ECONSTOR
Make Your Publications Visible.

A Service of

ZBW Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Drexl, Andreas; Salewski, Frank

Working Paper — Digitized Version

Distribution Requirements and Compactness Constraints in School Timetabling. Part II: Methods

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 384

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

Suggested Citation: Drexl, Andreas; Salewski, Frank (1996) : Distribution Requirements and Compactness Constraints in School Timetabling. Part II: Methods, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 384, Universität Kiel, Institut für Betriebswirtschaftslehre, Kiel

This Version is available at:
https://hdl.handle.net/10419/181063

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.

WWW.ECONSTOR.EU

Mitglied der Leibniz-Gemeinschaft

No. 384

Distribution Requirements and Compactness Constraints in School Timetabling. Part II: Methods

Drexl / Salewski

January 1996

Andreas Drexl, Frank Salewski
Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel,
Olshausenstr. 40, 24118 Kiel, Germany
Tel.&Fax ++49(0)431-880-1531
Email: Drexl@bwl.uni-kiel.de, Salewski@bwl.uni-kiel.de.

Abstract: Constraints of different types have to be regarded in school timetabling. In part I of this paper we modelled requirements for distributing large weekly teaching units into small peaces by the use of the multiple mode concept in a novel way. In addition, it has been shown that several types of constraints may be modelled using the unifying framework of partially renewable resources: no class, subject, room, and teacher overlaps; class, subject, room, and teacher unavailabilities; compactness constraints; preassignment constraints; lectures to be given simultaneously; lunch breaks, etc. The main contributions of part II of this paper are: We present two-phase parallel greedy randomized and genetic methods. In addition, we provide an instance generator for the generation of a representative set of instances. The generator along with a statistical model is used for a thorough experimental evaluation of the methods. Computational results show that the methods solve the instances investigated close to optimality.

Keywords: Timetabling, distribution requirements, multiple modes, mode identity, compactness constraints, partially renewable resources, greedy randomized / genetic algorithms

1. Introduction

The school timetabling problem requires to schedule a set of meetings between teachers and pupils over a set of time periods, where some resources must be available and several additional constraints have to be met. The timetabling problem includes a large variety of problems on different levels of an education system. What makes the difference between timetabling (for schools) and course scheduling (for universities) is explained in part I of this paper. In addition, there our work is related to that of other researchers.

Large weekly teaching units have to be split into small teaching pieces in school timetabling. In addition these small pieces have to be distributed over (the days of) the week. This splitting and distributing (which we summarize under the term distribution requirements) is modelled by the use of the multiple mode concept with mode identity constraints in a novel way. In addition, different constraints have to be satisfied: No class, subject, room, and teacher overlaps, are allowed; one has to take care of class, subject, room, and teacher availabilities; compactness constraints have to be taken into account; preassignment of lectures must be possible; some lectures have to be given simultaneously; lunch breaks have to be regarded, etc. Note that compactness constraints typically require no free time (other than lunch breaks or time to move from one building to another) between lessons for the pupils. Noteworthy, that most of the constraints may modelled using the unifying framework of partially renewable resources.

We present greedy randomized and genetic algorithms. For the greedy randomized algorithms we provide several priority rules. Each basic algorithm with one of the priority rules and one set of control parameters yields one specific algorithm. But how to decide about how much CPU-time to spend in which of the algorithmic variants? We do this "automatically" by sequential analysis (hypothesis testing), or in other words, we exclude inferior variants from further consideration based on statistical tests. In addition, we provide an instance generator for the generation of a representative set of instances. The generator along

with a statistical model is used for a thorough experimental evaluation of the methods. Computational results show that the methods solve the instances investigated to suboptimality.

The remainder of the paper is organized as follows: In Section 2 we present the greedy randomized algorithms. In Section 3 the genetic algorithms are introduced. Section 4 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 5. Finally, Section 6 provides a brief summary and some conclusions. Fundamentals of sequential analysis are stated in the Appendix.

2. Two-Phase Parallel Greedy Randomized Algorithms

From part I of this paper it is clear, that the timetabling problem under consideration is NP-complete and NP-hard, respectively. Therefore, the only line of attack for tackling practical problem sizes comprising thousands of binary variables (cp. Section 4.1) is provided by approximation methods.

