Salewski, Frank; Bartsch, Thomas

**Working Paper**

A comparison of genetic and greedy randomized algorithms for medium-to-short-term audit-staff scheduling

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 356

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

Suggested Citation: Salewski, Frank; Bartsch, Thomas (1994) : A comparison of genetic and greedy randomized algorithms for medium-to-short-term audit-staff scheduling, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 356, Universität Kiel, Institut für Betriebswirtschaftslehre, Kiel

This Version is available at:
http://hdl.handle.net/10419/155427

**Standard-Nutzungsbedingungen:**
Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden. Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

**Terms of use:**
Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.
A Comparison of Genetic and Greedy Randomized Algorithms for Medium-to-Short-Term Audit-Staff Scheduling

Salewski / Bartsch

November 1994
Abstract: Based upon an empirical survey among the 200 biggest CPA firms in Germany an hierarchical modeling framework for audit-staff scheduling with three levels has been developed. For the second level, the so-called medium-to-short-term planning, a binary optimization model is introduced which is closely related to resource-constrained project scheduling. In an extensive computational study several genetic algorithms (GA) with and without domain-specific knowledge as well as a greedy randomized algorithm (RA) are investigated.

Besides introducing a generalization concerning the representation of the genes of a GA which is applicable to a wide range of recent procedures it is proved that the RA is a special case of a GA. Furthermore, we observed that a GA of a modest population size with domain-specific knowledge performed best if at least a certain number of individuals is generated.

Key Words: LOCAL SEARCH, GENETIC ALGORITHMS, GREEDY RANDOMIZED ALGORITHMS, REGRET-BASED BIASED RANDOM SAMPLING, PROJECT MANAGEMENT / SCHEDULING, AUDIT-STAFF SCHEDULING

I. Introduction

As the name suggests, genetic algorithms (GA) are motivated by the theory of evolution and date back to the early works of Rechenberg (1973), Holland (1975), and Schwefel (1977), see also Goldberg (1989), Liepins and Hillard (1989), and Michalewicz (1992). Compared to standard heuristics "GA's are not well suited for fine-tuning structures which are very close to optimal solutions" (Grefenstette 1987). Therefore it is essential, if a competitive GA is desired, to incorporate (local search) improvement operators (cp. Johnson 1990). Ways how to take into account improvement operators or in other words domain-specific knowledge are described in Storer, Wu and Vaccari (1992), Bean (1994), Pesch (1994) and Dornendorf and Pesch (1995). On the other hand it seems to be of interest to compare the results obtained by GA's with those produced by greedy randomized algorithms (RA) (cp. Laguna, Feo, and Elrod 1994) which perform very well if applied to project scheduling problems (cp. e.g. Drexel 1991 and Kolisch 1995). By reflecting how to allow a comparison that is not based upon CPU-times we found out that under certain assumptions a RA can be interpreted as a special case of a GA. The problem we have chosen as an example is taken from the field of audit-staff scheduling and closely related to resource-constrained project scheduling.

The remainder of this paper is organized as follows. In Section 2 we formulate the medium-to-short-term audit-staff scheduling problem in terms of a binary optimization model. Section 3 is devoted to the explanation of knowledge-based (priority) rules for job selection and job scheduling, while in Section 4 the underlying algorithmic scheme and details of gene representation are discussed. Section 5 outlines the design of a problem specific instance generator and defines the statistical model, the experimental design, and the performance measures
used in an extensive computational study whose results are covered in Section 6. Finally, Section 7 provides a brief summary.

2. Problem Setting

Traditional audit-staff scheduling models (Balachandran and Zoltners 1981, Chan and Dodin 1986, Dodin and Chan 1991, Drexl 1991) are single-level models which construct a direct assignment of auditors to tasks and periods. Due to the absence of extensive empirical investigations all these models have been assumed to be more or less gross simplifications of practical planning situations. This assumption led us to conduct a survey among the 200 biggest CPA firms in Germany (Salewski and Drexl 1993). Its results have shown that the process of audit-staff scheduling requires a number of different decisions which differ markedly in terms of decision makers involved, length of the planning horizon and the planning periods, degree of aggregation of the audit engagements and tasks, problem to be solved, and decision objective. Based upon these findings we formulated an hierarchical modeling framework comprising three levels, viz. the medium-term, the medium-to-short-term, and the short-term audit-staff scheduling. These levels may be characterized as follows (for details cp. Salewski 1995):

- The medium-term planning assigns teams of auditors to the engagements. It constructs a schedule of all engagements, and determines the workload per auditor and week over a planning horizon of between three and twelve months (RA's capable of solving the problem are discussed in Salewski, Drexl, and Schirmer 1994 and Salewski, Schirmer, and Drexli 1994).

- The medium-to-short-term planning disaggregates the results of the medium-term 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 jobs of those engagements in which he is involved in the considered week (RA's capable of solving the problem are discussed in Bartsch and Salewski 1993, see also Salewski, Bartsch, and Pesch 1994).

- The short-term planning is based upon the results of the medium-to-short-term 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. This is done for a period length of one hour (RA's capable of solving the problem are discussed in Böttcher and Salewski 1993).

In the following we consider the medium-to-short-term planning. The notation used in the subsequent Sections is summarized in Table 1.

