Drexl, Andreas; Frahm, Johannes; Salewski, Frank

Audit-staff scheduling by column generation

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 464

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

Suggested Citation: Drexl, Andreas; Frahm, Johannes; Salewski, Frank (1997) : Audit-staff scheduling by column generation, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 464, Universität Kiel, Institut für Betriebswirtschaftslehre, Kiel

This Version is available at:
https://hdl.handle.net/10419/177322

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 464

Audit–Staff Scheduling by Column Generation

A. Drexl, J. Frahm, F. Salewski

November 1997

Andreas Drexl, Johannes Frahm, Frank Salewski
Institut für Betriebswirtschaftslehre, Lehrstuhl für Produktion und Logistik
Christian–Albrechts–Universität zu Kiel, Olshausenstr. 40, 24118 Kiel
email: {Drexl, jf, Salewski}@bwl.uni-kiel.de
URL: http://www.wiso.uni-kiel.de/bwlinstitute/Prod
ftp://ftp.wiso.uni-kiel.de/pub/operations-research

Abstract

When scheduling its audit–staff, the management of an auditing firm encompasses a number of decisions. These may be grouped into several categories which differ markedly in terms of organizational echelon involved, length of the planning horizon and the planning periods, degree of aggregation of the audit tasks, degree of detail of the required information, and decision objective. In this paper we consider the medium–term problem the focus of which is to assign teams of auditors to the engagements. It constructs a schedule by determining the workload per auditor and week over a planning horizon of between three and twelve months. The objective of the model is to maximize the sum of the preferences of all assigned teams. As restrictions job completion, precedence and resource constraints have to be met.

The problem is formulated as a set partitioning problem. We solve the LP–relaxation of the set partitioning problem by column generation in order to compute tight upper bounds. Fortunately, the upper bounds obtained with the column generation approach are much tighter than those obtained by solving the LP–relaxation of a 'standard' project scheduling model formulation. Furthermore, the column generation process is an order of magnitude faster than solving the LP–relaxation of the standard model formulation. Finally, the optimal solution of the continuous relaxation of the set partitioning problem is integral frequently and, hence, we have an optimal solution of the overall integer programming problem.

KEYWORDS: AUDIT–STAFF SCHEDULING, ASSIGNMENT OF TEAMS OF AUDITORS TO ENGAGEMENTS, SET PARTITIONING/COLUMN GENERATION

1 Introduction

When scheduling its audit–staff, the management of an auditing firm encompasses a number of decisions. These may be grouped into several categories which differ markedly in terms of organizational echelon involved, length of the planning horizon and the planning periods, degree of aggregation of the audit tasks, degree of detail of the required information, and decision objective. However, traditional audit–staff scheduling models (Balachandran and Zoltners 1981, Chan and Dodin 1986, Gardner, Huefner, and Lofti 1990, Dodin and Chan 1991, Drexl 1991, Dodin, Elimam, and Rolland 1996, Dodin and Elimam 1997) are single–level models which try to construct a direct assignment of auditors to tasks and periods. To facilitate algorithmic treatment, all these models are more or less gross simplifications of practical planning situations.

These observations led us to conduct a survey among the 200 biggest certified public accountant (CPA) firms in Germany. Based upon its results we formulated an hierarchical model (Salewski and Drexl 1993, Salewski 1995) comprising three levels: The medium–term planning assigns teams of auditors to the engagements; it constructs a schedule by determining the workload per auditor and week over a planning horizon of between three and twelve months. The medium–to–short–term planning disaggregates the results of the first level for one week and all auditors; the outcome is a schedule for each auditor that covers — on the basis of periods of four hours — all engagements in which he is involved

in the considered week. The short–term planning is based upon the results of the second level for one week and one engagement; it assigns the auditors involved in the auditing of that engagement to the corresponding audit tasks and schedules these tasks to periods of one hour. Here, we will focus on the first, that is, the medium–term level.

The paper is organized as follows: In Section 2 we define the problem formally and investigate where it is positioned in the context of audit–staff/project scheduling. In Section 3 the problem is reformulated as a set partitioning problem with an exponential number of columns. The LP–relaxation of this model can be solved to optimality by column generation. Next we show in Section 4 that the columns of the the LP–relaxation can be efficiently computed by means of a shortest path model. The description of the test bed is provided in Section 5. Section 6 presents the results of an in–depth computational study. Finally, Section 7 provides a brief summary, along with our conclusions.

2 Problem Setting

The Medium–Term Audit–Staff Scheduling Problem (MASSP) may be characterized by the following assumptions (cp. Salewski, Schirmer, and Drexl 1997 also):

- A firm employs one or more *auditors*, which have to audit one or more *engagements* within a given *planning horizon* of normally 13, 26, or 52 weeks.
- Each engagement is made up of one or more *phases*, e.g. preliminary, intermediate, and final audit, which in turn can be decomposed into one or more *subphases*. Some subphases may not be executed before the completion of certain others (*predecessors*). More exactly: the phases of each engagement as well as the subphases of each phase must be processed in a strictly linear order which implies that each subphase except of the first subphase of the first phase of each engagement possesses exactly one predecessor. Work on some phases may not commence before a specific *release time*, or may have to be completed by a certain *deadline*.
- The *availability* of some auditors may be restricted in certain periods, e.g. due to holidays or vacation. In addition, for some periods a client may want to limit the time during which the auditing takes place (*maximum processing time*), e.g. due to vacation periods or stock–taking activities.
- Often an engagement could be audited by several alternative audit teams (*modes*). Different team compositions will result in different auditor processing times. Usually some modes will be preferable to others: Factors influencing the suitability of an auditor for a specific engagement are e.g. qualification level, industry experience, familiarity with the clients business, and degree of difficulty of the audit tasks. The preferability of a mode as a whole may e.g. be linked to the total processing time needed. Hence, a preference value will be assigned to each mode.
- Finally, mode–dependent *minimum and maximum time–lags* may be given between subsequent subphases (which thus belong to the same engagement).

The objective then is to assign the overall best–suited teams to the engagements (*mode assignment with maximization of preferences*), and to determine when the individual subphases are to be executed (subphase scheduling).

