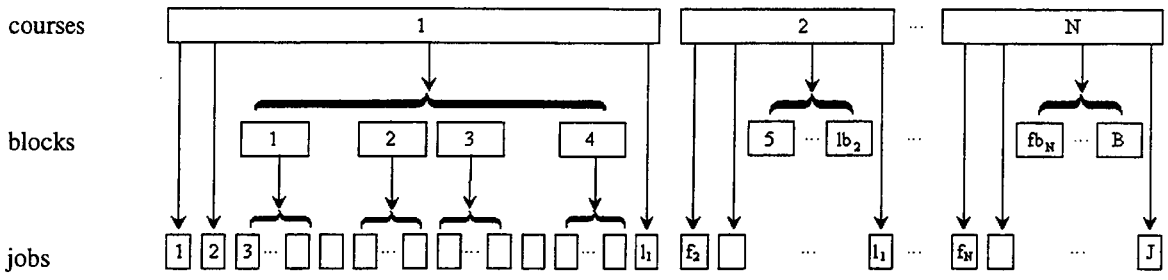

Figure 2: The Relation between Courses, Blocks, and Jobs

Fees and costs: The fees p_{nm} due from a course n depend on the mode assigned to the participant job f_n+1. In addition to instructors which belong to the staff, tutors (qualified technical Lufthansa personnel or former LTT instructors) can be used for certain topics; both incur costs. Instructor cost arises only if instructors are used away from their home bases, it is composed of a fixed part reflecting transportation and opportunity cost (for working hours lost to traveling), and a variable part representing accommodation and expenses; (variable) tutor cost is due since tutors are paid per lesson taught. To summarize this for modelling purposes, using an instructor $r \in R^{\tau}_b$ in a block b incurs fixed cost of t^{ζ}_{br} whereas conducting a job j in mode m entails variable cost of t^{ω}_{jm}.

Exclusion sets: Specific combinations of job, mode, and start period can be mutually conflicting. To exclude such conflicts, we introduce for each such triple (j, m, t) an exclusion set E_{jmt}, consisting of all conflicting triples (j', m', t') which may not be scheduled along with it. The concept of exclusion sets is a rather powerful instrument since it allows to formulate a variety of different constraints which otherwise would require separate constraint types.

- Each job can be forced to be processed in at most one combination of mode and start period, i.e. either once or not at all.

- Simultaneous (parallel) execution of jobs can be prevented by including all triples of the respective jobs having identical or overlapping start periods.

Parameter	Definition
B	Number of blocks, indexed by b
c_r	Total capacity of (nonrenewable) resource r
c_{rt}	Per-period capacity of (renewable) resource r
CT	Number of course types, indexed by c
d_{jt}	Duration of job j when started in period t
E_{jmt}	Exclusion set of job j when performed in mode m and started in period t
f_n (l_n)	First (last) job of course n
$\bar{f}_b$ ($\bar{l}_b$)	First (last) job of block b
$\hat{f}_f$ ($\hat{l}_f$)	First (last) job of frame f
fb_n (lb_n)	First (last) block of course n
F	Number of job frames, indexed by f
G^{τ}_b (G^{ρ}_b)	Set of instructor (room) groups available for block b
GT (GR)	Set of instructor (room) groups, indexed by g
I_f	Set of infeasible jobs of frame f
J	Number of jobs, indexed by j
k^{θ}_{jmr} (k^{ν}_{jmr})	Per-period usage (total consumption) of renewable (nonrenewable) resource r required to perform job j in mode m
m_{bg}	Maximum number of resources of resource group g usable for block b
M_j	Number of modes, indexed by m
N	Number of courses, indexed by n
p_{nm}	Fees from holding course n in mode m of job f_n+1
r_{bg}	Requirement of block b for resource group g
R^{τ} (R^{ρ}, R^{γ}, $R^{\gamma\gamma}$)	Set of instructors (rooms, customers, customer-course type resources)
R_g	Set of resources contained in resource group g, indexed by r
R^{τ}_b	Set of instructors assignable to block b
S_j	Set of start periods of job j, indexed by t
t^{ζ}_{br}	Fixed cost incurred by assigning instructor r in block b
t^{ω}_{jm}	Variable cost incurred by performing job j in mode m
T	Number of periods, indexed by t

Table 1: Parameters of the CSP

- Precedence relations can be modelled by including into the exclusion sets the violating mode-start period combinations of the immediate predecessors.

- Travel or rest times of instructors can be taken into account by excluding all conflicting triples where the amount of travel or rest times is unsufficient.

Frames: Another means to restrict scheduling of conflicting jobs is job frames. Each frame f is associated with a first job $\hat{f}_f$, a last job $\hat{l}_f$, and a set of jobs I_f for which processing between start of the first and completion of the last job is infeasible. Here, all such jobs belong to the same course. Note that a frame has an effect if and only if its first and last job are actually scheduled. With respect to the practical situation at hand, job frames serve to guarantee e.g. that between the first and the last lesson of one block no lesson of another block is scheduled.

The parameters used are summarized (in alphabetical order) in Table 1. W.l.o.g. all parameters are assumed to be nonnegative integers.

3.2. Model Formulation

Using the above conceptualization, the assignment of modes and periods to jobs can be represented by binary variables x_{jmt} $(1 \le j \le J;\ 1 \le m \le M_j;\ t \in S_j)$ where

$$x_{jmt} = \begin{cases} 1 & \text{if job j is performed in mode m and started in period t} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

and assignment of instructors to blocks by binary variables y_{br} $(1 \le b \le B;\ r \in R^{\tau}_b)$ where

$$y_{br} = \begin{cases} 1 & \text{if instructor r is assigned to (at least) one job of block b} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Now the CSP can be expressed in terms of the following model:

Maximize

$$Z(x) = \sum_{n=1}^{N} \sum_{m=1}^{M_{f_n+1}} p_{nm} \sum_{t \in S_{f_n+1}} x_{f_n+1,m,t} - \sum_{b=1}^{B} \sum_{r \in R^{\tau}_b} t^{\zeta}_{br} \cdot y_{br} - \sum_{j=1}^{J} \sum_{m=1}^{M_j} t^{\omega}_{jm} \sum_{t \in S_j} x_{jmt} \tag{4}$$

subject to

$$\sum_{(j',m',t') \in E_{jmt}} x_{j'm't'} \le \left|E_{jmt}\right| \cdot (1 - x_{jmt}) \qquad (1 \le j \le J;\ 1 \le m \le M_j;\ t \in S_j) \tag{5}$$

$$(l_n - f_n + 1) \cdot \sum_{m=1}^{M_{f_n}} \sum_{t \in S_{f_n}} x_{f_n mt} = \sum_{j=f_n}^{l_n} \sum_{m=1}^{M_j} \sum_{t \in S_j} x_{jmt} \qquad (1 \le n \le N) \tag{6}$$

$$\sum_{j\in I_f}\sum_{m=1}^{M_j}\sum_{\substack{t=s\\ t\in S_j}}^{u+d_{\hat{I}_f,u}-1} x_{jmt} \le |I_f|\cdot\left(2-\sum_{m=1}^{M_{\hat{f}_f}} x_{\hat{f}_f ms}-\sum_{m=1}^{M_{\hat{I}_f}} x_{\hat{I}_f mu}\right) \quad (1\le f\le F;\ s\in S_{\hat{f}_f};\ u\in S_{\hat{I}_f}) \quad (7)$$

$$\sum_{r\in R_g}\sum_{j=\bar{f}_b}^{\bar{I}_b}\sum_{m=1}^{M_j} k^{\theta}_{jmr}\sum_{t\in S_j} x_{jmt} \ge r_{bg}\sum_{m=1}^{M_{\bar{f}_b}}\sum_{t\in S_{\bar{f}_b}} x_{\bar{f}_b mt} \qquad (1\le b\le B;\ g\in G^{\tau}_b\cup G^{\rho}_b) \quad (8)$$

$$\sum_{j=\bar{f}_b}^{\bar{I}_b}\sum_{m=1}^{M_j} k^{\theta}_{jmr}\sum_{t\in S_j} x_{jmt} \le (\bar{I}_b-\bar{f}_b+1)\cdot y_{br} \qquad (1\le b\le B;\ r\in R^{\tau}_b) \quad (9)$$

$$\sum_{r\in R_g} y_{br} \le m_{bg} \qquad (1\le b\le B;\ g\in G^{\tau}_b) \quad (10)$$

$$\sum_{j=1}^{J}\sum_{m=1}^{M_j} k^{\theta}_{jmr}\sum_{\substack{u\in S_j\\ u\le t\\ u+d_{ju}>t}} x_{jmu} \le c_{rt} \qquad (r\in R^{\tau}\cup R^{\rho}\cup R^{\gamma};\ 1\le t\le T) \quad (11)$$

$$\sum_{j=1}^{J}\sum_{m=1}^{M_j} k^{\nu}_{jmr}\sum_{t\in S_j} x_{jmt} \le c_r \qquad (r\in R^{\tau}\cup R^{\gamma\gamma}) \quad (12)$$

$$x_{jmt}\in\{0,1\} \qquad (1\le j\le J;\ 1\le m\le M_j;\ t\in S_j) \quad (13)$$

$$y_{br}\in\{0,1\} \qquad (1\le b\le B;\ r\in R^{\tau}_b) \quad (14)$$

The objective function (4) maximizes the total profit margin, calculated as the cumulated fees of all scheduled courses less all incurred costs. Constraints (5) - (7) formulate the temporal restrictions. Constraints (5) ensure for each job that no element of the corresponding exclusion set is scheduled. By appropriately instantiating these sets, each job is assigned at most one combination of mode and start period. Constraints (6) guarantee that only complete courses are scheduled. Constraints (7) relate to the job frames: The term in parentheses becomes zero for at most one combination of start periods s of the frame's first and u of the frame's last job. In that case, none of the jobs $j \in I_f$ can be scheduled within the interval $[s, u+d_{\hat{I}_f u}-1]$; otherwise, constraints (7) have no effect. Constraints (8) - (12) represent the appropriate resource restrictions. Constraints (8) ensure for each block that all requirements for instructor and room groups are met. Each such group corresponds to a specific topic to be taught within the block. If the block is not scheduled, which due to (6) is indicated by its first job, constraints (8) remain without effect; otherwise, the requirements of all topics of the block have to be satisfied by the jobs constituting the block, i.e. a feasible combination of modes for the jobs

must be scheduled. If an instructor r is used in at least one job of block b then the corresponding variable y_{br} is set to one by constraints (9); otherwise, it takes value zero due to (4). Constraints (10) limit the number of different instructors to be used within the same block. Renewable and nonrenewable capacities of all resource types are respected due to constraints (11) and (12).

4. Solution Method

4.1. Algorithmic Scheme

The above model is based on concepts from the field of project scheduling and in this way, the CSP can be regarded as a multi-project scheduling problem. One might therefore wonder whether standard project scheduling algorithms can be applied. Yet, some specific characteristics apply to this problem, which differ from the standard assumptions of project scheduling and can be summarized as follows.

- Each course can either be scheduled as a whole or not at all.
- All courses scheduled must be completed within the fixed planning horizon.
- In addition to conventional precedence relations, some jobs have to be scheduled maintaining a minimum time-lag between them. Also, for each course there is a maximum time-lag between first and last scheduled job, ensuring course compactness.
- The start of some jobs is restricted to a set of feasible start periods.
- The objective is to maximize the profit margin induced by a schedule.

We therefore describe a tailored algorithmic scheme for the CSP which can be categorized as a priority rule-based construction method. It employs a serial scheduling scheme (Kelley 1963) and uses deterministic priority rules to solve selection or assignment conflicts. Partial course schedules, starting from the empty one, are augmented in a stage-wise fashion until all courses have been considered. We use two sets, viz. the set of completed courses which have already been feasibly scheduled, and the set of remaining courses. In each stage, a specific priority rule serves to select exactly one of the remaining courses. We then examine combinations of start period, block sequence, and assignable resources until a feasible one is found. In that case, the course is scheduled in that combination and resource capacities are updated accordingly; otherwise it fails. Once a course is scheduled, this assignment remains fixed. We should emphasize that the algorithmic scheme, though applicable to the CSP, is adjusted to the specific situation found at LTT, namely by using tutors and preferring the standard block sequence.

In order to examine combinations in a systematic way, we distinguish four levels within each stage. Each level is associated with a subspace of the search space spanned by the above combinations and can be described in terms of two characteristics. The first one determines the instructors which might be assigned to a topic: Either the staff of all offices or only those of the performing office, i.e. the office where the course is to be held, are considered. The second one determines the sequences in which the blocks of a course may be performed: Either only the standard sequence or all feasible sequences are considered. Table 2 presents the levels and the corresponding realizations of the defining characteristics.

Proceeding in this way ensures low travel cost because possibilities to use less expensive instructors from the performing office are examined first. As an additional benefit the algorithm becomes very fast since usually a high percentage of courses can be scheduled in standard sequence; note that with increasing level also the number of combinations to be examined increases. However, this procedure does not guarantee the minimum cost combination for a course to be found since the search on each stage follows a first-fit strategy, i.e. the search terminates with the first feasible combination found.