Deterministic greedy priority rule-based methods have been widely adopted in scheduling (cp. the surveys in Haupt 1989, Panwalkar and Iskander 1977). Partial schedules are extended, starting with the empty set of scheduled jobs, i.e. the intialization $x_{jmt} := 0$ for all the decision variables. These methods are commonly used when scheduling large problem instances, yield only one solution for an instance, even if applied several times. Having in mind that this solution may be arbitrarily bad or even infeasible, determinism seems to be a major deficiency of such methods. Semi-greedy (cp. Hart and Shogan 1987), greedy randomized (cp. Feo, Resende, and Smith 1994, Laguna, Feo, and Elrod 1994), or regret-based biased random sampling methods (cp. Drexl 1991, 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 step every schedulable job may be chosen, though those sharing higher probabilities 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. Moreover, no tiebreaker needs to be specified for randomized methods, since ties cannot occur.

Generally, common priority rule-based methods for (project) scheduling are distinguished to be serial or parallel (cp. the early work of Kelley 1963 and the recent improvements obtained by Kolisch 1996). The former schedule one of the precedence-feasible jobs as early as possible w.r.t. resource constraints and the latter proceed chronologically over all periods of the planning horizon trying to schedule in each period as many jobs as possible. While the basic principle of both methods is simple and intuitive, some specific details have to be designed appropriately in order to get reliable and fast methods for the problem class under consideration.

Section 2 is organized as follows: First we give an outline of the Two-Phase Greedy Randomized Method (TPGRM). Second, we present the (single) priority rule used in phase 1. Third, the priority rules of phase 2 are described. Finally, we give the basic ideas of sequential tests which allow to improve the greedy randomized algorithms.

2.1 Outline of the Algorithmic Scheme

The TPGRM works as follows: In phase 1 a BTU (Basic Teaching Unit) is selected and one of the available modes is assigned to that BTU. (Note that from a conceptual point of view we might distinguish between step 1 of phase 1 where a BTU is selected, while in step 2 of phase 1 one of the available modes is assigned to the chosen BTU. This is exactly what we do in the genetic algorithms described below.) In phase 2 the jobs are scheduled within a parallel scheduling scheme on account of priority rules.

Let denote ω a priority rule. Whenever necessary we will use the superscripts (ω^0) ω^1 and ω^2 with the following meaning: ("0" resembles to a priority rule used within step 1 of phase 1 for BTU-selection) "1" points to a priority rule used within step 2 of phase 1 for mode-selection, while "2" relates to a priority rule used within phase 2 for job-selection of the two-phase algorithm. Moreover, let I denote the set of (locally) available elements for which we want to compute priority values.

In Kolisch 1995 it has been shown for the multiple resource-constrained project scheduling problem that regret-based biased random sampling methods are superior to pure and biased random sampling algorithms, respectively. Hence, in the following we employ the former which works as follows:

Let us consider e.g. the priority rule ω = SPT (Shortest Processing Time). Then from the local context it will be evident what I currently stands for. Let ω_i denote the specific processing time of $i \in I$. Then

$$\bar{\omega}_i := \max \{ \omega_h \mid h \in I \} - \omega_i \qquad (\forall i \in I)$$

gives a regret-based priority value, i.e. in case of a "minimizing" priority rule the point of reference is the alternative with the maximum entry. We modify $\bar{\omega}_i$ to

$$\bar{\bar{\omega}}_i := (\bar{\omega}_i + \epsilon)^b \qquad (\forall i \in I)$$

and compute probabilities $\tilde{\omega}_i$ as follows:

$$\tilde{\omega}_i := \bar{\bar{\omega}}_i \,/\, \sum_{h \in I} \bar{\bar{\omega}}_h \qquad (\forall i \in I)$$

Clearly, $\epsilon > 0$ makes sure that each local decision alternative may be chosen with a positive probability; b transformes the term (.) exponentially and thus gives way to control the generation of probabilities. Note that $b = 0$ produces the priority rule RANDOM.

In case we have e.g. the priority rule ω = LPT (Longest Processing Time) we only have to replace the computation of $\bar{\omega}_i$ as follows:

$$\bar{\omega}_i := \omega_i - \min \{ \omega_h \mid h \in I \} \qquad (\forall i \in I)$$

I.e. in case of a "maximizing" priority rule the point of reference is the alternative with the minimum entry.