The problem parameters of the MASSP are summarized as follows:

A	:	number of auditors, indexed by a
E	:	number of engagements, indexed by e
v^{em}	:	preference value corresponding to the processing of engagement e in mode m
C_{at}	:	capacity of auditor a in period t
D_{et}	:	maximum processing time of engagement e in period t
δ_{ep}	:	deadline of phase (e,p)
k_{epsma}	:	time auditor a needs to process subphase (e,p,s) in mode m (capacity usage)
λ_{ep}	:	release time of phase (e,p)
M_e	:	number of modes of engagement e, indexed by m
P_e	:	number of phases of engagement e, indexed by p
$q_{epsp's'm}$	:	minimum (finish–to–start) time–lag between subsequent
	:	subphases (e,p,s) and (e,p',s') when processing e in mode m
$\hat{q}_{epsp's'm}$	:	maximum (finish–to–start) time–lag between subsequent
	:	subphases (e,p,s) and (e,p',s') when processing e in mode m
S_{ep}	:	number of subphases of phase (e,p), indexed by s; w.l.o.g. each subphase has a duration of one period
T	:	number of periods, indexed by t
V_{eps}	:	set of all immediate predecessors of subphase (e,p,s)

Note, precedence relations expressed via V_{eps} exist only between subphases belonging to the same engagement. This characteristic is called 'isolating' in Salewski, Schirmer, and Drexl (1997).

The problem under consideration is formulated as a binary optimization problem in Salewski, Schirmer, and Drexl (1997). Furthermore, it is shown, that the MASSP is a special case of the more general project scheduling problem with resource and mode identity constraints. In addition, it is proven that the (feasibility variant of the) MASSP is (strongly $\mathcal{NP}$–complete) strongly $\mathcal{NP}$–hard. In order to keep this paper self–contained we reproduce the project scheduling–based binary optimization model in Appendix A.

Throughout this paper we make use of the illustrative example which is provided in Appendix B.

3 Set Partitioning Model

The basic idea is to iteratively compute sequences for each of the engagements by means of a shortest (in fact a longest) path model. From the set of sequences on hand those are chosen via a set partitioning model which respect the capacity constraints of the auditors (in the LP–relaxation). A description of basic column generation techniques can be found in, e.g., Bradley, Hax, and Magnanti 1977.

In order to describe the set partitioning model formally we use the following parameters and variables:

$e(i)$	:	engagement e column i is associated with
$\mathcal{S}^e$	:	set of columns representing sequences for engagement e, index i
b_{eiat}	:	1, if engagement e is processed within sequence i by auditor a and finished in period t
c_i	:	objective function coefficient of sequence i related to engagement $e(i)$
y_i	:	1, if sequence i is part of the optimal solution (0, otherwise)

Based on these definitions the (restricted) master problem can be stated by equations (1)–(4) as a set partitioning model.

$$\max \sum_{e=1}^{E} \sum_{i \in \mathcal{S}^e} c_i y_i \tag{1}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{S}^e} y_i = 1 \qquad e = 1, \ldots, E \tag{2}$$

$$\sum_{e=1}^{E} \sum_{i \in \mathcal{S}^e} b_{eiat} y_i \leq C_{at} \qquad a = 1, \ldots, A,\ t = 1, \ldots, T \tag{3}$$

$$y_i \in \{0, 1\} \qquad e = 1, \ldots, E,\ i \in \mathcal{S}^e \tag{4}$$

The objective (1) is to select a subset of columns at minimum costs. Equations (2) make exactly one sequence per engagement to be part of the solution, while inequalities (3) require to respect the capacity constraints of the auditors in each of the periods of the planning horizon. Finally, restrictions (4) define the decision variables to be binary–valued.

4 Shortest Path Model

For each tupel of engagement and mode (e, m), sequences are computed by solving shortest path problems. Apparently, these constraints essentially render each (e, m) subgraph, $1 \leq e \leq E, 1 \leq m \leq M_e$, to consist of nodes $\boxed{p, s, t}$ solely (cp. Figure 1 also).

In order to simplify the presentation we relabel the nodes $h \leftarrow (p, s, t)$ and $j \leftarrow (p', s', t')$ similarly to what is done in equation (15) in Appendix A. Obviously, this relabeling has to be done for each (e, m) subgraph. Now, let denote

$\mathcal{N}^{em}$	:	set of nodes of the graph associated with engagement e and mode m
$\mathcal{A}^{em}$	:	set of arcs of the graph associated with engagement e and mode m
$p(j)$	:	phase p node $j \in \mathcal{N}^{em}$ is associated with
$s(j)$	:	subphase s node $j \in \mathcal{N}^{em}$ is associated with
$t(j)$	:	period t node $j \in \mathcal{N}^{em}$ is associated with

Figure 1 illustrates the preliminary shortest path graph for enagement $e = 2$ and mode $m = 2$ of the instance provided in Appendix B. This shortest path graph shows that — for a given tupel (e, m) — two nodes $(p(h), s(h), t(h))$ and $(p(j), s(j), t(j))$ are connected by

an arc only if the conditions (6) to (11) are met. Note that nodes can be connected only if h and j belong to the same engagement e ('isolating' precedence structure — cp. (5)) and if they are processed in the same mode.

$$(e, p(h), s(h)) \in V_{e,p(j),s(j)} \quad \text{(precedence relations)} \tag{5}$$

$$t(j) - t(h) \geq q_{e,p(h),s(h),p(j),s(j),m} + 1 \quad (h, j \in \mathcal{A}^{em}, h \in V_{e,p(j),s(j)} \text{ — min time lag}) \tag{6}$$

$$t(j) - t(h) \leq \hat{q}_{e,p(h),s(h),p(j),s(j),m} + 1 \quad (h, j \in \mathcal{A}^{em}, h \in V_{e,p(j),s(j)} \text{ — max time lag}) \tag{7}$$

$$\sum_{a=1}^{A} k_{e,p(h),s(h),m,a} \leq D_{e,t(h)} \quad (h \in \mathcal{N}^{em} \text{ — maximum processing time}) \tag{8}$$

$$\sum_{a=1}^{A} k_{e,p(j),s(j),m,a} \leq D_{e,t(j)} \quad (j \in \mathcal{N}^{em} \text{ — maximum processing time}) \tag{9}$$

$$\lambda_{e,p(h)} \leq t(h) \quad \text{and} \quad \lambda_{e,p(j)} \leq t(j) \quad (h, j \in \mathcal{N}^{em} \text{ — release dates}) \tag{10}$$

$$\delta_{e,p(h)} \geq t(h) \quad \text{and} \quad \delta_{e,p(j)} \geq t(j) \quad (h, j \in \mathcal{N}^{em} \text{ — deadlines}) \tag{11}$$