Level	Instructing Staff Considered	/	Block Sequences Considered
1	Staff of performing office	/	Standard sequence
2	Staff of performing office	/	All other sequences
3	Complete staff	/	Standard sequence
4	Complete staff	/	All other sequences

Table 2: Levels of the Algorithmic Scheme

Since the capacity of tutors is virtually unlimited, employing them allows to overcome shortages of instructing capacity which otherwise might make certain courses impossible to schedule. We distinguish two strategies: The first is to regard tutors as ordinary instructors (TUO); usage of tutors is rated by the appropriate (high) costs and is only possible at levels 3 and 4. TUO is a local strategy as it considers only the course to be scheduled and assigns the lowest cost instructors. This implies a reduction of the solution space since assignments, once fixed, will not be changed and tutors can only be used in specific topics. Therefore, we propose a second, global strategy where tutors are preferred whenever possible (TUP); so they can be assigned at all four levels. A so-constructed schedule can be improved afterwards by a straightforward procedure that replaces tutors in topics where (less costly) instructors are still available. Employing more tutors also allows to schedule more courses than the rather restrictive TUO strategy. To summarize, TUP seems to be more fitting for situations with a high workload and thus a low service level, whereas TUO seems adequate when facing low work-

loads such that a high service level can be achieved using staff only.

Table 3 lists the additional notation used to formulate the algorithmic scheme which in turn is presented in Table 3. We refrain from discussing the (rather straightforward) algorithmic scheme in detail and restrict ourselves to filling in the gaps within the scheme. In Step 1, remaining (instructor and room) capacities rc_r are initialized according to

$$rc_r := c_r \qquad (r \in R^\tau) \quad (15)$$

$$rc_r := \sum_{t=1}^{T} c_{rt} \qquad (r \in R^\rho) \quad (16)$$

Notation	Definition
l	Level of algorithmic scheme
$\vec{m}$	Mode tuple
rc_r	Remaining capacity of resource r
rd_r	Remaining demand for resource r
SOCC	Set of completed courses
$SOFMT_{nl\vec{t}}$	Set of feasible mode tuples course n at level l w.r.t. period tuple $\vec{t}$
$SOFPT_{nls}$	Set of feasible period tuples course n at level l w.r.t. start period s
$SOFSP_n$	Set of feasible start periods of course n
SORC	Set of remaining courses
s	Start period
$\vec{t}$	Period tuple

Table 3: Notation Used in the Algorithmic Scheme

while remaining (instructor and room) demand rd_r is calculated by apportioning the demand for each resource group among the corresponding resources, i.e.

$$rd_r := \sum_{n \in SORC} \sum_{b=fb_n}^{lb_n} \sum_{g \in G_b^\tau \cup G_b^\rho} \sum_{r \in R_g} \frac{r_{bg}}{|R_g|} \qquad (r \in R^\tau \cup R^\rho) \quad (17)$$

Priority rules for selecting courses in Step 2 and start periods in Step 4 will be discussed in the next section.

In Step 4, the set $SOFPT_{nls}$ of feasible period tuples for a course n starting in period s is determined on level l by considering all tuples $\vec{t} = ((f_n, t_{f_n}), \ldots, (l_n, t_{l_n}))$ which assign to the

first job f_n the period s, to all other jobs j ($f_n+1 \le j \le l_n$) a start period t_j ($t_j \in S_j$). Each such tuple is feasible iff it respects the corresponding exclusion sets, heeds all frames within the course, and - if on level 1 or 3 - schedules the course jobs in standard sequence.

With the subordinate goal of adherence to standard sequence in mind also for levels 2 and 4, we select in Step 5 the period tuple having the least deviation from standard sequence. Since the standard sequence entails the least possible duration of a course, this deviation can be measured in terms of the start periods of the course's jobs. Consider tuples $\vec{t} = ((f_n, t_{f_n}),...,(l_n, t_{l_n}))$ and $\vec{t}' = ((f_n, t'_{f_n}),...,(l_n, t'_{f_n}))$: We say that the former has less deviation from standard sequence iff there is a job j ($f_n \le j \le l_n$) such that $t_j < t'_j$ and $t_i = t'_i$ for all i ($f_n \le i \le j-1$).

Step 1: (Initialize)
SORC := {1,...,N}; SOCC := ∅;
calculate rc_r, rd_r ($r \in R^\tau \cup R^\rho$);
$x_{jmt} := 0$ ($1 \le j \le J$; $1 \le m \le M_j$; $t \in S_j$);
$y_{br} := 0$ ($1 \le b \le B$; $r \in R^\tau$);

Step 2: (Select course)
if SORC = ∅ then goto Step 8;
select $n \in$ SORC by course selection rule;
SORC := SORC \ {n};
l := 0;

Step 3: (Set level)
if l = 4 then begin update rd_r ($r \in R^\tau \cup R^\rho$); goto Step 2; end;
l := l + 1;
$SOFSP_n := S_{f_n}$;

Step 4: (Select start period)
if $SOFSP_n = \emptyset$ then goto Step 3;
select $s \in SOFSP_n$ by start period selection rule;
$SOFSP_n := SOFSP_n \setminus \{s\}$;
compute $SOFPT_{nls}$;

Step 5: (Select job sequence)
if $SOFPT_{nls} = \emptyset$ then goto Step 4;
select $\vec{t} \in SOFPT_{nls}$;
$SOFPT_{nls} := SOFPT_{nls} \setminus \{\vec{t}\}$;

Step 6: (Check sequence feasibility)
compute $SOFMT_{nl\vec{t}}$;
if $SOFMT_{nl\vec{t}} = \emptyset$ then goto Step 5;
select $\vec{m} \in SOFMT_{nl\vec{t}}$;

Step 7: (Schedule course)
$x_{jm_jt_j} := 1$ ($f_n \le j \le l_n$; $m_j \in \bar{m}$; $t_j \in \bar{t}$);

update y_{br} ($fb_n \le b \le lb_n$; $r \in R^\tau$);

update rc_r, rd_r ($r \in R^\tau \cup R^\rho$);
SOCC := SOCC $\cup$ {n};
goto Step 2;

Step 8: (Evaluate schedule)
compute objective function value;

Table 4: Algorithmic Scheme

In Step 6, the set $SOFMT_{nl\bar{t}}$ of feasible mode tuples for a course n with period tuple $\bar{t}$ is determined on level l by considering all tuples $\bar{m} = ((f_n, m_{f_n}), \ldots, (l_n, m_{l_n}))$ which assign to each job j a mode m_j. Each such tuple is feasible iff resource requirements of the topics are satisfied, the maximum number of resources (instructors) usable is respected, travel and resting times are considered, availability of all resources is verified, and to each topic only instructors compatible with current level and applied tutor strategy are assigned. Selection of a mode tuple capitalizes on the fact that all blocks of the same course are mutually independent with respect to resource requirements as well as travel and resting times. Hence, mode selection can be done separately for each block and the participant job. To achieve a high profit margin, a mode tuple where the mode of the participant job offers the maximum number of participants is selected for the latter. Blocks are considered in the sequence induced by the period tuple already selected. Within a block, we start mode selecting with the topic with most required instructors; ties are resolved by selecting the topic with longest duration. Under tutor strategy TUP, all available tutors are assigned first, followed by staff instructors in order of increasing cost; under strategy TUO, instructors and tutors are considered in that order. In any case, among these as well as among the room resources those resources having the lowest load factor are preferred, where the load factor expresses the ratio of remaining demand to remaining capacity. In case of similar load factors (difference smaller than 15%), that resource having more remaining capacity is preferred.