When computing $\bar{\bar{\omega}}_j$ within (step 2 of) phase 1, we use the control parameter b. Analogously we use the control parameter c in phase 2, while the computation scheme remains unchanged. (Note that in Subsection 4.2 an additional control parameter a will be introduced within step 1 of phase 1.)

2.2 Phase 1 Priority Rule

Now we describe the priority rule MIN_COST_P1, which is used for the selection of BTUs and modes.

MIN_COST_P1 (MINimal COST in Phase 1): Select the BTU u ($1 \leq u \leq U$) and the mode m ($0 \leq m \leq M_u$) in accordance with costs c_{um}, i.e.

$$\omega^1_{um} := c_{um} \qquad (1 \leq u \leq U;\ 0 \leq m \leq M_u)$$

Note that due to the static nature of MIN_COST_P1 used within (step 1 and step 2 of) phase 1 the order in which the BTUs are considered is of no relevance. In the case of dynamic rules (cp. Subsection 4.2) the introduction of an additional rule ω^0 for BTU selection is appropriate within step 1 of phase 1.

In Drexl and Salewski 1994 two other priority rules for phase 1 have been developed and tested experimentally. The first one, denoted as MIN_JOB takes care of the minimum number of jobs j with $d_{jm} \neq 0$. The second one, denoted as MAX_JOB looks for the maximum number of jobs j with $d_{jm} \neq 0$. Both will not be investigated in more detail in the sequel, because they have been outperformed by MIN_COST_P1.

2.3 Phase 2 Priority Rules

Let u_j denote the BTU the job j belongs to. Then, after phase 1 the mode m_{u_j} is known in which BTU u_j has to be processed. Let t denote the time instant under consideration within the parallel method. Moreover, let EJ denote the set of Eligible Jobs at time instant t; clearly, EJ consists of the set of jobs, which *(i)* are not finished, which *(ii)* do not violate the constraints (5) and (6) introduced in Subsection 3.1 of part I of this paper, and for which *(iii)* $t + d_{jm_{u_j}} - 1 \leq T \ \wedge \ (t + d_{jm_{u_j}} - 1) \notin N_{jm_{u_j}}$ holds.

Now, a short description of the priority rules is given, which are used for the selection of the jobs within the parallel method.

MIN_COST_P2 (MINimal COST in Phase 2): Selecting "cheap" jobs j ($j \in EJ$) is preferred; more precisely:

$$\omega_j^2 := c_{u_j m_{u_j}} \qquad (\forall\, j \in EJ)$$

MAX_COST_P2 (MAXimal COST in Phase 2): Selecting "expensive" jobs j ($j \in EJ$) is preferred.

SPT_JOB (Shortest Processing Time of JOBs): Selecting jobs j ($j \in EJ$) with short processing times is preferred; more precisely:

$$\omega_j^2 := d_{j m_{u_j}} \qquad (\forall\, j \in EJ)$$

LPT_JOB (Longest Processing Time of JOBs): Selecting jobs j ($j \in EJ$) with long processing times is preferred.

MIN_TRU_JOB (MINimum Total Resource Usage of JOBs): Selecting jobs j ($j \in EJ$) with a minimum total resource usage is preferred. With $\lceil \delta \rceil$ as the smallest integer greater than δ, more precisely we compute:

$$\omega_j^2 := \sum_{r=1}^{R} \left\lceil k_{j m_{u_j} r} \cdot d_{j m_{u_j}} \right\rceil \qquad (\forall\, j \in EJ)$$

MAX_TRU_JOB (MAXimum Total Resource Usage of JOBs): Selecting jobs j ($j \in EJ$) with a maximum total resource usage is preferred.