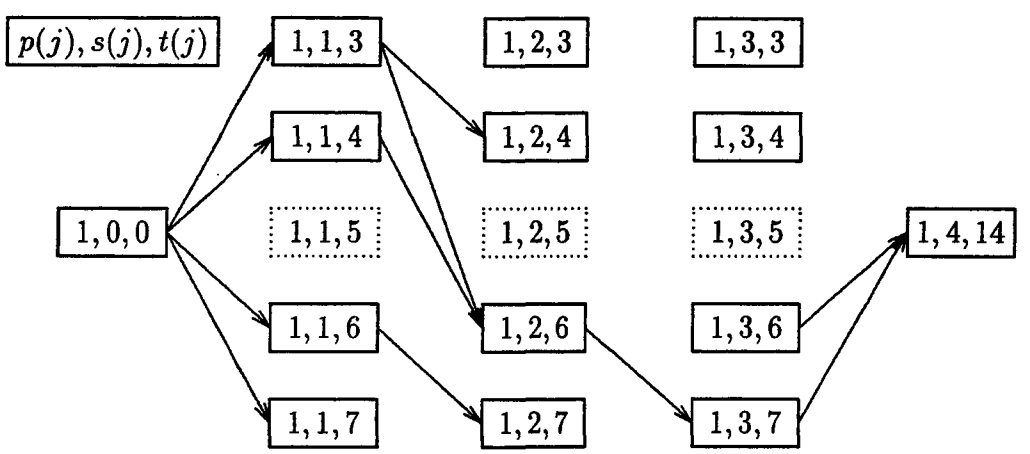

Figure 1: Preliminary Shortest Path Graph — $e = 2$ and $m = 2$

The following comments shall be made with respect to Figure 1 and, in addition, Figure 2:

- The nodes (1,0,0) and (1,4,14), where we set $4 = S_{e,P_e} + 1 = S_{21} + 1$ and $14 = T + 1$, are dummy source and sink nodes, respectively.
- The nodes (1,1,5), (1,2,5) and (1,3,5) are dotted, because $D_{25} = 0$ does not allow to schedule these subphases in period $t = 5$, and, hence, they can be eliminated. Clearly, incident arcs have not to be considered also.
- There must be no arc connecting nodes (1,1,3) and (1,2,7) because of the maximal time lag.

- The dashed nodes (1,2,3), (1,3,3) and (1,3,4) cannot be reached because of minimal time–lags and, henceforth, are eliminated.
- There is no outgoing arc from the dashed nodes (1,1,7) and (1,2,7) because of minimal time lags in connection with the deadline. Removing node (1,2,7) in turn produces node (1,1,6) to have no successor node and, hence, it can be eliminated also. Similarly, for node (1,2,4). Likewise, the dashed node (1,3,6) has no ingoing arc and can be eliminated too.
- Summarizing, Figure 2 provides the reduced shortest path graph where all the dotted and dashed nodes and the incident (dashed) arcs have been eliminated.

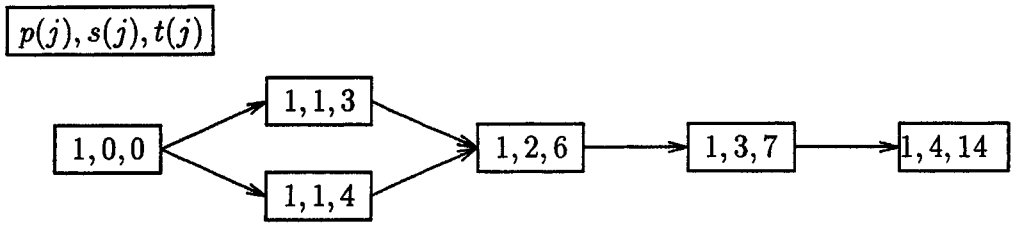

Figure 2: Reduced Shortest Path Graph — $e = 2$ and $m = 2$

Now we are going to explain how the arc weights are calculated. Without loss of generality, consider any engagement–mode–tupel (e, m), i.e. subgraph of the overall shortest graph. Furthermore, consider any pair of related nodes $h \in \mathcal{N}^{em}$ and $j \in \mathcal{N}^{em}$ of the reduced shortest path (sub–)graph and, hence, the arc $(h, j) \in \mathcal{A}^{em}$ connecting both nodes. Now, let denote

d_{hj}^{em}	:	original weight of arc $(h, j) \in \mathcal{A}^{em}$
g_{hj}^{em}	:	updated weight of arc $(h, j) \in \mathcal{A}^{em}$
μ_e	:	dual variable associated with the one sequence per engagement constraint (2), $\mu_e \in \mathbb{R}$
π_{at}	:	dual variables associated with the capacity constraints (3), $\pi_{at} \geq 0$
x_{hj}^{em}	:	1, if arc $(h, j) \in \mathcal{A}^{em}$ is element of the shortest path (0, otherwise)

Then equation (12) formally defines the original weight d_{hj}^{em} of the arcs. Obviously, all arcs have to be initialized to zero except the arcs emanating from the single source node $q = (1, 0, 0)$. This way the constant preference v^{em} for each engagement–mode–tupel (e, m) is taken into account appropriately.

$$d_{hj}^{em} = \begin{cases} v^{em}, & (q, j) \in \mathcal{A}^{em} \\ 0 \quad , & \text{otherwise} \end{cases} \tag{12}$$