Step 7 updates remaining instructor and room capacities by

$$rc_r := rc_r - \sum_{j=f_n}^{l_n} k^{\nu}_{jm_jr} \qquad (r \in R^\tau) \quad (18)$$

$$rc_r := rc_r - \sum_{j=f_n}^{l_n} k^{\rho}_{jm_jr} \qquad (r \in R^\rho) \quad (19)$$

Remaining demand is updated according to (17).

4.2. Course Selection Rules

We will now turn our attention to several priority rules which can be incorporated into this scheme. Each combination of the scheme with one course and one start period selection rule defines one algorithm for the CSP. In the sequel, we describe some rules for selecting the next course to be scheduled. While the first rule tries to improve the objective function value in a greedy fashion, the other ones represent different ways of quantifying how difficult a specific course is to schedule. Given a decision set D of selection candidates, we denote the priority value for each candidate $d \in D$ by $v(d)$; whether the minimum or the maximum priority value determines the selectee is indicated by extremum $\in \{\min, \max\}$.

- Highest Profit Margin (HPM)

  Since the foremost objective is to maximize the profit margin incurred by a schedule, this rule selects the course offering the highest profit margin. Let m^*_n denote a mode of the participant job of course n with maximum number of participants.

$$v(n) = p_{nm^*_n} \qquad \text{extremum} = \max \quad (n \in \text{SORC}) \quad (20)$$

- Longest Duration (LDN)

  Based on the experience from manual scheduling, the duration of a course, which can easily be derived from the number of jobs, is a meaningful indicator of the difficulty of scheduling a particular course.

$$v(n) = l_n - f_n + 1 \qquad \text{extremum} = \max \quad (n \in \text{SORC}) \quad (21)$$

- Number of Topics (NOT)

  A large number of topics within a course implies that additional instructor groups need to be considered, possibly incurring additional travel and resting times, and thus making the construction of feasible schedules more difficult. The number TN_n of topics of a course can be determined by accumulating the number of required instructor groups per block, i.e.

$$TN_n := \sum_{b=fb_n}^{lb_n} \left| G^{\tau}{}_b \right| \qquad (1 \le n \le N) \quad (22)$$

$$v(n) = TN_n \qquad \text{extremum} = \max \quad (n \in \text{SORC}) \quad (23)$$

- Least Feasible Start Periods (LFSP)

The less start periods are available for a course, the less flexible it can be scheduled. The number of start periods provides a unique discriminative potential because it allows - in contrast to all other measures considered - to distinguish courses of the same type.

$$v(n) = \left|S_{f_n}\right| \qquad \text{extremum} = \min \quad (n \in \text{SORC}) \quad (24)$$

- Average Instructors per Topic (AIPT)

The availability of instructors plays the most important role for the difficulty of a CSP-instance. The more instructors can be assigned to a topic, the larger the probability that the required staff is available. AIPT states the average number of instructors that can be employed within a topic of a course:

$$v(n) = \frac{\sum_{b=fb_n}^{lb_n} \sum_{g \in G^{\tau}_b} \left|R_g\right|}{TN_n} \qquad \text{extremum} = \min \quad (n \in \text{SORC}) \quad (25)$$

- Average Load Factor per Topic (ALFT)

The load factor lf_r of an instructor r is defined as

$$lf_r := \frac{rd_r}{rc_r} \qquad (r \in R^{\tau};\ rc_r > 0) \quad (26)$$

and the average load factor avg_lf_g of an instructor group g as

$$avg\_lf_g := \frac{\sum_{\substack{r \in R_g \\ rc_r > 0}} lf_r}{\left|\left\{r \middle| r \in R_g \wedge rc_r > 0\right\}\right|} \qquad (g \in GT) \quad (27)$$

Note that if all instructors in the instructor group of a topic have a remaining capacity of zero the course cannot be scheduled. The average load factor per topic is calculated as follows:

$$v(n) = \frac{\sum_{b=fb_n}^{lb_n} \sum_{g \in G^{\tau}_b} avg\_lf_g}{TN_n} \qquad \text{extremum} = \max \quad (n \in \text{SORC}) \quad (28)$$

4.3. Start Period Selection Rules

Two rules dealing with the selection of the next start period to be examined are described.

- Effective Course Duration (ECD)

  This rule explicitly considers the subordinate objective of minimizing the effective course duration, specified by the duration of the participant job of the course.

$$v(s) = d_{f_n+1,s} \qquad \text{extremum} = \min \quad (s \in SOFSP_n) \quad (29)$$

- Already Scheduled Load (ASL)

  A more promising strategy could be to choose those start periods that hold higher remaining capacities with respect to instructors and room. Therefore, this rule prefers periods with a low number of lessons scheduled for the completed courses.

$$v(s) = \sum_{b=1}^{B} \sum_{j=\underline{f}_b}^{\bar{I}_b} \sum_{m=1}^{M_j} \sum_{\substack{t \in S_j \\ s \le t < s+d_{f_n+1,s}}} x_{jmt} \qquad \text{extremum} = \min \quad (s \in SOFSP_n) \quad (30)$$

Table 5 summarizes all priority rules introduced so far (in alphabetical order) and gives a characterization in terms of some straightforward criteria (Kolisch 1995).

Priority Rule	Selecting	Static vs. Dynamic	Local vs. Global	Simple vs. Composite
AIPT	course	S	L	S
ALFT	course	D	G	S
ASL	start period	D	G	S
ECD	start period	S	L	S
HPM	course	S	L	S
LDN	course	S	L	S
LFSP	course	S	L	S
NOT	course	S	L	S

Table 5: Classification of the Priority Rules

4.4. Composite Rules and Local Search

As can be seen from Table 5, all above rules can be characterized as simple priority rules. However, since usually more than one course of each type has to be scheduled into an operational plan, all these rules have the inherent deficiency that many courses of the same type may have the same or similar priority values. In that case, the discriminative potential of such rules is limited. To overcome this deficiency, it is straightforward to combine several different informations and compose them into one priority value. This motivates the application of composite selection rules.