Defining variables

\[ x_{jt} = \begin{cases} 1, & \text{if job } j \text{ is finished in period } t \\ 0, & \text{otherwise} \end{cases} \]

allows to formulate a binary program as follows:
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition / Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Number of auditors</td>
</tr>
<tr>
<td>a</td>
<td>Specific auditor, a = 1,...,A</td>
</tr>
<tr>
<td>( C_{at} )</td>
<td>Capacity of auditor a in period t (renewable resource) where ( C_{at} \in {0, 1} ) indicates that auditor a is not available (is available) in period t</td>
</tr>
<tr>
<td>( c_t )</td>
<td>Costs of processing in period t</td>
</tr>
<tr>
<td>( D_{et} )</td>
<td>Maximum duration of processing on engagement e in period t (renewable resource) where ( D_{et} \in {0, A} ) indicates that engagement e cannot (can) be processed in period t</td>
</tr>
<tr>
<td>E</td>
<td>Number of engagements</td>
</tr>
<tr>
<td>e</td>
<td>Specific engagement, e = 1,...,E</td>
</tr>
<tr>
<td>EF_j</td>
<td>MPM earliest finish time of job j</td>
</tr>
<tr>
<td>( \mathcal{H}_t )</td>
<td>Set of jobs which possibly can be finished in period t with ( \mathcal{H}_t = { j</td>
</tr>
<tr>
<td>J</td>
<td>Number of jobs over all engagements, ( J = \mathcal{L}_1 + \mathcal{L}_2 + \ldots + \mathcal{L}_E )</td>
</tr>
<tr>
<td>j</td>
<td>Specific job, j = 1,...,J; each job is assumed to have a duration of exactly one period</td>
</tr>
<tr>
<td>( \mathcal{J}_a )</td>
<td>Set of jobs auditor a has to work on with ( \mathcal{J}_a = { j</td>
</tr>
<tr>
<td>k_{ja}</td>
<td>Usage of (renewable resource) auditor a by job j where ( k_{ja} \in {0, 1} ) indicates that a is not working (is working) on j</td>
</tr>
<tr>
<td>LF_j</td>
<td>MPM latest finish time of job j</td>
</tr>
<tr>
<td>( \mathcal{L}_e )</td>
<td>Set of jobs belonging to engagement e</td>
</tr>
<tr>
<td>see'</td>
<td>Travel / setup time required between the end of processing of (a job j from) engagement e and the start of work on (a job j' of) engagement e'</td>
</tr>
<tr>
<td>T</td>
<td>Number of periods, planning horizon (regularly one week)</td>
</tr>
<tr>
<td>t</td>
<td>Specific period (regularly covering a halfshift of four hours), t = 1,...,T</td>
</tr>
<tr>
<td>( \mathcal{V}_j )</td>
<td>Set of all immediate predecessors of job j</td>
</tr>
<tr>
<td>x_{jt}</td>
<td>Job j is finished in period t</td>
</tr>
<tr>
<td>Z(\cdot)</td>
<td>Objective function value</td>
</tr>
</tbody>
</table>
Minimize \[ Z(x) = \sum_{a=1}^{A} \sum_{j \in \mathcal{J}_a} \sum_{t=EF_j}^{LF_j} c_t x_{jt} \] (1)

s.t.

\[ \sum_{t=EF_j}^{LF_j} x_{jt} = 1 \quad (1 \leq j \leq J) \] (2)

\[ \sum_{t=EF_{j'}}^{LF_{j'}} t x_{j't} \leq \sum_{t=EF_j}^{LF_j} (t-1) x_{jt} \quad (1 \leq j' \leq J, 1 \leq j \leq J, j' \in \mathcal{J}_e') \] (3)

\[ \sum_{j \in \mathcal{J}_e} \sum_{a=1}^{A} k_{ja} x_{jt} \leq D_{et} \quad (1 \leq e \leq E, 1 \leq t \leq T) \] (4)

\[ \sum_{j \in \mathcal{J}_e} \sum_{j' \in \mathcal{J}_a} k_{ja} x_{jt} \leq C_{at} \quad (1 \leq a \leq A, 1 \leq t \leq T) \] (5)

\[ x_{jt} + \sum_{t'=t}^{t+g_{ee'}} x_{j't'} \leq 1 \quad (1 \leq a \leq A, 1 \leq e \leq E, 1 \leq e' \leq E, e \neq e'); \]

\[ j \in \mathcal{J}_e; j' \in \mathcal{J}_e'; j, j' \in \mathcal{J}_a; EF_j \leq t \leq LF_j \] (6)

\[ x_{jt} \in \{0, 1\} \quad (1 \leq j \leq J, EF_j \leq t \leq LF_j) \] (7)

In (1), \( x \) represents the vector of all binary variables; \( Z(x) \) is the objective function value for any feasible vector \( x \) with respect to (2) - (7). The job completion constraints (2) stipulate that each job is completed exactly once. The precedence constraints (3) enforce respect of the predecessor-successor-relations between the jobs where we have unit processing times. The capacity constraints (4) and (5) represent per-period capacity restrictions of the renewable resources engagements and auditors, respectively. By virtue of the travel / setup constraints (6) it is guaranteed that there remains enough travel time for an auditor who is working on jobs \( j \) and \( j' \) that are belonging to two different engagements \( e \) and \( e' \).
3. Knowledge-Based Rules for Job Selection and Job Scheduling

Obviously the aforementioned problem is NP-complete and NP-hard, respectively. (In Salewski, Schirmer, and Drexel 1994 it is proved that any scheduling feasibility (optimization) problem comprising at least two periods and one renewable resource with arbitrary usage is NP-complete (NP-hard).) Therefore, the only line of attack for tackling practical problems comprising up to 200,000 variables (cp. Section 5.1.) is provided by approximation algorithms.

Deterministic approximation algorithms yield only one solution for an instance, even if applied several times. Considering that this solution may be arbitrarily bad or even infeasible, determinism seems to be a major deficiency for approximation algorithms. Randomized methods which are also known as semi-greedy (cp. Hart and Shogan 1987) or regret-based biased random sampling algorithms (cp. Kolisch 1995) try to overcome the shortcoming of determinism by performing the selection process randomly, but according to probabilities which are proportional to priority values. In this way, in each scheduling step every job may be chosen, though those sharing higher priorities will have a higher probability of being selected. Due to their nondeterminism, repeated application of randomized methods will produce a set of solutions rather than one sole solution. Usually some of these solutions will be better than the one found with the deterministic version of the same method (cf. Cooper 1976). Moreover, no tie-breaking rules need to be specified for randomized methods: since they operate randomly ties cannot occur. However, even if the implemented methods are borrowing from some of the ideas incorporated in the randomized scheduling method STOCOM (Drexel and Grunewald 1993), common approximation algorithms cannot be utilized directly since these would not take the nonregular objective function (1) sufficiently into account.