Finally, equation (13) explains how to calculate the weights g_{hj}^{em} of all arcs $(h, j) \in \mathcal{A}^{em}$ taking the dual variables π_{at} into account.

$$g_{hj}^{em} = d_{hj}^{em} - \sum_{a=1}^{A} \sum_{(h,j)\in\mathcal{A}^{em}} k_{e,p(h),s(h),m,a}\, \pi_{a,t(h)} \tag{13}$$

Then the objective function of the shortest path model for engagement e can be stated by equation (14).

$$Z^e = \max\left\{ \sum_{(h,j)\in\mathcal{A}^{em}} g_{hj}^{em}\, x_{hj}^{em} - \mu_e \mid m = 1,\ldots,M_e \right\} \tag{14}$$

Note that the shortest path graph is acyclic with node weights $g_{hj}^{em} \in \mathbb{R}$. Because of the topological structure, the shortest path problems are solvable in linear time.

Apparently, pricing out occurs if $\max\{Z^e \mid 1 \leq e \leq E\} \leq 0$. This is accomplished by computing the shortest path in the overall shortest path graph comprising all the engagements. In our implementation, we compute at most E columns per iteration, one for each engagement e with $Z^e > 0$ (multiple pricing).

A step–by–step description of the overall set partitioning/column generation algorithm shall now be given where we use the following notation:

$\mathcal{S}$	:	current set of columns, i.e. $\mathcal{S} = \{\mathcal{S}^1 \cup \ldots \cup \mathcal{S}^e \cup \ldots \cup \mathcal{S}^E\}$
$SPP(\mathcal{S})$	:	(restricted) master problem/set partitioning model defined for set $\mathcal{S}$
$CG(e)$	:	column generation/shortest path model for engagement e
$Z_{CG(e)}^{(\mathcal{S})*}$	:	optimal objective function value of $CG(e)$ defined for the current set of columns $\mathcal{S}$

Algorithm

1. Initialize $\mathcal{S}^e, e = 1,\ldots,E$.
2. Solve the LP–relaxation of the set partitioning model $SPP(\mathcal{S})$; $e = 0$.
3. $e = e + 1$.
4. Solve $CG(e)$; if $Z_{CG(e)}^{(\mathcal{S})*} > 0$ then set $\mathcal{S} = \mathcal{S} \cup (|\mathcal{S}| + 1)$.
5. If $e < E$ then go to Step 3.
6. If at least one $e \in \{1,\ldots,E\}$ with $Z_{CG(e)}^{(\mathcal{S})*} > 0$ has been computed then go to Setp 2.
7. Stop.

Apparently, our algorithm generates at most E columns per iteration (multiple pricing). In Section 6 we will show, that this variant produces slightly more columns than the single pricing (i.e. one column per iteration) counterpart but is faster on the average because less LPs have to be solved.

5 Test Bed

The set of instances which is used for experimental purposes is identical to the one defined in Salewski, Schirmer, and Drexl (1997). In order to keep this paper self–contained we outline in the following the procedure followed for generating a sample of test instances of practical relevance. We also describe the design of an extensive experimental study conducted, along with the definitions of the performance measures used to evaluate the results of the study.

Even in current literature, the systematic generation of test instances does not receive much attention. For the well–researched field of project scheduling, Kolisch, Sprecher, and Drexl (1995) report that "very little research concerned with the systematic generation of benchmark instances has been published. [...] most efforts are only briefly described."

Generally, two possible approaches can be found adopted in literature when having to come up with test instances. 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. Even the 110 instances of the widely known Patterson–set (Patterson 1984), which have become a quasi benchmark in project scheduling, have been collated from different sources, rather than being generated from a systematic design. 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. A detailed such procedure for generating project scheduling instances has been proposed by Kolisch, Sprecher, and Drexl (1995)European Journal of Operational Research. 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 the strengths of both approaches while avoiding their drawbacks. Within the cited survey among the 200 biggest CPA firms in Germany, we asked the respective official in charge of staff planning (if existent) or one of the firm's partners to provide details about length of planning horizon, number of auditors, number and structure of audit engagements, auditor working capacities (working hours per day or week) and possible variations therein (e.g. due to vacation, training), etc. In addition, we carried out interviews with several experts in the field of auditing to clarify our understanding of the peculiarities of the auditing sector. Then, to ensure a systematic and consistent generation of the instances, for each of the parameters of the MASSP a domain and a discrete distribution function on the domain were defined, based upon the survey and the interview results. From these definitions, a test bed of representative instances was generated randomly, using a classification scheme to build instances with specific properties. In this way we tried to construct instances reflecting the specifics of audit–staff scheduling in the industry as closely as possible, yet to employ a systematic design for the generation procedure.

We assumed that only two instance–related factors do have a major influence on the performance of a solution method, viz. the size and the tractability of the instance attemp-

ted. Although the size of an instance is determined by the length of the planning horizon, the number of subphases, and the number of modes, statistical analyses of the survey results found all these to depend on the length of the planning horizon. In the sequel, three types of instances will be distinguished with respect to their size: the planning horizon equals 13 weeks for *small* instances (13 weeks with up to 30 auditors and 95 engagements), 26 weeks for *medium–size* instances (with up to 55 auditors and 280 engagements), and 52 weeks for *large* instances (with up to 125 auditors and 880 engagements). In addition, very small instances (13 weeks with up to 6 auditors and 10 engagements) were generated. While these instances are too small to bear practical relevance, they can be solved to optimality with standard MIP–solvers and thus can be used as benchmarks.

The *tractability* of an instance intends to reflect how easy or difficult it is to solve. In our study, the auditor capacities are assumed to be the only factor influencing the tractability of an instance: the higher the auditor capacities are, the easier the corresponding instance is ceteris paribus to solve since its solution space becomes larger. Accordingly, the auditor capacities are calculated from the average expected demand, adjusted by a multiplicative factor RS (resource strength). Throughout this work, three types of instances will be distinguished with respect to their tractability: *easy* instances where RS is taken equal to 3.5, *medium* instances where RS equals 2.5, and *hard* instances where RS is 1.5.