Given a decision set D, then let for each $d \in D$ denote $v_i(d)$ $(1 \leq i \leq I)$ priority values derived from different rules i and β_i ($\beta_i \in [-1, 1]$) the (exponential) weight accorded to rule i. Now, we can define a composite rule in general as

$$v(d) = \prod_{i=1}^{I} v_i(d)^{\beta_i} \qquad \text{extremum} = \max \quad (d \in D) \quad (31)$$

Note that this definition includes the special case of a simple rule for $I = 1$. We should emphasize that for each priority rule i, the corresponding weight allows to vary the influence of the rule as well as whether candidates with high or with low priority values are to be preferred. On one hand, $\beta_i \in (0, 1]$ implies extremum = max and increasing values tend to pronounce the differences between the courses. On the other hand, $\beta_i \in [-1, 0)$ implies extremum = min and increasing values tend to reduce the differences between the courses. For $\beta_i = 0$, all candidates receive the same priority value. To illustrate the general applicability of this concept, we demonstrate a combination of the rules HPM and LFSD with weights of $\beta_1 = 1$ and $\beta_2 = -0.5$.

- Weighted Profit Margin and Start Periods (WPMSP)

$$v(d) = p_{nm^*_n} \cdot \frac{1}{\sqrt{|S_{fn}|}} \qquad \text{extremum} = \max \quad (n \in \text{SORC}) \quad (32)$$

Other composite rules are introduced below (cf. Section 5.3) because they were derived drawing from the results of the performance evaluation of the simple rules.

It is easy to see that combining rules producing priority values from different domains may overemphasize the influence of one rule. Suppose for instance that rule NOT produces priorities from {1,...,10} while HPM returns values from {2000,...,20000}, then a composite rule with identical weights will essentially behave like HPM. If we assume all I employed rules to yield positive values and to have the same decision set D, we can counter this effect by scaling the priority values to the interval [0, 1] by

$$v_i(d) := \begin{cases} \dfrac{\min\{v_i(d') \mid d' \in D\}}{v_i(d)} & \text{if extremum}_i = \text{minimum} \\ \dfrac{v_i(d)}{\max\{v_i(d') \mid d' \in D\}} & \text{if extremum}_i = \text{maximum} \end{cases} \qquad (1 \le i \le I;\ d \in D) \quad (33)$$

Since (33) transforms the priority values of all rules i with $\text{extremum}_i = \min$ to values with $\text{extremum}_i = \max$, it suffices to consider only $\beta_i \in [0, 1]$. Each specific algorithm, derived from above algorithmic scheme by incorporating priority rules for the selection of courses and start periods, can be regarded as a mapping from the problem to the solution space. Since all priority rules are employed in a deterministic manner, each weighting vector $(\beta_1,...,\beta_I)$ can be seen to encode one particular solution.

Having laid this groundwork, we now describe a local search method which systematically determines weights guiding the computational effort to promising regions of the search space (cf. Haase 1996). The method is an iterative procedure which in each iteration constructs a number of solutions. If the best so-found solution is better than the incumbent best one, the search process is intensified in the surrounding region; otherwise, the procedure terminates.

The method proceeds by spanning an I-dimensional grid over the parameter space by defining a set of equidistant points. In each iteration, this grid, which initially covers the entire parameter space $[0,1]^I$, is refined to ever smaller subspaces. For each gridpoint, one solution is constructed. The number of gridpoints to be examined is pre-set by a parameter $\sigma \ge 2$, the so-called number of steps. Let for each iteration denote $L\beta_i$ and $U\beta_i$ the lower and upper bound of the parameter subspace of rule i, then the so-called step-width Δ_i of rule i is defined for each iteration as

$$\Delta_i := (U\beta_i - L\beta_i) / \sigma \qquad (1 \le i \le I) \quad (34)$$

One iteration of the procedure starts off with $\beta_i = 0$, $L\beta_i = 0$, and $U\beta_i = 1$ for each rule i and determines a solution. It then increments β_I by its step-width Δ_I and constructs a second solution, and so on, until eventually $\beta_I = U\beta_I$. Then β_I is reset to $L\beta_I$ whereas β_{I-1} is incremented by Δ_{I-1}. When also β_{I-1} has reached its upper bound, it is reset to its lower bound while β_{I-2} is incremented by Δ_{I-2} and so forth. The number of gridpoints considered per iteration is $(\sigma + 1)^I$. If the incumbent best solution could not be improved in an iteration, the algorithm halts. Otherwise, let denote β^*_i that weight of rule i which produced the best solution in that iteration; then new bounds are calculated from

$$L\beta_i := \max\{0, \beta^*_i - \Delta_i \cdot (\sigma-1)/\sigma\} \qquad (1 \le i \le I) \quad (35)$$

$$U\beta_i := \min\{1, \beta^*_i + \Delta_i \cdot (\sigma-1)/\sigma\} \qquad (1 \le i \le I) \quad (36)$$

the step-widths Δ_i of all rules are updated according to (34), and the next iteration starts.

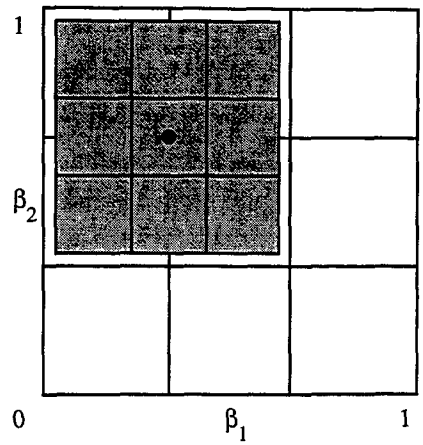

Figure 3: Exemplary Application of Local Search Method

Figure 3 illustrates the procedure for I = 2, $\sigma = 2$, $\beta^*_1 = 1/3$, and $\beta^*_2 = 2/3$ where • denotes the best weighting vector of one iteration and the shaded area the parameter subspace searched in the next iteration.

5. Performance Analysis

5.1. LTT Operational Instance

Since the foremost intention of this project was to provide efficient solutions for the planning situation of LTT, we applied our algorithms to the operational data of 1996. Acquisition of this data required substantial effort. Nearly all available data was available only in written form. A considerable part of the data, such as fixed and variable travel cost, had to be gathered for the first time. Another part, such as information on requirements of different course types, had to be restructured completely since the parameters presented in Section 3.1 are much more detailed than the information existing so far. A complete description of the operational instance of 1996 used, to which we refer as LTT96, would be beyond the scope of this contribution. Therefore, we restrict the presentation to some characteristics (cf. Table (6)).