Generally, common approximation algorithms for project scheduling are distinguished to be serial or parallel (cp. Kelley 1963). While the former regularly schedule a selected job as early as possible, the latter proceed chronologically over all periods of the planning horizon trying to schedule in each period as many jobs as possible. Unfortunately, both procedures yield poor results with respect to the problem under consideration. E.g., imagine a working day with four periods from 8 a.m. to 12 a.m., 12 a.m. to 4 p.m., 4 p.m. to 8 p.m., and 8 p.m. to 12 p.m. with a regular working time from 8 a.m. to 4 p.m. Then processing a job in one of the latter two periods may cause some extra costs for overtime working. If there remains a sufficient number of cost-less regular periods on other days, it is not desirable to schedule jobs in one of these more expensive overtime periods. The solution to this problem we have chosen is to select primarily a job according to a priority rule \( \omega \) (job selection), and then to decide according to another priority rule \( \pi \) if to schedule this job in a prechosen period or not (job scheduling).

Assuming priorities values \( \bar{\omega}_j \) for all jobs in the set \( EJ \) of currently eligible jobs (\( \forall j \in EJ \)) to be calculated according to a rule \( \omega \) for the job selection, modified priorities can be derived from them as
\[ \bar{w}_j = (\bar{w}_j + \delta)^\alpha \]  
(\forall j \in EJ) \quad (8)

where the control parameters \( \delta \in \mathbb{R}_{>0} \) and \( \alpha \in \mathbb{R}_{\geq 0}. \) \( \delta > 0 \) guarantees \( \bar{w}_j \) to be nonzero and thus gives also a job with a priority value of zero a chance of being chosen; the control parameter \( \alpha \) allows to diminish or enforce the differences between the rule-dependent values for \( \alpha < 1 \) or \( \alpha > 1, \) respectively. The probability \( \bar{w}_j \) (\( \forall j \in EJ \)) that job \( j \) will be selected is

\[ \bar{w}_j = \frac{\bar{w}_j}{\left( \sum_{j' \in EJ} \bar{w}_{j'} \right)} \]  
(\forall j \in EJ) \quad (9)

Obviously, the sum of all \( \bar{w}_j (\forall j \in EJ) \) is equivalent to 1. Thus, a random number \( z_1 \) drawn from the continuous interval \( [0, 1] \) identifies unambiguously one specific job.

In pretests documented in Bartsch and Salewski (1993) we have considered the rules Minimum Latest Finish Time, Minimum Slack, and Maximum Resource Usage (MAX RU) for the job selection. Since the rule MAX RU performed slightly better than the others and in order to restrict the computations which are anyway extremely extensive we investigate in this study only MAX RU. Its priority value can be calculated from

\[ \bar{w}_j = \frac{A}{\sum_{a=1}^A k_{ja} - \min \left \{ \sum_{a=1}^A k_{ja} \mid j \in EJ \right \}} \]  
(\forall j \in EJ) \quad (10)

Analogously, the job scheduling probabilities \( \pi_r \) (\( r \in \{0, 1\} \)), where \( r = 0 \) \((1)\) indicates the case of not scheduling (scheduling) the job selected, can be obtained from priority rules \( \pi \) using control parameters \( \beta \in \{0, 1\} \) and \( \varepsilon \in \mathbb{R}_{>0} \) as well as random numbers \( z_2 \in [0, 1] \). Note that choosing \( \beta = 0 \) yields probabilities of 0.5 for each of the events "scheduling" and "not scheduling". In other words, a rule has no effect in this case, while by setting \( \beta = 1 \) it has.

The rules for the job scheduling are based on three criteria, namely the Job Status (JST) where \( JST_j \in [0, 1] \), the Auditor Status (AST) where \( AST_j \in [0, 1] \), and the Setup Status (SST) where \( SST_j \in [0, 1] \), which have been identified to be of practical relevance. A formal definition of the criteria is given in the Appendix. Each of the criteria yields a value of 1 for a selected job \( j \) if the prechosen period \( t \) has costs \( c_t \) which are minimal within the planning horizon. Additionally, high values are received

- for \( JST_j \) if "many" (MPM-) successors of \( j \) already must be scheduled with costs higher than \( c_t \),

- for \( AST_j \) if "many" remaining jobs of the auditors involved in processing \( j \) already must be scheduled with costs higher than \( c_t \), and

- for \( SST_j \) if setup / travel times to "many" jobs still unscheduled arise.

Now, several rules can be obtained straightforward by combining the criteria arbitrarily. We restrict ourselves again on basis of pretests documented in Bartsch and Salewski (1993) to one rule which performed significantly better than any other. As the only one this rule takes the three criteria simultaneously into account and thus is named Job Auditor Setup Status
(JASST). The priority value $\pi_1$ for scheduling a selected job $j$ in a prechosen period $t$ is given by

$$\pi_1 = 1 - (1 - JST_j)(1 - AST_j)(1 - SST_j)$$  \hspace{1cm} (11)

while

$$\pi_0 = 1 - \pi_1$$  \hspace{1cm} (12)

Accordingly, a job $j$ is also scheduled with a high probability if only one criterion has a high value.

4. Algorithmic Scheme

The problem decomposes into a number of subproblems each of which is determined by a set of jobs that can be performed by a group of auditors independently from the jobs of each other subproblem (for details cp. Salewski 1995). The composition of the optimal (suboptimal) solutions of each subproblem provides an optimal (suboptimal) solution to the overall problem. Hence, furtheron let us consider any of these subproblems. (For ease of notation no indices for denomenating the subproblems are used.)

Before discussing an appropriate representation of genes the underlying genetic scheme and the way how to decode individuals are explained.