Clearly, the performance of an algorithm cannot be evaluated from running it on infeasible instances. It is therefore noteworthy that, in spite of the strong $\mathcal{NP}$–completeness of the associated feasibility problem, it was possible to rig up the design of the (complicated) generation procedure in a way guaranteeing that for each constructed instance there exists at least one provably feasible solution.

Due to the computational effort required to attempt a sample of all sizes, the scope of the experiment was limited to include only small and very small instances. Furthermore, for instances of these sizes lower and upper bounds are available from literature in order to benchmark the results obtained with the set partitioning/column generation approach.

6 Computational Results

The methods described earlier have been imlemented using AMPL (cp. Fourer, Gay, and Kernighan 1993) and the CPLEX callable library (cp. Bixby and Boyd 1996) on an IBM RS 6000 F40 workstation with 192 MB RAM.

Each instance has been solved starting with E initial columns, one for each engagement e. The computational results of our experiments are summarized in Tables 1 and 2. The CPU–times required by our experiments are summarized in Tables 3 and 4. The symbols have the following meaning:

no.	:	identification number of instances
easy	:	instances whose tractability is easy (likewise medium and hard)
LB	:	best feasible solution (lower bound) known so far (computed with the tabu search procedure of Salewski 1996)
CG	:	objective function value computed by column generation
$\star$	:	column generation solution integral
UB	:	optimal objective function value of the LP–relaxation of the model (16) to (22) (upper bound)
CG	:	CPU–time in sec required by the CPLEX solver for the column generation process (without AMPL times)
LP	:	CPU–time in sec required by CPLEX for the solution of the LP–relaxation of the model (16) to (22)

The result tables can be interpreted as follows:

- The set partitioning/column generation approach produces an integral solution for almost all of the 30 very small instances. These results verify the lower bounds produced with tabu search to be very good also. Moreover, the LP–relaxation of the model (16) to (22) is tight for the very small instances.

- For the small instances only one data set can be solved to optimality by the column generation approach. Eight (one) of the easy (medium) instances are (is) solved to optimality also because the lower and the column generation–based upper bounds coincide.

- In general the upper bounds produced by set partitioning/column generation are far better than the upper bounds of the model (16) to (22). This is due to the following fact: In the LP–relaxation of the model (16)–(22) implicitely the constraints (21) imposed by the maximum processing times D_{et} per engagement e and per period t are relaxed also. On the other hand, these constraints are taken into account by conditions (8) and (9) when constructing the shortest path graph, and, hence, cannot be relaxed implicitely when solving the LP–relaxation of the set partitioning model.

- While the LP–relaxation of the model (16) to (22) can be solved in zero sec for the very small instances, the small ones already require a considerable amount of CPU–time. Especially for the hard instances the CPU–times increase drastically while the quality of the upper bounds deteriorates.

Table 5 shows the sizes of the last master problems in terms of the columns generated. Apparently, for the small instances the number of columns generated increases with increasing problem hardness. That is, easy instances need less columns than the medium ones which in turn need less than the hard ones.

For getting the results presented so far, we generated at most E columns per iteration, one for each enagement e with $Z^e > 0$ (multiple pricing). To reveal that this is indeed a good idea, we also show some results when we generated at most one column per iteration which is determined by the overall shortest path (single pricing). Table 6 shows the figures

	easy			medium			hard		
no.	*LB*	*CG*	*UB*	*LB*	*CG*	*UB*	*LB*	*CG*	*UB*
1	43	43*	43	43	43*	43	43	43*	43
2	35	35*	35	35	35*	35	35	35*	35
3	44	44*	44	44	44*	44	43	43.81	43.81
4	44	44*	44	44	44*	44	44	44*	44
5	45	45*	45	45	45*	45	45	45*	45
6	41	41*	41	40	40.9	41	39	39.74	40.82
7	47	47*	47	47	47*	47	47	47	47
8	44	44*	44	44	44*	44	44	44	44
9	40	40*	40	40	40*	40	40	40*	40
10	47	47*	47	47	47*	47	47	47*	47

Table 1: Computational Results — Very Small Instances

	easy			medium			hard		
no.	*LB*	*CG*	*UB*	*LB*	*CG*	*UB*	*LB*	*CG*	*UB*
1	302	302	304.70	297	300.20	303.60	264	291.36	296.94
2	299	299	304.71	292	297.97	303.67	274	288.78	297.91
3	542	543	543	510	541.62	541.95	433	511.64	524.85
4	180	180	184	173	179.60	184	162	170.63	176.75
5	298	298	299.75	295	298	299.75	267	293.59	296.15
6	434	434	434	420	432.09	433.03	386	418.85	424.34
7	751	751	753	750	751	753	666	742.87	747.16
8	317	317	319.97	309	317	319.97	278	307.50	316.45
9	380	380	380	380	380	380	357	378.05	378.93
10	414	414*	414	413	414	414	363	403.29	409.07

Table 2: Computational Results — Small Instances

	easy		medium		hard	
no.	*CG*	*LP*	*CG*	*LP*	*CG*	*LP*
1	0.28	0.00	0.28	0.00	0.51	0.00
2	0.21	0.00	0.20	0.00	0.37	0.00
3	0.20	0.00	0.17	0.00	0.21	0.00
4	0.28	0.00	0.24	0.00	0.47	0.00
5	0.26	0.00	0.28	0.00	0.39	0.00
6	0.36	0.00	0.50	0.00	0.80	0.00
7	0.45	0.00	0.51	0.00	0.69	0.00
8	0.38	0.00	0.44	0.00	0.57	0.00
9	0.52	0.00	0.57	0.00	1.02	0.00
10	0.52	0.00	0.44	0.00	0.83	0.00