T	CT	N	$\lvert R^\tau \rvert$	$\lvert GT \rvert$	$\lvert R^\rho \rvert$	$\lvert GR \rvert$	$\lvert R^\gamma \rvert$
3,000	79	164	82	122	3	3	6

Table 6: LTT Operational Instance - General Characteristics

The number of periods T arises from 250 working days, each comprising a day and a night shift of six lessons each. Customers demand only complete courses; the number of training days requested totals 3,318. Courses have on average a number of 55 start days; the majority of courses is required to be held within a certain month. The average number of vacation days per instructor is about 36 days while the average size of an instructor group is 3.4. The max number of courses of the same type is nine. Some statistics on the distribution of number of blocks and duration over the courses are shown in Tables (7) and (8).

Number of blocks	1-4	5-8	9-12	13-16	17-20	Average
Count of courses	56	11	53	32	12	9.19

Table 7: LTT Operational Instance - Numbers of Blocks

Duration (days)	1-10	11-20	21-30	31-40	41-50	51-57	Average
Count of courses	64	23	39	24	12	2	20.23

Table 8: LTT Operational Instance - Durations

5.2. Additional Test Instances

An experimental evaluation of algorithms aiming at producing scientifically valid insight has to be based on a representative sample of test instances. However, providing the required data turned out to be a time-consuming and costly task. Efforts to reproduce problem instances of past years failed due to the prohibitive cost involved to collect the planning data and to reconstruct and evaluate the corresponding manually constructed schedules. The special structure of the problem prevented utilizing a general-purpose instance generator. Due to the complexity of the problem, the development and implemention of a dedicated instance generator was considered beyond the scope of this project. Hence, we choose to modify the operational instance LTT96 appropriately to derive additional realistic test instances.

For the purposes of this study we assumed the load factor LF of the instructing staff to be the only factor influencing the tractability of an instance. Basically, the staff load factor could be determined from dividing the total demand by the total working capacity of all staff instructors. However, this measure could be severely distorted if some instructors are in negligible demand (as is the case in LTT96). Therefore, we calculate LF as the average of the instructor load factors lf_r over all instructors r ($r \in R^T$) as defined in (26). The staff load factor of LTT96 is 0.44. Since all courses of LTT96 had successfully been scheduled manually by LTT, we classified the operational instance as easy and used its LF-value as stepping stone for classify-

ing CSP-instances in general. Throughout this work, we distinguish three types of instances with respect to their tractability: *easy* instances where $0.0 \leq LF < 0.6$, *medium* instances where $0.6 \leq LF < 0.8$, and *hard* instances where $0.8 \leq LF \leq 1.0$.

Tractability	Average LF	Minimum LF	Maximum LF	Average Number of Training Days
Easy	*0.48*	*0.42*	*0.52*	*1,988*
Medium	0.75	0.65	0.86	3,093
Hard	0.99	0.87	1.19	4,027

Table 9: Test Instances

For each of these three classes, ten instances were generated on which the algorithms were tested. Staff qualification structure, instructor and room group composition, and course types were directly adopted from LTT96. Vacation days for the instructors were generated randomly. The planning horizon was restricted to six months. Variation between different instances was achieved by randomly generating course demand. Starting with no courses initially, courses were built by drawing random values for course type, performing office, start periods, and profit margin until the desired tractability was reached. The probabilities used were derived from the characteristics of LTT96. An overview of the test instances is provided in Table 9. Note that under this procedure the number of training days to be scheduled and thus the size of the instances increases linearly with the tractability since the staff capacity is identical for all instances.

5.3. Computational Results

In the sequel, we describe the design and results of three experiments. Due to the lack of appropriate exact methods, and with the size of operational instances foreclosing the use of standard solvers, optimum solutions to compare the heuristically found solutions are not available. An upper bound for each instance could be derived from summing up the fees of all courses. Yet, especially for the hard instances it is impossible to schedule all courses due to limited instructor capacities, so the total fees achieved may be substantially lower. Even for the easy instance LTT96, the optimum objective function value is considerably less than the above bound since for some of the courses instructors from other offices are mandatory, incurring travel cost which in turn reduces the solution value. Therefore, given a solution for a particular instance, we use its percentage deviation from the best known solution as a performance measure.

Recall that the above algorithmic scheme allows to incorporate different instantiations of course selection rule, start period selection rule, and tutor strategy. In the first experiment, we performed one run for each combination of (simple) course selection rule (cf. Section 4.2), start period selection rule (cf. Section 4.3), and tutor strategy (cf. Section 4.1). In addition, we we selected the next course to be scheduled by a pure random sampling rule (RND) assigning the same probability to each course in the decision set (Baker 1974). The results of applying RND are derived from averaging the outcome of 100 runs. The complete results, in terms of above mentioned percentage deviations, are shown in Tables 10 - 13, where the course selection rules are listed in the top row while the respective tutor strategy (TUP or TUO) and start period selection rule (ASL or ECD) tested are noted below the table.

Tutor strategy TUO is outperformed by TUP for all algorithms. Since courses may not be modified once they are scheduled, TUP uses less of the scarce instructor capacities in the first place by assigning more tutors than TUO does. With respect to the selection of start periods, ASL outperforms ECD in about 80% of the cases. This advocates to select the start periods considering the available capacities; yet more tuned rules might produce better results since ASL is a rather coarse measure. Regarding the course selection, rules WPMSP, HPM, and LDN perform best whereas ALFT does not better than pure random sampling. All algorithms become less efficient with decreasing tractability of the instances attempted.

Motivated by the promising performance of the composite course selection rule WPMSP, in a second experiment we examined another composite rule, made up of four simple ones, namely HPM, LDN, LFSP, and AIPT. The first two were chosen because of their good performance in the first experiment, LFSP since it is the only rule that distinguishes between different courses of the same type, and AIPT because it differentiates between courses that bring in the same amount of fees. The weights accorded to these four rules were determined by the local search method with a number of steps $\sigma = 3$. Also, due to the above results, from now on start periods were selected exclusively by ASL while tutors were handled only by TUP. The results of the second experiment are reported in Table 14, where the percentage deviation of the best

Instance Class	WPMSP	HPM	LDN	NOT	LFSP	AIPT	ALFT	RND
LTT96	4.46	4.60	6.04	2.48	6.05	5.14	3.78	6.81
Easy	2.58	2.25	2.55	5.36	6.27	5.93	8.32	7.50
Medium	4.10	4.44	4.67	6.54	9.19	9.20	11.62	11.23
Hard	4.87	4.93	6.54	8.65	9.20	11.41	12.42	11.22
Average	3.87	3.90	4.63	6.71	8.15	8.73	10.56	9.88