MIN_TRU_BTU (MINimum Total Resource Usage of BTUs): Selecting BTUs u_j ($j \in EJ$) with a minimum total resource usage is preferred. Recall that a_{u_j} and e_{u_j} denote the first and the last job of BTU u_j, respectively. Then more precisely we compute:

$$\omega_j^2 := \sum_{i=a_{u_j}}^{e_{u_j}} \sum_{r=1}^{R} \left\lceil k_{i m_{u_i} r} \cdot d_{i m_{u_i}} \right\rceil \qquad (\forall\, j \in EJ)$$

MAX_RRU (MAXimum Relative Resource Usage): The larger the ratio "resource demand / resource availability", the scarcer the resource. Therefore, selecting jobs j ($j \in EJ$) with scarce resources is preferred. More precisely we define

$$B(j) := \sum_{r=1}^{R} \left\lceil k_{j m_{u_j} r} \right\rceil \sum_{q=t}^{t+d_{j m_{u_j}}-1} \min \{ K_{r\pi} \mid 1 \le \pi \le \Pi;\, q \in \mathcal{P}_\pi \} \qquad (\forall\, j \in EJ)$$

and then get:

$$\omega_j^2 := \sum_{r=1}^{R} \left\lceil k_{jm_{u_j}r} \cdot d_{jm_{u_j}} \right\rceil / B(j) \qquad (\forall j \in EJ)$$

Note that $\lceil k_{jmr} \rceil$ equals 1 by definition, if job j uses resource r in mode m (0, otherwise). Thus, B(j) is the sum of the minima for all the resources required by job j.

MAX_AVA (MAXimum AVAilable jobs decrease): The smaller the decrease of the cardinality of EJ after job $j \in EJ$ has been selected, the more preferable it is to select that job. More precisely we denote EJ' to be the updated set of eligible jobs and then get:

$$\omega_j^2 := |\{i \mid i \in EJ'\}| \qquad (\forall j \in EJ)$$

MIN_BTU (MINimum number of BTUs already selected): The less jobs j of BTU u_j with duration $d_{jm} \neq 0$ have been selected so far, the more preferable it is to select such jobs. More precisely we define

$$A := |\{i \mid a_{u_i} \leq i \leq e_{u_i} \wedge d_{im_{u_i}} \neq 0\}|$$

and then get:

$$\omega_j^2 := \sum_{i=a_{u_j}}^{e_{u_j}} \sum_{q=t}^{t+d_{im}-1} (x_{jm_{u_j}q}/A) \qquad (\forall j \in EJ)$$

Recall that t denotes the time instant under consideration within the parallel method.

MAX_BTU (MAXimum number of BTUs already selected): The more jobs j of BTU u_j with duration $d_{jm} \neq 0$ have been selected so far, the more preferable it is to select such jobs.

Three other priority rules for phase 2 (MAX_TRU_BTU, MIN_RRU, MIN_AVA) have been developed and tested experimentally in Drexl and Salewski 1994. They will not be investigated, because they have been outperformed by the others.

The following has to be noticed: Having finished phase 2 there might be some jobs which could not have been selected. Then constraints (3) and (4) require all jobs j of the corresponding BTU u_j to be reassigned to mode $m_{u_j} = 0$.

2.4 Sequential Analysis

Clearly, the performance of the two-phase parallel method depends on the rules employed and the control parameters used. In fact, we get a huge variety of specific algorithms and the question arises how much CPU-time to spend on a specific variant for a specific problem

instance in order to get the "best possible" results. One answer to that question may be given by sequential analysis of the parameter space.

Let denote

$\underline{\Omega}^1$ = {the set of priority rules (of step 2) of phase 1}

$\underline{\Omega}^2$ = {the set of priority rules of phase 2}

as well as $\omega^1 \in \underline{\Omega}^1$ and $\omega^2 \in \underline{\Omega}^2$ specific rules. Note that we consider the priority rules as "parameters" of the overall two-phase parallel method. Moreover, let denote $\underline{B} = \{b \geq 0\}$ and $\underline{C} = \{c \geq 0\}$ the parameter spaces of both phases. In order to make the parameter spaces computationally tractable we restrict them to:

$\underline{B} \subseteq \{\{0\},\{1,2\},\{3,4\}\}$ (parameter b (of step 2) of phase 1)

$\underline{C} \subseteq \{\{0\},\{1,2\},\{3,4\}\}$ (parameter c for phase 2)

Clearly, for $b = 0$ we have random sampling. $b \in \{1,2\}$ slightly transforms the selection probabilities, whereas $b \in \{3,4\}$ enforces the differences of the selection probabilities up to a large extent. The same holds true for the parameter c. When the cardinality of a subset equals 2, one element is choosen randomly.