Table 3: CPU–Times — Very Small Instances

no.	easy		medium		hard	
	CG	*LP*	*CG*	*LP*	*CG*	*LP*
1	3.42	2.92	5.30	4.51	10.81	11.35
2	3.12	1.99	4.66	2.32	7.91	8.95
3	9.69	6.85	15.88	15.24	33.81	80.85
4	1.61	0.92	3.05	1.34	3.67	2.49
5	2.72	2.22	5.54	3.54	10.88	9.56
6	4.33	3.63	5.45	8.45	11.38	23.28
7	6.22	10.56	10.81	17.01	62.92	146.95
8	2.49	2.71	5.53	6.33	15.79	18.41
9	4.24	2.81	3.26	3.78	10.75	9.25
10	3.36	3.51	6.94	4.35	19.96	20.30

Table 4: CPU–Times — Small Instances

no.	easy	medium	hard
1	123	143	200
2	121	130	198
3	224	279	374
4	80	100	115
5	123	141	215
6	168	195	261
7	285	328	484
8	120	138	217
9	169	165	240
10	165	202	274

Table 5: Size of the Last Master Problem — Small Instances

no.	easy	medium	hard
1	126	142	186
2	126	139	180
3	214	252	333
4	81	96	107
5	127	141	202
6	173	191	253
7	296	320	428
8	127	139	203
9	169	175	217
10	168	187	262

Table 6: Size of the Last Master Problem — Small Instances — One Column Per Iteration

for the size of the last master problem. Comparing this with Table 5 in general less columns have to be generated. Unfortunately, the run–time upon termination is much shorter when more than just one column is generated per iteration. This is due to the fact that far more LPs have to be solved in the case of single pricing. Hence, multiple pricing is advantageous.

7 Summary and Conclusions

When scheduling its audit–staff, the management of an auditing firm encompasses a number of decisions. These may be grouped into several categories which differ markedly in terms of organizational echelon involved, length of the planning horizon and the planning periods, degree of aggregation of the audit tasks, degree of detail of the required information, and decision objective. However, traditional audit–staff scheduling models are single–level models which try to construct a direct assignment of auditors to tasks and periods. To facilitate algorithmic treatment, all these models are more or less gross simplifications of practical planning situations.

In this paper, we introduce an audit–staff scheduling model which comprises many features being important with respect to audit management in practice. For dealing with this model, a set partitioning/column generation approach is developed. The LP–relaxation of the set partitioning problem is solved by column generation in order to compute tight upper bounds. Frequently, the solution of the continuous relaxation is integral and, hence, an optimal solution is obtained.

Acknowledgement

This work was done with partial support from the Deutsche Forschungsgemeinschaft. Furthermore, we are indepted to Knut Haase who motivated us to start thinking about audit–staff scheduling and column generation.

References

[1] BALACHANDRAN, B.V. AND A.A. ZOLTNERS (1981), "An Interactive Audit–Staff Scheduling Decision Support System", *The Accounting Review*, Vol. 56, pp. 801–812.

[2] BARTUSCH, M., R.H. MÖHRING, AND F.J. RADERMACHER (1988), "Scheduling Project Networks with Resource Constraints and Time Windows", *Annals of Operations Research*, Vol. 16, pp. 201-240.

[3] BIXBY, N. AND E. BOYD (1996),*Using the CPLEX Callable Library*, CPLEX Optimization Inc., 7710–T Cherry Park, Houston.

[4] BRADLEY, S.P., A.C. HAX, AND T.L. MAGNANTI (1977), *Applied Mathematical Programming*, Addison–Wesley, Reading.

[5] CHAN, K.H. AND B. DODIN (1986), "A Decision Support System for Audit–Staff Scheduling with Precedence Constraints and Due Dates", *The Accounting Review*, Vol. 61, pp. 726–733.

[6] DODIN, B. AND K.H. CHAN (1991), "Application of Production Scheduling Methods to External and Internal Audit Scheduling", *European Journal of Operational Research*, Vol. 52, pp. 267–279.

[7] DODIN, B. AND A.A. ELIMAM (1997), "Audit Scheduling with Overlapping Activities and Sequence Dependent Setup Costs", *European Journal of Operational Research*, Vol. 97, pp. 22–33.

[8] DODIN, B., A.A. ELIMAM AND E. ROLLAND (1996), "Tabu Search in Audit Scheduling", Working Paper, University of California/Riverside.

[9] DREXL, A. (1991), "Scheduling of Project Networks by Job Assignment", *Management Science*, Vol. 37, pp. 1590–1602.

[10] FOURER, R., D.M. GAY AND B.W. KERNIGHAN (1993), *AMPL — A Modelling Language for Mathematical Programming*, The Scientific Press, San Francisco.

[11] GARDNER, J.C., R.J. HUEFNER, AND V. LOFTI (1990), "A Multiperiod Audit Staff Planning Model Using Multiple Objectives: Development and Evaluation", *Decision Sciences*, Vol. 21, pp. 154–170.

[12] KOLISCH, R., A. SPRECHER, AND A. DREXL (1995), "Characterization and Generation of a General Class of Resource–Constrained Project Scheduling Problems", *Management Science*, Vol. 41, pp. 1693–1703.

[13] PATTERSON, J.H. (1984), "A Comparison of Exact Approaches for Solving the Multiple Constrained Resource, Project Scheduling Problem", *Management Science*, Vol. 30, pp. 854–867.

[14] PRITSKER, A.A.A., W.D. WATTERS, AND P.M. WOLFE (1969), "Multiproject Scheduling with Limited Resources: A Zero–One Programming Approach", *Management Science*, Vol. 41, pp. 93–108.

[15] NEUMANN, K. AND J. ZHAN (1995), "Heuristics for the Minimum Project-Duration Problem with Minimal and Maximal Time Lags under Fixed Resource Constraints", *Journal of Intelligent Manufacturing*, Vol. 6, pp. 145–154.

[16] SALEWSKI, F. (1995), *Hierarchische Personaleinsatzplanung in Wirtschaftsprüfungsgesellschaften*, Physica, Heidelberg.

[17] SALEWSKI, F. (1996), *Tabu Search Algorithms for Project Scheduling Under Resource and Mode Identity Constraints,* IFORS 14th Triennial Meeting, Vancouver, July 9.

[18] SALEWSKI, F. AND A. DREXL (1993), "Personaleinsatzplanung in Wirtschaftsprüfungsgesellschaften — Bestandsaufnahme und konzeptioneller Ansatz", *Zeitschrift für Betriebswirtschaft* , Vol. 63, pp. 1357–1376.