Table 10: Percentage Deviations of Priority Rules
- All Instances, All Course Selection Rules, TUP, ASL

Instance Class	WPMSP	HPM	LDN	NOT	LFSP	AIPT	ALFT	RND
LTT96	4.18	5.06	6.08	8.16	8.92	5.11	12.04	11.00
Easy	2.67	3.44	3.47	3.76	6.17	5.00	7.89	7.62
Medium	4.73	5.14	5.58	6.89	11.09	9.19	13.31	12.26
Hard	4.98	5.03	6.81	7.73	10.52	10.21	14.05	12.36
Average	4.13	4.56	5.31	6.19	9.25	8.04	11.76	10.76

Table 11: Percentage Deviations of Priority Rules - All Instances, All Course Selection Rules, TUP, ECD

Instance Class	WPMSP	HPM	LDN	NOT	LFSP	AIPT	ALFT	RND
LTT96	7.88	11.45	9.40	7.95	13.31	8.71	15.85	14.10
Easy	6.80	6.84	6.10	9.39	12.83	10.82	13.62	14.20
Medium	13.69	14.62	15.57	14.60	18.78	19.34	21.39	21.11
Hard	15.87	15.98	18.13	18.30	20.91	22.05	24.20	22.97
Average	11.98	12.45	13.14	13.90	17.37	17.12	19.61	19.25

Table 12: Percentage Deviations of Priority Rules - All Instances, All Course Selection Rules, TUO, ASL

Instance Class	WPMSP	HPM	LDN	NOT	LFSP	AIPT	ALFT	RND
LTT96	10.33	10.42	9.57	11.93	15.80	11.22	17.78	15.62
Easy	7.24	7.09	6.57	9.15	12.00	11.11	14.34	14.14
Medium	13.92	15.08	16.16	15.23	20.06	19.36	22.23	21.21
Hard	16.49	16.78	18.67	19.63	20.50	22.29	25.02	22.73
Average	12.48	12.90	13.66	14.58	17.47	17.38	20.44	19.24

Table 13: Percentage Deviations of Priority Rules - All Instances, All Course Selection Rules, TUO, ECD

found solution compared to the best known one, the percentage of demand actually met, the percentage of courses scheduled in standard sequence, the average number of iterations required, and the computation times (using an implementation in Borland C on a 486DX, 100 MHz personal computer) are given.

Instance Class	Minimum Deviation (%)	Service Level (%)	Courses in Standard Sequence (%)	Iterations Required	CPU-Time (s)
LTT96	0.00	100.00	95.12	7.0	4,499.0
Easy	0.00	99.76	99.45	3.6	1,801.2
Medium	0.00	92.87	90.62	3.2	4,541.0
Hard	0.00	82.99	89.43	4.6	11,196.0

Table 14: Results of Local Search Method - All Instances, (HPM, LDN, LFSP, AIPT), TUP, ASL, $\sigma = 3$

Using this composite rule produced the best solution for each instance attempted, hence the deviations are zero. As on the other hand the deviations of all simple rules grow with increasing intractability, we infer that the local search method becomes more efficient on harder instances. Note that for the easy instances even the algorithms using simple course selection rules achieved very high service levels. The computation times of the composite algorithm are about six times higher on the hard than on the easy instances, although the hard instances comprise on average only twice as much training days as the easy ones (cf. Table 9). This higher computational effort is caused by two effects. First, for hard instances the number of courses scheduled in a non-standard sequence is higher: For these courses, at least level 1, and possibly also levels 2 and 3, of the algorithmic scheme failed, increasing computation times. Second, also the number of failed courses is higher: For these courses, all four levels failed, driving the computation times even higher.

Iteration	Minimum Deviation (%)	Average Deviation (%)	Solutions Scheduling All Courses (%)
1	1.55	3.92	1.95
2	0.69	3.26	15.23
3	0.67	1.94	79.30
4	0.36	1.60	78.90
5	0.12	1.28	82.03
6	0.00	1.10	77.21
7	0.00	1.07	79.87

Table 15: Results of Local Search Method - LTT96, (HPM, LDN, LFSP, AIPT), TUP, ASL, $\sigma = 3$

Table 15 provides a more detailed look into the local search method: its performance in the second experiment is examined separately for each iteration it took on LTT96. The percentage deviation for the best solution found and as an average over all solutions found, along with the percentage of schedules which comprise all requested courses are given for each iteration. These results indicate the capability of the method to find good operational schedules.

In a third experiment, carried out on LTT96, we investigated another eight composite course selection rules, made up of promising combinations of three simple rules. In this case, we had the local search method search a finer grid with $\sigma = 8$, amounting to 729 solutions per iteration. Since the weight vector of the best solution found in the previous experiment was ($\beta_{HPM} = 0.04$, $\beta_{LDN} = 0.04$, $\beta_{LFSP} = 0.81$, $\beta_{AIPT} = 0.8$), we considered mostly combinations including the latter two rules. The rules used and the corresponding results are summarized in Table 16.

Employed Rules	Maximum Deviation (%)	Average Deviation (%)	Solutions with All Courses Included (%)	CPU-Time (s)
HPM, LFSP, AIPT	0.42	1.67	99.72	15,145
LDN, LFSP, AIPT	0.44	2.73	45.81	15,009
LFSP, AIPT, ALFT	0.48	1.95	100.00	46,759
HPM, NOT, LFSP	0.81	2.76	0.00	27,492
HPM, NOT, AIPT	1.02	2.76	0.14	16,615
HPM, LDN, ALFT	1.44	3.84	0.00	30,302
HPM, NOT, ALFT	1.67	3.99	0.00	18,343
NOT, LSFP, ALFT	1.76	4.31	0.00	19,550

Table 16: Results of Local Search Method - LTT96, TUP, ASL, $\sigma = 8$

Clearly, rules LFSP and AIPT demonstrate the best performance. Note from columns two and three that an increase of the maximum profit margin is accompanied by an increase of the average profit margin over the examined part of the solution space. This consolidates the hypothesis that the local search method indeed determines parameter subspaces corresponding to "good" regions of the solution space. Indeed, this advocates storing such parameter subspaces and using them as starting regions when tackling planning instances of future years. In column four, the service level is reported. These numbers are of high significance for the operational planning of LTT since a high service level is an important requirement for achieving and maintaining market share. With regard to this objective, rules LFSP and AIPT produce the best results. Since we deal with a problem of long-term planning, computation times

as given in the last column seem to be acceptable. Still, it is noteworthy that another experiment with a course selection rule composed of the two best-performing simple rules of the previous experiment, viz. LFSP and AIPT, and $\sigma = 3$ produced a schedule with deviation of only 0.6% off the best known solution value within 330 seconds.