Defining the number of individuals $t \in \mathbb{N}^+$ and the genetic parameters population size $\lambda \in \mathbb{N}^+$, crossover rate $v \in [0, 1]$ (if $v = 0$, no crossover takes places and thus no children exist; if $v = 1$, $\lambda$ children are produced), and mutation rate $\mu \in [0, 1]$ (if $\mu = 0$, no gene is mutated; if $\mu = 1$, each gene of each child is mutated) allows to describe the scheme of the GA as follows:

random initialization of the first generation;
for generation = 1 to $t / \lambda$ do
{  
crossover with rate $v$;
random mutation of children's genes with rate $\mu$;
decoding of individuals;
reproduction where the individual with the best objective function value survives in any case and the others are selected with a probability proportional to their objective function value;
}

Note that for guaranteeing the survival of the individual with the best objective function it is necessary to allow a mutation only on the children produced by the crossover. If the population size $\lambda = 1$ and the crossover rate $v = 1$ then one child is produced in each generation. Furthermore, a mutation rate of $\mu = 1$ means that for this child a completely new initialization of
each of its genes takes place. However, this is exactly what is occurring when we are applying a RA without embedding it into a genetic scheme. This implies that under these assumptions a RA can be interpreted as a special case of a GA. As a consequence the results obtained by GA's are comparable with those of a RA, e.g. on the basis of the number of individuals produced. (While common approaches compare methods on the basis of the CPU-time needed they may be unprecise if a multi-task operating system is used and the workload varies during the computation.)

For the purpose of decoding each individual is interpreted primarily as consisting of a number of genes each of which is representing one rule for job selection and one for job scheduling. (At the end of this section a gene will be defined more precisely.) Applying these rules to a problem we receive a solution from each individual by the following parallel scheme for decoding:

\[ EJ = \text{Set of Eligible Jobs} \]
\[ FJ = \text{Set of Finished Jobs} \]

\[ g = 1; \quad /\!* \text{Number of the gene decoded currently from the considered individual} */\!
\]
for \( t = 1 \) to \( T \) do
\{
    compute \( EJ \);
    while \( EJ \neq \emptyset \) do
        \{
            select \( j \in EJ \) according to \( \hat{d}_j \) from \( \text{geneg} \);
            \quad /\!* \text{Job selection} */\!
            decide if to schedule \( j \) according to \( \hat{\pi}_1 \) from \( \text{geneg} \);
            \quad /\!* \text{Job scheduling} */\!
            \( EJ = EJ \setminus j \);
            if (\( j \) is to schedule) then
                \{
                    \( x_{jt} = 1 \);
                    \( FJ = FJ \cup \{ j \} \);
                    update \( EJ \);
                \}
            \}
            \( g = g + 1 \);
        \}
\}

Because of the job scheduling decision a specific job may possibly be regarded more than once. Thus, the number of genes of an individual needed for decoding is not fixed in advance. To overcome this problem the sum of jobs contained in each set \( \mathcal{S}_t \) multiplied by 0.25 was found after pretests to be an appropriate number of genes for an individual (instead of the upper bound \( J \cdot T \)).
Finally, an appropriate representation of genes has to be developed. While in Goldberg (1989) a gene corresponds to a job, Dorndorf and Pesch (1995) tried to overcome the deficiencies of this approach when being applied to problems with precedence constraints by defining a gene to be a priority rule. Two disadvantages of this idea may be seen in the absence of a possibility to incorporate further improvement operators as well as in the effort required to implement a larger number of different rules which are needed for an efficient use. Based upon the idea of Dorndorf and Pesch (1995) as well as building upon the random key concept of Bean (1994) we primarily define a gene as a tuple \((\omega, \alpha, z_1)\) consisting of a rule \(\omega\), a control parameter \(\alpha\), and a random number \(z_1\) as above. Obviously, a constant choice of \(\alpha\) and \(z_1\) is equivalent to the representation of Dorndorf and Pesch (1995). On the other hand, a further special case which is very simple to implement occurs if \(\omega\) and \(\alpha\) are fixed. (This equals roughly the approach of Bean (1994).) Lending from these observations we used in our algorithms tuples \((\omega, \alpha, \delta, z_1, \pi, \beta, \varepsilon, z_2)\) where only the random numbers \(z_1\) and \(z_2\) must (but \(\omega, \alpha, \delta, \pi, \beta, \) and \(\varepsilon\) also could easily) be implemented because the rules as well as the control parameters are fixed within a run (defined as the production of \(i\) individuals) of the experiment described in the next section.

5. Experimental Performance Analysis

Before presenting the performance measures the generation of test instances and a statistical model as well as the experimental design are introduced.

5.1. Generation of Test Instances

To ensure a systematic and consistent generation of test instances, for each of the parameters of the problem described in Section 2 a domain and a discrete distribution function on the domain were defined from which parameter instantiations were generated randomly. In order to construct instances reflecting closely the practical problem setting of audit-staff scheduling, these definitions take the results of the above-mentioned survey among German CPA firms into account as far as possible.

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 attempted.

Although the size of an instance is determined by different factors, statistical analysis of the results of the survey revealed that all these parameters depend on the length of the planning horizon. In the sequel, three types of instances will be distinguished with respect to their size: small (20 periods with up to 30 auditors, 300 jobs, and 9,000 binary variables), medium-size (30 periods with up to 55 auditors, 550 jobs, and 16,500 binary variables), and large instances (60 periods with up to 125 auditors, 2,500 jobs, and 200,000 binary variables). In addition, very small instances (12 periods with up to 6 auditors, 36 jobs, and 436 binary variables) were
generated. While these instances are too small to bear practical relevance, they can be solved to optimality with standard codes as LINDO and OSL and thus serve as benchmarks.

The tractability of an instance is intended to reflect how easy or how difficult that particular instance is to solve. For the purposes of this study, the number of jobs some auditors have to work on simultaneously are assumed to be the only factor influencing the tractability of an instance: the lower the number of jobs some auditors have to work on simultaneously, the easier the corresponding instance is ceteris paribus to solve since its solution space becomes larger. The test problems will be provided by the authors upon request.