Thus we get the overall parameter space $\Theta = \{\underline{\Omega}^1 \times \underline{\Omega}^2 \times \underline{B} \times \underline{C}\}$. Θ will partitioned into Γ disjoint subspaces $\theta_1, \theta_2, \ldots, \theta_\Gamma$ (where the case of pure random sampling in both phases is chosen only once). We have $\{\omega^1 \times \omega^2 \times b \times c\} \subset \Theta$ and $|\omega^1| = |\omega^2| = |b| = |c| = 1$. Moreover, it is $\Theta = \cup_{\gamma=1}^{\Gamma} \theta_\gamma$ and $\theta_\gamma \cap \theta_\eta = \phi$ for $\gamma \neq \eta$.

Now, we proceed as follows: First, we define an arbitrary total order on the supspaces. Second, for each subspace 30 trials are performed. Third, the q-quantil ξ of each subspace is estimated. Fourth, perform a prespecified number of (additional) trials iteratively as follows: *(i)* Choose parameters out of the next subspace at random. *(ii)* Compute a schedule using the TPGRM-method. *(iii)* Perform the Sequential Probability Ratio Test (SPRT; see Appendix) and discard the subspace under consideration if the null-hypothesis is rejected. For the parameters α, β, p^0, and p^1 introduced in the Appendix we set in the following $\alpha = 0.05$, $\beta = 0.1$, $p^0 = 0.2$, and $p^1 = 0.05$, respectively. For the sake of shortness this method is abbreviated as TPGRM/SPRT in what follows. Note that this idea has already been successfully used for solving lotsizing and scheduling problems in Haase 1994 and Drexl and Haase 1996.

3. Genetic Algorithms

As the name suggests, genetic algorithms (GAs) are motivated by the theory of evolution. Early work dates back to Rechenberg 1973, Holland 1975, and Schwefel 1977; see also Goldberg 1989, Mühlenbein, Gorges-Schleuter, and Krämer 1988, Liepins and Hilliard 1989, and

Michalewicz 1992. As stated by Grefenstette 1987, "GA's are not well suited for fine-tuning structures which are very close to optimal solutions". If a competitive GA is desired, it is therefore essential, to incorporate (local search) improvement operators or in other words domain-specific knowledge (cp. Johnson 1990). Storer, Wu, and Vaccari 1992, Bean 1994, Pesch 1994, and Dorndorf and Pesch 1995 among others describe how to take into account improvement operators. Note that a GA for a timetabling problem has been recently provided by Colorni, Dorigo, and Maniezzo 1992 as well, but their model is less general then ours and only limited details about the method and the computational results are provided.

Section 3 is organized as follows: First, we give a description of the GA. Second, we present the priority rules used for BTU-, mode- and job-selection.

3.1 Description of the Algorithm

To start with, let denote $\iota \in \mathbb{N}^+$ the number of individuals, $\lambda \in \mathbb{N}^+$ the population size, $\nu \in [0,1]$ the crossover rate, and $\mu \in [0,1]$ the mutation rate, respectively. Now, first the basic genetic algorithm and the decoding of individuals are explained. Second, an appropriate representation of genes is introduced.

random initialization of the first generation;
for generation = 1 to ι/λ do
{
crossover with rate ν;
random *mutation* of children's genes with rate μ;
decoding of individuals;
reproduction where the individual with the best objective function value survives in any case (survival of the fittest) and the others are selected with probability proportional to their objective function value;
}

Clearly, a mutation is only allowed on the children produced by the crossover in order to guarantee the survival of the fittest.

For the purpose of decoding each individual primarily consists of a number of genes each of which is representing one priority rule for each of the activities "selection of a BTU", "selection of a mode for that BTU", and for "selection of jobs of that BTU for scheduling", respectively. At the end of this section a gene will be defined more precisely.

Now, a serial one-phase GA is presented. As stated in Section 2, serial means that the algorithm does not proceede chronologically (in contrast to parallel). Contrary to the two-phase algorithm presented above one-phase accounts for the fact that the activities outlined in the preceeding paragraph w.r.t. decoding are performed "BTU after BTU". In

Drexl and Salewski 1994 a two-phase serial and a two-phase parallel GA have been developed and tested experimentally as well. Both will not be described here, because they have been outperformed by the serial one-phase GA.