[19] SALEWSKI, F., A. SCHIRMER, AND A. DREXL (1997), "Project Scheduling under Resource and Mode Identity Constraints: Model, Complexity, Methods, and Application", *European Journal of Operational Research*, Vol. 102, S. 88–110.

Appendix A: Project Scheduling–based Mathematical Programming Formulation

To simplify the mathematical formulation of the problem, we perform some preliminary computations. First, the indices e, p, s are replaced by

$$j = g(e,p,s) = \sum_{e'=1}^{e-1} \sum_{p'=1}^{P_{e'}} S_{e'p'} + \sum_{p'=1}^{p-1} S_{ep'} + s \tag{15}$$

Thus, the parameters V_{eps}, $q_{epsp's'm}$, $\hat{q}_{epsp's'm}$ and k_{epsma} become V_j, $q_{jj'm}$, $\hat{q}_{jj'm}$ and k_{jma}. Then, let denote $f_e = g(e,1,1)$ the first and $l_e = g(e,P_e,S_{eP_e})$ the last subphase of each engagement e. Further, the maximum lags can be transformed into minimum lags (with a corresponding update of V_j) (cf. Bartusch, Möhring, and Radermacher 1988), and from the time–lags, the release times, and the deadlines earliest and latest finish times EF_j and LF_j can be computed, respectively.

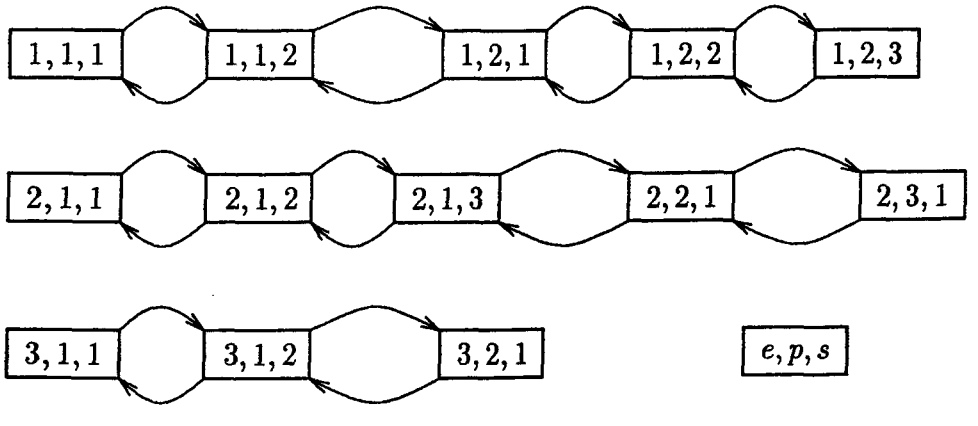

Figure 3: Chain Structure of Time Lags

Note that our problem setting covers not only the standard type of time–lags, namely minimum time–lags, but also the less common maximum time–lags. These can easily be converted into minimum time–lags using the transformation introduced in Bartusch, Möhring, and Radermacher (1988). It is quite obvious that the presence of maximum lags

tends to render scheduling problems even more difficult. In fact, to our knowledge there are only very few results so far on the performance of solution procedures for this kind of problems (cf. e.g. Neumann and Zhan 1995). The presence of minimum and maximum time–lags between subsequent subphases, along with their special sequence, which arises from the above mentioned decomposition process, imply for each engagement a chain structure of the time lags as illustrated in Figure 3, where each node has the format $\boxed{e,p,s}$. Each structure may be seen as being composed of arc–disjoint cycles of length 2, one between first and second subphase, one between second and third one, and so forth up to the last cycle between last but one and last subphase. Finally, recall from Section 2 that precedence relations exist only between subphases belonging to the same engagement, a characteristic denoted as 'isolating'.

Now, the assignment of subphases to modes and periods can be represented by binary variables $x_{jmt} = 1$, if subphase j is performed in mode m and completed in period t ($x_{jmt} = 0$, otherwise). This allows to formulate a binary program — using the general framework given in Pritsker, Watters, and Wolfe (1969) — as follows:

$$\max \sum_{e=1}^{E} \sum_{m=1}^{M_e} v^{em} \sum_{t=EF_{f_e}}^{LF_{f_e}} x_{f_e mt} \tag{16}$$

s.t.

$$\sum_{m=1}^{M_e} \sum_{t=EF_{f_e}}^{LF_{f_e}} x_{f_e mt} = 1 \qquad e = 1,\ldots,E \tag{17}$$

$$\sum_{t=EF_{f_e}}^{LF_{f_e}} x_{f_e mt} - \sum_{t=EF_j}^{LF_j} x_{jmt} = 0 \qquad \begin{array}{l} e = 1,\ldots,E \\ j = f_e+1,\ldots,l_e \\ m = 1,\ldots,M_e \end{array} \tag{18}$$

$$\sum_{m=1}^{M_e} \sum_{t=EF_{j'}}^{LF_{j'}} (t + q_{j'jm})\, x_{j'mt} - \sum_{m=1}^{M_e} \sum_{t=EF_j}^{LF_j} (t-1)\, x_{jmt} \leq 0 \qquad \begin{array}{l} e = 1,\ldots,E \\ j = f_e,\ldots,l_e \\ j' \in V_j \end{array} \tag{19}$$

$$\sum_{e=1}^{E} \sum_{m=1}^{M_e} \sum_{\substack{j=f_e \\ t\in\{EF_j,\ldots,LF_j\}}}^{l_e} k_{jma}\, x_{jmt} \leq C_{at} \qquad \begin{array}{l} a = 1,\ldots,A \\ t = 1,\ldots,T \end{array} \tag{20}$$

$$\sum_{m=1}^{M_e} \sum_{\substack{j=f_e \\ t\in\{EF_j,\ldots,LF_j\}}}^{l_e} k_{jma}\, x_{jmt} \leq D_{et} \qquad \begin{array}{l} e = 1,\ldots,E \\ a = 1,\ldots,A \\ t = 1,\ldots,T \end{array} \tag{21}$$

$$x_{jmt} \in \{0,1\} \qquad \begin{array}{l} e = 1,\ldots,E \\ j = f_e,\ldots,l_e \\ m = 1,\ldots,M_e \\ t = EF_j,\ldots,LF_j \end{array} \tag{22}$$