5.2. Statistical Model

For the purpose of this study, the execution of the GA is regarded as a random experiment, the outcome of which is determined by the following factors:

- $\omega$ priority rule employed for job selection
- $\pi$ priority rule employed for job scheduling
- $\alpha$ first control parameter for job selection
- $\beta$ first control parameter for job scheduling
- $\delta$ second control parameter for job selection
- $\epsilon$ second control parameter for job scheduling
- $\lambda$ population size
- $\mu$ mutation rate
- $\nu$ crossover rate
- $\sigma$ size of instances
- $\tau$ tractability of instances
- $\iota$ number of individuals

Specifying a set of values for each factor describes over which levels it is varied during an experiment, while one value for each factor determines one run of an experiment.

**Definition 1:** An experiment is a tuple $(\Omega, \Pi, \Delta, \mathbf{A}, \mathbf{E}, \mathbf{M}, \mathbf{N}, \Sigma, \mathbf{T}, \mathbf{H})$, where

- $\Omega$ is a set of priority rules for job selection
- $\Pi$ is a set of priority rules for job scheduling
- $\Delta \subseteq \mathbb{R}_{\geq 0}$ is a set of values for the first job selection control parameter
- $\mathbf{B} \subseteq \mathbb{R}_{\geq 0}$ is a set of values for the first job scheduling control parameter
- $\mathbf{A} \subseteq \mathbb{R}_{> 0}$ is a set of values for the second job selection control parameter
- $E \subseteq \mathbb{R}_{>0}$ is a set of values for the second job scheduling control parameter
- $\Delta \subseteq \mathbb{N}^+$ is a set of population sizes
- $M \subseteq [0, 1]$ is a continuous interval of mutation rates
- $N \subseteq [0, 1]$ is a continuous interval of crossover rates
- $\Sigma \subseteq \{\text{very small, small, medium-size, large}\}$ is a set of sizes
- $T \subseteq \{\text{easy, medium, hard}\}$ is a set of tractabilities
- $H \subseteq \mathbb{N}^+$ is a set of numbers of individuals

**Definition 2**: A run of an experiment $(\Omega, \Pi, A, B, \Delta, E, \Delta, M, N, \Sigma, T, H)$ is a tuple $(\omega, \pi, \alpha, \beta, \delta, \varepsilon, \lambda, \mu, \upsilon, \sigma, \tau, i) \in \Omega \times \Pi \times A \times B \times \Delta \times E \times \Delta \times M \times N \times \Sigma \times T \times H$

The outcome of a run is - for each instance attempted - summarized in terms of two result variables. One, $BF_{\omega \pi \alpha \beta \delta \varepsilon \lambda \mu \upsilon \sigma \tau \theta}$ denotes the objective function value of the best solution found after $\theta$ individuals have been generated in that run. Two, $CPU_{\omega \pi \alpha \beta \delta \varepsilon \lambda \mu \upsilon \sigma \tau \theta}$ denotes the average CPU-time for producing and decoding an individual in that run. These variables are regarded as random variables which are assumed to be functions of the factors mentioned above.

### 5.3 Experimental Design

Due to the computational effort required to attempt a sample of all sizes, the scope of the experiment was limited to include only very small and small instances. Though no obstacle for using the developed method even on larger instances, this effort prevents the undertaking of a full factorial design experiment covering all instance classes. However, it is a widely accepted conjecture that algorithms performing well on smaller instances are also the best-performing ones for the larger ones (Davis and Patterson 1975, Badiru 1988, Alvarez-Valdés and Tamarit 1989). Of each instance class $(\sigma, \tau)$, ten instances were considered in the experiment. After pre-tests not further documented they were tackled by the algorithm $(\omega = \text{MAX RU}, \pi = \text{JASST})$ in connection with the 41 most promising combinations of control parameter values $(\alpha, \beta, \delta, \varepsilon)$ and genetic parameter values $(\lambda, \mu, \upsilon)$ described in Table 2 generating $i = 1,000,000$ individuals each.

Hence, $\Omega = \{ \text{MAX RU} \}, \Pi = \{ \text{JASST} \}, A = \{0, 2, 4, 8\}, B = \{0, 1\}, \Delta = \{1\}, E = \{0.05, 0.1, 0.2\}, M = \{0, 0.05, 0.1, 0.15, 1\}, N = \{0.5, 1.0\}, \Sigma = \{\text{very small, small}\}, T = \{\text{easy, medium, hard}\}$ as well as $H = \{1,000,000\}$ and the runs restricted as shown in Table 2. From this it follows that a specific combination of algorithm and parameters can be identified by $\alpha, \beta, \varepsilon, \lambda, \mu,$ and $\upsilon$. 
Table 2

*Algorithm, Genetic and Control Parameters*

<table>
<thead>
<tr>
<th></th>
<th>α</th>
<th>β</th>
<th>δ</th>
<th>ε</th>
<th>λ</th>
<th>μ</th>
<th>ν</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA_01_α</td>
<td>0, 2, 4, 8</td>
<td>1</td>
<td>1</td>
<td>0.05</td>
<td>1</td>
<td>1.00</td>
<td>1.0</td>
</tr>
<tr>
<td>GA_01_α</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.05</td>
<td>50</td>
<td>0.05</td>
<td>0.5</td>
</tr>
<tr>
<td>GA_02_α</td>
<td>0, 2, 4, 8</td>
<td>1</td>
<td>1</td>
<td>0.05</td>
<td>50</td>
<td>0.00</td>
<td>0.5</td>
</tr>
<tr>
<td>GA_03_α</td>
<td>0, 2, 4, 8</td>
<td>1</td>
<td>1</td>
<td>0.05</td>
<td>50</td>
<td>0.05</td>
<td>0.5</td>
</tr>
<tr>
<td>GA_04_α</td>
<td>0, 2, 4, 8</td>
<td>1</td>
<td>1</td>
<td>0.10</td>
<td>50</td>
<td>0.05</td>
<td>0.5</td>
</tr>
<tr>
<td>GA_05_α</td>
<td>0, 2, 4, 8</td>
<td>1</td>
<td>1</td>
<td>0.20</td>
<td>50</td>
<td>0.05</td>
<td>0.5</td>
</tr>
<tr>
<td>GA_06_α</td>
<td>0, 2, 4, 8</td>
<td>1</td>
<td>1</td>
<td>0.05</td>
<td>50</td>
<td>0.10</td>
<td>0.5</td>
</tr>
<tr>
<td>GA_07_α</td>
<td>0, 2, 4, 8</td>
<td>1</td>
<td>1</td>
<td>0.05</td>
<td>50</td>
<td>0.15</td>
<td>0.5</td>
</tr>
<tr>
<td>GA_08_α</td>
<td>0, 2, 4, 8</td>
<td>1</td>
<td>1</td>
<td>0.05</td>
<td>100</td>
<td>0.05</td>
<td>0.5</td>
</tr>
<tr>
<td>GA_09_α</td>
<td>0, 2, 4, 8</td>
<td>1</td>
<td>1</td>
<td>0.05</td>
<td>250</td>
<td>0.05</td>
<td>0.5</td>
</tr>
<tr>
<td>GA_10_α</td>
<td>0, 2, 4, 8</td>
<td>1</td>
<td>1</td>
<td>0.05</td>
<td>500</td>
<td>0.05</td>
<td>0.5</td>
</tr>
</tbody>
</table>