Let denote EU, EM, and EJ the set of eligible BTUs, modes, and jobs, respectively. Moreover, let denote FJ the set of finished jobs. Finally, $K^{!}_{r\pi}$ denotes the left-over capacity for all r and π. Like in Section 2, all the decision variables are initialized to $x_{jmt} := 0$.

```
K'_rπ := K_rπ      (1 ≤ r ≤ R; 1 ≤ π ≤ Π);                    /* left-over capacities */
x_jmt := 0         (1 ≤ u ≤ U; a_u ≤ j ≤ e_u; 0 ≤ m ≤ M_u; 1 ≤ t ≤ T);
EU  := {1,...,U};
FJ  := φ;
g := 1;     /* number of the gene currently decoded from the considered individual */
while EU ≠ φ do
  {
  u     := SelectBTU(EU);                                    /* BTU-selection */
  EM    := {0,...,M_u};
  while EM ≠ φ do
    {
    m_u := SelectMode(EM);                                   /* mode-selection */
    EJ  := {j | a_u ≤ j ≤ e_u ∧ d_jmu ≠ 0};
    FJ  := FJ ∪ {j | a_u ≤ j ≤ e_u ∧ d_jmu = 0};
    while EJ ≠ φ do
      {
      j   := SelectJob(EJ);                                  /* job-selection */
      ScheduleJob(j);                                        /* job scheduling */
      EJ  := EJ \ {j};
      FJ  := FJ ∪ {j};
      update K'_rπ;
      }
    g := g + 1;
    }
  };
```

For the sake of clarity some explanations of the procedures have to be given: The procedure "SelectBTU(EU)" selects a BTU u according to priority rule ω^0 from gene g. "SelectMode(EM)" selects a mode m according to priority rule ω^1 from gene g. The procedure "SelectJob(EJ)" selects a job j according to priority rule ω^2 from gene g. "ScheduleJob(j)" schedules job j as early as possible w.r.t. resource and precedence constraints in accordance with the serial scheme.

Clearly, like in Section 2, after decoding there might be some jobs which could not have been scheduled due to insufficient left-over capacities $K^{l}_{r\pi}$. Then constraints (3) and (4) introduced in Subsection 3.1 of part I of this paper require all jobs j of the corresponding BTU u_j to be reassigned to mode $m_{u_j} = 0$.

Finally, an appropriate *representation of genes* has to be developed. Goldberg 1989 suggested to represent a gene as a job. Clearly, when applied to problems with precedence constraints it is difficult to maintain feasibility of offsprings. In order to overcome this deficiency Dorndorf and Pesch 1995 defined a gene to be a priority rule. Unfortunately, this approach has some drawbacks as well. Salewski and Bartsch 1994 define a gene as a tuple (ω, a, z) consisting of a rule ω, a control parameter a, and a random number z, respectively. Obviously, in the case of a constant choice of a and z we get the idea of Dorndorf and Pesch as a special case. Another special case may derived by fixing ω and a (roughly speaking, this is the idea proposed by Bean 1994) within a run (defined as the production of ι individuals) and coding only the random number z. Clearly, when three activities have to be encoded simultaneously (as it is the case here for BTU-, mode- and job-selection) this idea can be generalized as follows: Define a gene to be a tuple $(\omega^0, a, z^0; \omega^1, b, z^1; \omega^2, c, z^2)$. Then fix the rules ω^0, ω^1 and ω^2 and the corresponding control parameter a, b and c for each activity within a run, and encode the genes by the use of three random numbers z^0, z^1, and z^2, respectively.

3.2 Priority Rules for BTU-, Mode- and Job-Selection

In the sequel priority rules for BTU-, mode- and job-selection will be described in this order. To start with two *BTU-selection* priority rules are presented first.

MAX_BLK (MAXimal number of BLocKs): Selecting BTUs u ($1 \le u \le U$) with many blocks is preferred; more precisely:

$$\omega^0_u := | e_u - a_u + 1 | \qquad (\forall\, u \in EU)$$

MAX_RES_BTU (MAXimum number of RESources required by BTUs): Selecting BTUs u ($1 \le u \le U$) which require many of the resources is preferred; more precisely:

$$\omega^0_u := | \{ r \mid k_{a_u m_u r} > 0 \} | \qquad (\forall\, u \in EU)$$

For the *selection of mode* $m \in EM$ to the already fixed BTU $\bar{u} \in EU$ only the priority rule ω^1 = **MIN_COST_P1** is used. Clearly, this rule is identical to the MIN_COST_P1 rule described in Subsection 2.2.