The objective function (16) maximizes the total team preference over all engagements. Due to (17) it suffices to include only one subphase of each engagement in (16). The choice of the first one is arbitrary. The subphase completion constraints (17) stipulate that the first subphase of each engagement is completed exactly once in one of its modes. The

mode identity constraints (18) guarantee for each engagement that if the first subphase is completed then the other subphases will be completed as well, and in the same mode as the first one. Thus, (17) and (18) combine to ensure that all subphases of each engagement will be processed in the same mode, and that all of them will be completed. The temporal constraints (19) represent the precedence order on the subphases and enforce respection of the time–lags between them. The auditor capacity constraints (20) assure that for no auditor his per–period workload exceeds his capacity. In this regard, each auditor is treated as a renewable resource. The engagement capacity constraints (21) guarantee that for no engagement and no auditor the maximum per–period processing time is exceeded. In this regard, each engagement is treated as a renewable resource.

Table 7 summarizes the instances and, in addition, states the problem size in terms of the number of binary variables of the model formulation (16) to (22).

size	# weeks	# auditors	# engagements	#variables
very small	13	6	10	10,400
small	13	30	95	98,800
medium–size	26	55	280	728,000
large	52	125	880	5,948,800

Table 7: Instance Characteristics

Appendix B: Illustrative Example

Throughout this paper we make use of the instance provided in Tables 8, 9, 10, 11, 12, and 13, respectively. Note, the index i serves to interrelate the precendence relations with respect to Tables 11 and 12.

$E = 4, T = 13, A = 6$
$M_1 = 5, M_2 = 3, M_3 = 8, M_4 = 2$
$P_1 = 2, P_2 = P_3 = P_4 = 1$
$S_{11} = 1, S_{12} = 5, S_{21} = 3, S_{41} = S_{51} = 1$
$C_{at} = 96, a = 1, \ldots, 6, t = 1, \ldots, 13$
$D_{et} = 40, e = 1, \ldots, 4, t = 1, \ldots, 13, D_{25} = 0$

Table 8: Instance Data — Basic Parameters

e/m	1	2	3	4	5	6	7	8
1	4	5	9	2	1			
2	10	4	2					
3	7	6	6	5	2	5	2	1
4	5	7						

Table 9: Instance Data — Preference Values v^{em}

e/p	1	2	
1	1	7	λ_{ep}
2	3		
3	6		
4	5		
1	2	12	δ_{ep}
2	7		
3	8		
4	6		

Table 10: Instance Data — Release Times λ_{ep} and Deadlines δ_{ep}

	$V_{111} = \emptyset$	$V_{211} = \emptyset$	$V_{311} = \emptyset$	$V_{411} = \emptyset$	
	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$
$e = 1$	$V_{121} = \{(1,1,1)\}$	$V_{122} = \{(1,2,1)\}$	$V_{123} = \{(1,2,2)\}$	$V_{124} = \{(1,2,3)\}$	$V_{125} = \{(1,2,4)\}$
	$i = 6$	$i = 7$			
$e = 2$	$V_{212} = \{(2,1,1)\}$	$V_{213} = \{(2,1,2)\}$			

Table 11: Instance Data — Predecessors V_{eps}

i/m	1	2	3	4	5	e
1	5/13	0/13	0/13	1/13	1/13	1
2	0/0	-1/13	-1/13	0/0	0/0	
3	-1/13	-1/13	-1/13	0/0	0/0	
4	-1/13	-1/13	-1/13	0/0	0/0	
5	-1/13	-1/13	-1/13	0/1	0/0	
6	0/1	0/2	0/0			2
7	0/0	0/0	0/0			

Table 12: Instance Data — Minimal/Maximal Time Lags $q_{epsp's'm}/\hat{q}_{epsp's'm}$

a	1	2	3	4	5	6		
k_{1111a}							$e=1$	$m=1$
⋮	⋮	⋮	⋮	⋮	⋮	⋮		
k_{1251a}								
k_{1112a}	36	16	8	16	8			$m=2$
k_{1212a}		8	40	40	40			
k_{1222a}								
⋮	⋮	⋮	⋮	⋮	⋮	⋮		
k_{1252a}								
k_{1113a}		32	28	16	40	40		$m=3$
k_{1213a}		40	8	16	40	4		
k_{1223a}								
⋮	⋮	⋮	⋮	⋮	⋮	⋮		
k_{1253a}								
k_{1114a}	40							$m=4$
k_{1214a}								
k_{1224a}	16							
k_{1234a}	40							
k_{1244a}	40							
k_{1254a}	8							
k_{1115a}	16							$m=5$
k_{1215a}	32							
k_{1225a}	24							
k_{1235a}	16							
k_{1245a}	16							
k_{1255a}	40							
k_{2111a}							$e=2$	$m=1$
⋮	⋮	⋮	⋮	⋮	⋮	⋮		
k_{2131a}								
k_{2112a}		40		8		24		$m=2$
k_{2122a}		8	16	40		16		
k_{2132a}		24	8	40		24		
k_{2113a}			40	8	40	12		$m=3$
k_{2123a}			40	8	40	40		
k_{2133a}			32	16	8	24		
k_{3111a}							$e=3$	$m=1$
k_{3112a}	24	40		8		8		$m=2$
k_{3113a}		40	24			24		$m=3$
k_{3114a}				8	40	40		$m=4$
k_{3115a}		16			32	24		$m=5$
k_{3116a}	40	40						$m=6$
k_{3117a}			16	8				$m=7$
k_{3118a}	40			40	20	40		$m=8$
k_{4111a}							$e=4$	$m=1$
k_{4112a}		40						$m=2$

Table 13: Instance Data — Capacity Usages k_{epsma} (Missing Entries Are Zero)