5.4 Performance Measures

Based upon the result variables introduced above, we define two performance measures allowing to summarize the outcome of an experiment in a convenient way.

The efficiency ratio \( \phi(\omega, \pi) \) of the algorithm \((\omega, \pi)\) using the control parameter values \((\alpha, \beta, \delta, \varepsilon)\) and the genetic parameter values \((\lambda, \mu, \nu)\) for an instance class \((\sigma, \tau)\) after the individual \(\theta\) in a run producing in total \(\tau\) individuals is computed - as the average over all instances of that class attempted - as the quotient of the objective function value of the best known solution for each instance attempted and \(BF_{\phi(\omega, \pi, \alpha, \beta, \delta, \varepsilon, \lambda, \mu, \nu)}\). (This measure is based upon the rule efficiency ratio proposed by Badiru (1988)) Clearly, \(\phi(\omega, \pi, \alpha, \beta, \delta, \varepsilon, \lambda, \mu, \nu, \sigma, \tau) \in [0,1]\); a ratio of one indicates that during \(\theta\) individuals the best value currently known has been found, while a ratio of zero reflects the fact that no feasible solution was generated at all.

The average running time \(\delta_{\phi(\omega, \pi, \alpha, \beta, \delta, \varepsilon, \lambda, \mu, \nu, \sigma, \tau)}\) is defined equal to \(\text{CPU}_{\phi(\omega, \pi, \alpha, \beta, \delta, \varepsilon, \lambda, \mu, \nu)}\).

To evaluate the effect of varying the above factors, different aggregate measures were derived from the above definitions. These aggregations served to isolate the effects of certain factors.

We refrain, however, from the tedious task of citing the respective definitions since they use simple averaging over all factors except of the size of the instances and the factor to be tested. Clearly, the average values can be interpreted as approximations of the expected values of the performance measures in general.
6. Computational Results

The results of our experiment with respect to the efficiency and the running time are summarized in Tables 3, 4 and 5.

Primarily, the efficiency ratios of the parameter combinations and their average results as well as their ranks are listed in Table 3. (The best results within each of the following tables are shaded.)

Table 3

<table>
<thead>
<tr>
<th></th>
<th>α = 0</th>
<th></th>
<th>α = 2</th>
<th></th>
<th>α = 4</th>
<th></th>
<th>α = 8</th>
<th></th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA_01_α</td>
<td>0.948</td>
<td>21</td>
<td>0.926</td>
<td>28</td>
<td>0.932</td>
<td>27</td>
<td>0.901</td>
<td>33</td>
<td>0.927</td>
</tr>
<tr>
<td>GA_01_α</td>
<td>0.473</td>
<td>41</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.473</td>
</tr>
<tr>
<td>GA_02_α</td>
<td>0.724</td>
<td>37</td>
<td>0.689</td>
<td>38</td>
<td>0.678</td>
<td>39</td>
<td>0.654</td>
<td>40</td>
<td>0.686</td>
</tr>
<tr>
<td>GA_03_α</td>
<td>0.969</td>
<td>8</td>
<td>0.970</td>
<td>4</td>
<td>0.953</td>
<td>19</td>
<td>0.895</td>
<td>34</td>
<td>0.946</td>
</tr>
<tr>
<td>GA_04_α</td>
<td>0.973</td>
<td>1</td>
<td>0.964</td>
<td>14</td>
<td>0.944</td>
<td>23</td>
<td>0.877</td>
<td>35</td>
<td>0.939</td>
</tr>
<tr>
<td>GA_05_α</td>
<td>0.940</td>
<td>24</td>
<td>0.940</td>
<td>24</td>
<td>0.948</td>
<td>21</td>
<td>0.854</td>
<td>36</td>
<td>0.920</td>
</tr>
<tr>
<td>GA_06_α</td>
<td>0.971</td>
<td>3</td>
<td>0.970</td>
<td>4</td>
<td>0.970</td>
<td>4</td>
<td>0.904</td>
<td>31</td>
<td>0.953</td>
</tr>
<tr>
<td>GA_07_α</td>
<td>0.967</td>
<td>12</td>
<td>0.964</td>
<td>14</td>
<td>0.972</td>
<td>2</td>
<td>0.935</td>
<td>26</td>
<td>0.959</td>
</tr>
<tr>
<td>GA_08_α</td>
<td>0.964</td>
<td>14</td>
<td>0.966</td>
<td>13</td>
<td>0.957</td>
<td>18</td>
<td>0.914</td>
<td>30</td>
<td>0.950</td>
</tr>
<tr>
<td>GA_09_α</td>
<td>0.970</td>
<td>4</td>
<td>0.968</td>
<td>10</td>
<td>0.959</td>
<td>17</td>
<td>0.916</td>
<td>29</td>
<td>0.953</td>
</tr>
<tr>
<td>GA_10_α</td>
<td>0.969</td>
<td>8</td>
<td>0.968</td>
<td>10</td>
<td>0.950</td>
<td>20</td>
<td>0.903</td>
<td>32</td>
<td>0.948</td>
</tr>
<tr>
<td>Average</td>
<td>0.940</td>
<td>1</td>
<td>0.933</td>
<td>2</td>
<td>0.926</td>
<td>3</td>
<td>0.875</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