Given the fixed BTU $\bar{u}$ in the chosen mode $\bar{m}_{\bar{u}} \in EM$ for *selecting job* $j \in EJ$ two priority rules are employed. While the first rule is identical to the LPT_JOB-rule, the second one is a variant of the MAX_RRU-rule.

MAX_RES_JOB (MAXimum number of RESources required by JOBs): Selecting jobs j ($\forall j \in EJ$) which require many of the resources is preferred; more precisely:

$$\omega_j^2 := |\{r \mid k_{j\bar{m}_{\bar{u}_j} r} > 0\}| \qquad (\forall j \in EJ)$$

Clearly, a priori any of the rules presented in Subsection 2.3 could be used here. But, on account of the differences in the basic algorithmic schemes of the TPGRM and the GA, the two rules mentioned above provided superior results.

4. Experimental Performance Analysis

Before presenting the performance measures in Subsection 4.4 the generation of test instances (Subsection 4.1), a statistical model (Subsection 4.2), and the experimental design (Subsection 4.3) are introduced.

4.1 Generation of Test Instances

Even in current literature, the systematic generation of test instances does not receice 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 systematical structure to infer any general properties. Thus, even if an algorithm performs well on some practical instances, it is not guaranteed that it will continue to do so on other instances. 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 problem. 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. For a start, we carried out interviews with several experts in school timetabling for understanding of the peculiarities of the primary and secondary schools. Then, to ensure a systematic and consistent generation of the instances, for each of the parameters a domain and a discrete distribution function on the domain were defined, based on 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 timetabling in primary and secondary schools as close as possible, yet to employ a systematic design for the generation procedure.

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 NP-completeness of the associated feasibility problem, it was possible to rig up the design of the (complicated) generation procedure in a way that for each constructed instance there exists (by the introduction of the dummy mode 0) at least one provably feasible solution.

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

The size σ of an instance is measured in terms of the number of binary variables x_{jmt}. It depends on the parameters J, M_u, and T, respectively. J depends on the parameters H, F_h, and B_{hf}, respectively, M_u results from the transformations of the M_{hf}. An estimate of the number of variables is given by:

$$\Sigma_{h=1}^{H} \Sigma_{f=1}^{F_h} (B_{hf} \cdot M_{hf} \cdot T)$$

Five instance sizes σ will be considered: $\mathcal{XS}$ (= extra small), $\mathcal{TY}$ (= tiny), $\mathcal{SM}$ (= small), $\mathcal{MD}$ (= medium), and $\mathcal{LG}$ (= large), respectively. The sizes $\mathcal{SM}$, $\mathcal{MD}$, and $\mathcal{LG}$ represent instances which may be observed in small- to large-sized schools in practice. The sizes $\mathcal{XS}$ and $\mathcal{TY}$ form instances which are solvable to optimality with commercial MIP-solvers like LINDO and OSL. The average relation between the sizes and the number of binary variables is approximately as follows: $\mathcal{XS} \approx$ 2,300; $\mathcal{TY} \approx$ 18,000; $\mathcal{SM} \approx$ 54,000; $\mathcal{MD} \approx$ 122,000; $\mathcal{LG} \approx$ 610,000. Moreover, Table 10 (cp. Subsection 5.3) relates the problem size to the average number of BTUs, jobs, and resources, respectively.

The tractability τ of an instance is intended to reflect how easy or how difficult that particular instance is to solve. For the purpose of this study, we take the number of feasible solutions disregarding the dummy mode 0 as an estimate of the tractability. The tractability depends *(i)* on the number and tightness of the resource constraints, *(ii)* on the number of forbidden periods $\mathcal{N}_{jm}$, and *(iii)* on the number of precedence relations $\mathcal{E}_u$. Three tractabilities will be considered: $\mathcal{E}$ (= easy), $\mathcal{MD}$ (= medium), and $\mathcal{HD}$ (= hard), respectively. On the average, the easy instances (with resource availability four times as much as the hard ones) have

several feasible solutions, the medium ones (with resource availability two times as much as the hard ones) have a couple of them, while for the hard ones only a few feasible solutions exist.