Finally, we compare our best solution for LTT96, as obtained in the third experiment, to the schedule the LTT planners constructed manually. Table 17 summarizes some characteristic measures of both schedules. The gap between an upper bound of DM 1,432,800, which has been calculated as indicated above, and the actually found objective function value is about 25% for the manually and 7% for the algorithmically constructed solution. Since all courses could be scheduled successfully in both solutions, this gap, which amounts to about DM 275,000, can only be imputed to differing travel and tutor costs. As can be seen from rows three and four, the travel cost incurred by the manual schedule is substantially higher whereas the algorithmic schedule uses about twice as much tutors as the one of LTT. These findings demonstrate the suitability of the level scheme and the tutor strategy TUP which begins to check all possibilities to schedule courses without having to assign rather expensive instructors from other offices. This interpretation is supported by the findings in rows six and seven. Since about 93% of all courses are performed in Frankfurt, the extent to which staff from the other two offices in Hamburg (HAM) and Berlin (BER) are used is a meaningful indicator of the travel cost entailed. It turns out that the manual schedule makes about four times as much use of such instructors as the algorithmic one. The last two rows provide some insight into the workload distribution of the staff. On average, the number of days accorded to instructors is about the same for both schedules but the LTT-schedule assigns more than 10% more lessons to them than our schedule. Clearly, this is due to the higher use our schedule makes of tutors.

	LTT	Algorithm
Total profit margin	1,061,747	1,336,919
Fixed travel cost	108,800	16,000
Variable travel cost	238,733	32,361
Tutor cost	23,520	47,520
Avg. number of days assigned (staff HAM/BER)	44.13	11.2
Avg. number of lessons assigned (staff HAM/BER)	224.2	54.77
Avg. number of days assigned (all staff)	71.24	69.03
Avg. number of lessons assigned (all staff)	367.66	330.11

Table 17: Best Algorithmic vs. Manually Constructed Schedule

6. Summary and Outlook

In this contribution, we dealt with the course scheduling problem faced by LTT. We developed a heuristic algorithmic scheme and presented several simple priority rules to be incorporated into the scheme as well as a local search method able to determine well-suited weights for weighted composite rules. The operational planning situation of 1996 served as our major test instance; additional test instances were constructed by modifying this data. Several computational experiments were carried out to evaluate the performance of the algorithms. It turned out that the best so-found schedule is substantially better in terms of profit margin incurred than the solution manually constructed by LTT. Also, the time required to construct an operational schedule is heavily reduced by the algorithms presented. However, ongoing analysis will have to show whether the algorithmically derived solutions can be implemented without modifications. For example, the balanced distribution of preparation times and several other restrictions of minor importance have not been considered explicitly in our model.

An important issue for the practical usability of this work is the integration of the algorithms into a decision support system with database access and comfortable interactive dialogue features. Also, the algorithms will have to be complemented by re-scheduling procedures to adapt schedules to short-term changes. Benefits expected from such a system include increase of profit margin and reduction of travel cost and times, faster and thus less costly generation of operational schedules, usage of resources closer to capacity, ability to react more flexible to short-term changes, and provision of detailed statistical information for planners, staff, and customers. The development of such a system is in progress.

Acknowledgement

As anyone who has worked in industrial applications would know, many people were involved with this project. In particular, we wish to mention Christian Hollmann whose expertise was instrumental and Dieter Kohrs who provided the required data. We also recognize the managerial support of Hans Reichow. Finally we thank Andreas Drexl for his encouraging support and his constructive remarks on an earlier version of this paper.

References

ABRAMSON, D. (1991), "Constructing school timetables using simulated annealing: Sequential and parallel algorithms", Management Science 37, pp. 98-113.

ALVAREZ-VALDES, R., G. MARTIN, AND J.M. TAMARIT (1996), "Constructing good solutions for a school timetabling problem", Journal of the Operational Research Society 47, pp. 1203-1215.

AUBIN, J. AND J.A. FERLAND (1989), "A large scale timetabling problem", Computers and Operations Research 16, pp. 67-77.

BAKER, K.R. (1974), *Introduction to sequencing and scheduling*, Wiley, New York.

BÖTTCHER, J., A. DREXL, R. KOLISCH, AND F. SALEWSKI (1996), "Project scheduling under partially renewable resource constraints", Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel 398.

CANGALOVIC, M. AND J.A.M. SCHREUDER (1991), "Exact colouring algorithm for weighted graphs applied to timetabling problems of different length", European Journal of Operational Research 51, pp. 248-258.

CANGALOVIC, M. AND J.A.M. SCHREUDER (1992), "Modelling and solving an acyclic multiperiod timetabling problem", Discrete Applied Mathematics 35, pp. 177-195.

COSTA, D. (1994), "A tabu search algorithm for computing an operational timetable", European Journal of Operational Research 76, pp. 98-110.

DE GANS, O.B. (1981): "A computer timetabling system for secondary schools in the Netherlands", European Journal of Operational Research 7, pp. 175-182.

EGLESE, R.W. AND G.K. RAND (1987), "Conference seminar timetabling", Journal of the Operational Research Society 38, pp. 591-598.

HAASE, K. (1996), "Capacitated lot-sizing with sequence dependent setup costs", Operations Research Spektrum 18, pp. 51-59.

HAASE, K. AND J. LATTEIER (1996), "Deckungsbeitragsorientierte Lehrgangsplanung bei Lufthansa Technical Training GmbH", Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel 395.

HAASE, K., J. LATTEIER, AND A. SCHIRMER (1997), "Course planning at Lufthansa Technical Training - Constructing more profitable schedules", Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel 442.

HERTZ, A. (1992), "Finding a feasible course schedule using tabu search", Discrete Applied Mathematics 35, pp. 255-270.

KANG, L. AND G.M. WHITE (1992), "A logic approach to the resolution of constraints in timetabling", European Journal of Operational Research 61, pp. 306-317.

KELLEY, J.E. (1963), "The critical-path method: Resources planning and scheduling", in: *Industrial scheduling*, Muth, J.F. and G.L. Thompson (eds.), Prentice-Hall, Englewood Cliffs, NJ, pp. 347-365.

KOLISCH, R. (1995), *Project scheduling under resource constraints - Efficient heuristics for several problem classes*, Physica, Heidelberg.

SAMPSON, S.E., J.R. FREELAND, AND E.N. WEISS (1995), "Class scheduling to maximize participant satisfaction", Interfaces 25:3, pp. 30-41.

TRIPATHY, A. (1984), "School timetabling - a case in large binary integer linear programming", Management Science 30, pp. 1473-1489.