In order to guarantee a fair comparison the average for α = 0 is computed without GA_01_0. Obviously, most algorithms perform best for a small value of α. This may be interpreted in the way that the job selection rule MAX RU has only a limited influence on the generation of good solutions. On the other hand the really poor performance of GA_01_0 which is the only parameter combination without domain-specific knowledge indicates clearly that the use of knowledge-based rules improves the efficiency ratio substantially. Furtheron, the greedy randomized algorithm RA_01_α yields results which compared to others are not very promising. It can be stated that the best performance in general is obtained by using a population size of 50. The importance of embedding mutation is documented by the poor results of GA_02_α where the mutation rate μ = 0. Finally, no evaluation on the basis of different tractabilities is presented because the results for the efficiency ratios as well as for the CPU-times do not differ markedly.
In Table 4 the increase of the efficiency ratio within a run is documented for those algorithms performing best at the end of a run and in addition for some algorithms of special interest.

**Table 4**

*Efficiency Ratios \( \phi_{\alpha\beta\theta\mu\nu\sigma\theta} \) after Production of \( \theta \) Individuals where \( \sigma = \text{small} \)

<table>
<thead>
<tr>
<th>( \theta / 1,000 )</th>
<th>0.1</th>
<th>0.5</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>25</th>
<th>100</th>
<th>500</th>
<th>1,000</th>
<th>rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA(_04)_0</td>
<td>0.609</td>
<td>0.669</td>
<td>0.725</td>
<td>0.793</td>
<td>0.844</td>
<td>0.926</td>
<td>0.950</td>
<td>0.964</td>
<td>0.973</td>
<td>1</td>
</tr>
<tr>
<td>GA(_07)_4</td>
<td>0.661</td>
<td>0.740</td>
<td>0.774</td>
<td>0.823</td>
<td>0.858</td>
<td>0.918</td>
<td>0.944</td>
<td>0.961</td>
<td>0.972</td>
<td>2</td>
</tr>
<tr>
<td>GA(_06)_0</td>
<td>0.679</td>
<td>0.762</td>
<td>0.797</td>
<td>0.853</td>
<td>0.880</td>
<td>0.930</td>
<td>0.956</td>
<td>0.965</td>
<td>0.971</td>
<td>3</td>
</tr>
<tr>
<td>GA(_06)_2</td>
<td>0.658</td>
<td>0.750</td>
<td>0.776</td>
<td>0.839</td>
<td>0.857</td>
<td>0.909</td>
<td>0.940</td>
<td>0.955</td>
<td>0.970</td>
<td>4</td>
</tr>
<tr>
<td>GA(_06)_4</td>
<td>0.645</td>
<td>0.712</td>
<td>0.748</td>
<td>0.824</td>
<td>0.855</td>
<td>0.907</td>
<td>0.945</td>
<td>0.963</td>
<td>0.970</td>
<td>4</td>
</tr>
<tr>
<td>GA(_09)_0</td>
<td>0.763</td>
<td>0.794</td>
<td>0.810</td>
<td>0.838</td>
<td>0.904</td>
<td>0.941</td>
<td>0.965</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RA(_01)_0</td>
<td>0.689</td>
<td>0.763</td>
<td>0.779</td>
<td>0.820</td>
<td>0.844</td>
<td>0.868</td>
<td>0.915</td>
<td>0.944</td>
<td>0.948</td>
<td>21</td>
</tr>
<tr>
<td>GA(_01)_0</td>
<td>0.043</td>
<td>0.053</td>
<td>0.060</td>
<td>0.079</td>
<td>0.091</td>
<td>0.183</td>
<td>0.323</td>
<td>0.454</td>
<td>0.473</td>
<td>41</td>
</tr>
</tbody>
</table>

No results for 100 individuals are available for GA\(_09\)_0 because due to its population size of \( \lambda = 250 \) decoding still not has taken place according to the genetic scheme of Section 4. From Table 4 we conclude that for a small number of produced individuals larger population sizes are more favourable. Really surprising is the fact that the greedy randomized algorithm RA\(_01\)_0 with up to 5,000 individuals performed nearly as well as the other GA's and significantly better than GA\(_04\)_0 which is the best after 1,000,000 individuals have been generated. Therefore one may derive the recommendation to use a greedy randomized algorithm which is much easier to implement than a GA, if only a small number of individuals has to be produced.

Table 5 documents the average CPU-times needed on an IBM RISC / 6000 model 550 for generating one individual.

**Table 5**

*Average CPU-Times \( \beta_{\delta\theta\sigma\mu\nu\sigma\theta} \) in Milliseconds and Standard Deviations for \( \sigma = \text{small} \)

<table>
<thead>
<tr>
<th>CPU</th>
<th>10.02</th>
<th>3.46</th>
<th>5.01</th>
<th>5.00</th>
<th>5.02</th>
<th>4.95</th>
<th>5.06</th>
<th>5.26</th>
<th>5.22</th>
<th>5.16</th>
<th>5.32</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dev.</td>
<td>6.34</td>
<td>2.10</td>
<td>3.08</td>
<td>3.11</td>
<td>3.17</td>
<td>3.11</td>
<td>3.11</td>
<td>3.29</td>
<td>3.16</td>
<td>3.17</td>
<td>3.27</td>
</tr>
</tbody>
</table>

Table 5 shows that an increase of the population size \( \lambda \) has only a minor influence on the CPU-times (population sizes of 100, 250, and 500 for GA\(_08\), GA\(_09\), and GA\(_10\)). The mutation rate \( \mu \) also seems to have no major impact (\( \mu \) lower for GA\(_02\), higher for GA\(_06\) and GA\(_07\)), while together with a higher crossover rate \( v \) for RA\(_01\) where \( \mu = 1 \) and \( v = 1 \) a substantial difference is caused. An explanation may be that first each gene has to be mutated
and second in each generation decoding, which only needs to be executed for the children, takes place twice as much as for the GA's. However, even though the CPU-times of RA_01 are roughly twice as high as those of the other GA's a comparison based upon the CPU-times also would result in identifying RA_01 to be superior to GA_04 up to 2,500 individuals produced. A CPU-time based comparison keeps the results of the GA's documented in Table 4 nearly unchanged while the resulting efficiency ratios of RA_01 are 0.664, 0.727, 0.763, 0.793, 0.820, 0.855, 0.888, 0.932, and 0.944. This may be seen as another way of performing a fair comparison of the RA with GA's where the results are similar to those obtained by our analysis. The CPU-times of GA_01 are lower because the time-consuming computation of the three knowledge-based criteria can be omitted.

The only result of interest that can be stated from testing the very small instances is that each of the parameter combinations found the optimal solution (computed with OSL release 2) by producing no more than 2,000 individuals within approximately 10 % of the time needed by OSL. No major differences between the parameter combinations have been observed. Unfortunately, due to memory overflows OSL was unable to determine the optimal solutions of any of the small instances.

7. Summary

We introduced a competitive approach of genetics-based learning in order to solve a problem from the field of resource-constrained project scheduling, namely the medium-to-short-term audit-staff scheduling problem. By virtue of generalizing the ideas of Bean (1994) as well as the gene representation of Dorndorf and Pesch (1995) it is possible to incorporate easily domain-specific knowledge into a GA. The decoding can be characterized as a parallel regret-based biased random sampling method where under special consideration of the objective function a job primarily selected is not necessarily scheduled. For the purpose of a comparison it is proved that the RA is a special case of a GA. The computational results indicate that a GA which has a medium population size and is based upon domain-specific knowledge performed best in general while for a small number of individuals to be generated a RA seems to be a promising alternative.

Acknowledgement

The authors are indebted to Professor Andreas Drexl and Professor Erwin Pesch for their critical and helpful comments as well as to Andreas Schirmer for improving the phrasing substantially.
Appendix

In the following the criteria JST, AST, and SST for job scheduling are defined mathematically.

Let

\[ \mathcal{A}_j = \{ a \mid k_{ja} > 0 \} \]

consist of all auditors working on job j. Further

\[ c_{\text{min}} = \min \{ c_t \mid 1 \leq t \leq T \} \]

are the cost of the cheapest period (normally the periods where work regularly takes place) within the planning horizon.

\[ \mathcal{E} = \{ t' \mid t \leq t' \leq T \land c_{\text{min}} < c_{t'} \leq c_t \} \]

contains the actual period t and all subsequent periods where the costs are not minimum but less than or equal to the costs of the actual period, while

\[ \mathcal{D} = \{ t' \mid t < t' \leq T \land c_{\text{min}} < c_{t'} < c_t \} \]

is a subset of \( \mathcal{E} \) consisting only of periods where the costs are less than in the actual period.

Let \( u \) denote the number of jobs on the longest path including j which are still unscheduled and which cannot be scheduled with minimum costs.

Let \( v = 1 \), if the costs of the actual period are small or setup times to other jobs occur, otherwise \( v = 0 \).

Let \( w_a \) the number of jobs of auditor a which are still unscheduled and which cannot be scheduled with minimum cost, then

\[ w_{\text{max}} = \max \{ w_a \mid a \in \mathcal{A}_j \} \]

is the maximum number of corresponding jobs for an auditor.

Then, for job j

\[
\text{JST}_j = \begin{cases} 
1, & \text{if } c_t = c_{\text{min}} \lor u \geq |\mathcal{E}| \\
\frac{1}{|\mathcal{A}_j| \cdot (|\mathcal{E}| - |\mathcal{D}|)}, & \text{if } c_t \neq c_{\text{min}} \land |\mathcal{E}| > u > |\mathcal{D}| \\
0, & \text{otherwise}
\end{cases}
\]

\[ \text{(18)} \]

\[
\text{AST}_j = \begin{cases} 
1, & \text{if } c_t = c_{\text{min}} \lor w_{\text{max}} \geq |\mathcal{E}| \\
\frac{1}{|\mathcal{A}_j| \cdot (|\mathcal{E}| - |\mathcal{D}|)}, & \text{if } c_t \neq c_{\text{min}} \land |\mathcal{E}| > w_{\text{max}} > |\mathcal{D}| \\
0, & \text{otherwise}
\end{cases}
\]

\[ \text{(19)} \]
The definitions of the above criteria (18) - (20) were obtained from the following observations:
If the actual period has minimum costs or there remains a number of unscheduled jobs, that
must be processed in periods where the costs are higher than in the actual one, then a value of
1 results. If there is a number of priods with costs equal to the actual one or setup times have
to be considered then a value is computed which is based upon - among others - the number of
auditors involved in processing job j.

References

review and an empirical analysis", in: Slowinski, R., and J. Weglarz (eds.): Advances in project scheduling,

Badiru, A.B., "Towards the standardization of performance measures for project scheduling heuristics", IEEE


Bartsch, T., and F. Salewski, "Audit-staff scheduling: methods for the medium-to-short-term level", presented
at Joint DGOR/NSOR Conference, Amsterdam, August 26, 1993.


Böttcher, L., and F. Salewski, "Audit-staff scheduling: methods for the short-term level", presented at Joint
DGOR/NSOR Conference, Amsterdam, August 26, 1993.

Chan, K.H., and B. Dodin, "A decision support system for audit-staff scheduling with precedence constraints


Davis, E.W., and J.H. Patterson, "A comparison of heuristic and optimum solutions in resource-constrained

Dodin, B., and K.H. Chan, "Application of production scheduling methods to external and internal audit


Goldberg, D.E., Genetic algorithms in search, optimization and machine learning, Addison-Wesley, Reading,
1989.