4.2 Statistical Model

First we would like to point out that any greedy (randomized) algorithm is a special case of a GA, if the population size λ, the crossover rate ν, and the mutation rate μ equal 1 (cp. Salewski and Bartsch 1994). Clearly, then only one child with a completely new initialization is generated, and this is exactly what greedy algorithms do. On account of that, we are now able to provide a unifying statistical model for the methods presented in Sections 2 and 3.

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

ω^0 : priority rule employed for BTU-selection

ω^1 : priority rule employed for mode-selection

ω^2 : priority rule employed for job-selection

a : control parameter for BTU-selection

b : control parameter for mode-selection

c : control parameter for job-selection

λ : population size

μ : mutation rate

ν : crossover rate

σ : size of instances

τ : tractability of instances

ι : 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 $(\underline{\Omega}^0, \underline{\Omega}^1, \underline{\Omega}^2, \underline{A}, \underline{B}, \underline{C}, \underline{L}, \underline{M}, \underline{N}, \underline{S}, \underline{T}, \underline{I})$, where

$\underline{\Omega}^0$ is a set of priority rules for BTU-selection

$\underline{\Omega}^1$ is a set of priority rules for mode-selection

$\underline{\Omega}^2$ is a set of priority rules for job-selection

$\underline{A}$ $\subseteq \mathbb{R}_{\geq 0}$ is a set of values for the BTU-selection control parameter

$\underline{B}$ $\subseteq \mathbb{R}_{\geq 0}$ is a set of values for the mode-selection control parameter

$\underline{C}$ $\subseteq \mathbb{R}_{\geq 0}$ is a set of values for the job-selection control parameter

$\underline{L}$ $\subseteq \mathbb{N}^+$ is a set of population sizes

$\underline{M}$ $\subseteq [0,1]$ is a continuous interval of mutation rates

$\underline{N}$ $\subseteq [0,1]$ is a continuous interval of crossover rates

$\underline{S}$ $\subseteq$ {extra small ($\mathcal{X}$), tiny ($\mathcal{T}$), small ($\mathcal{S}$), medium ($\mathcal{M}$), large ($\mathcal{L}$)} is a set of sizes

$\underline{T}$ $\subseteq$ {easy ($\mathcal{E}$), medium ($\mathcal{M}$), hard ($\mathcal{H}$)} is a set of tractabilities

$\underline{I}$ $\subseteq \mathbb{N}^+$ is a set of numbers of individuals ■

Definition 2: A run of an experiment $(\underline{\Omega}^0, \underline{\Omega}^1, \underline{\Omega}^2, \underline{A}, \underline{B}, \underline{C}, \underline{L}, \underline{M}, \underline{N}, \underline{S}, \underline{T}, \underline{I})$ is a tuple $(\omega^0, \omega^1, \omega^2, a, b, c, \lambda, \mu, \nu, \sigma, \tau, \iota) \in \underline{\Omega}^0 \times \underline{\Omega}^1 \times \underline{\Omega}^2 \times \underline{A} \times \underline{B} \times \underline{C} \times \underline{L} \times \underline{M} \times \underline{N} \times \underline{S} \times \underline{T} \times \underline{I}$. ■

The outcome of a run is - for each instance attempted - summarized in terms of two result variables. One, $\mathcal{BF}_{\omega^0 \omega^1 \omega^2 abc\lambda\mu\nu\sigma\tau\iota}$ denotes the objective function value of the best solution found after ι individuals have been generated in that run. Two, $\mathrm{CPU}_{\omega^0 \omega^1 \omega^2 abc\lambda\mu\nu\sigma\tau\iota}$ 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.

4.3 Experimental Design

Due to the computational effort required to attempt a sample of all sizes $\underline{S}$, the scope of the experiment was limited to include only extra small ($\mathcal{X}$) and tiny ($\mathcal{T}$) instances. Though no obstacle for using the developed methods 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 larger instances (cp. e.g. Davis and Patterson 1975 and Badiru 1988). All the tractabilities were considered, i.e. $\underline{T}$ = {easy ($\mathcal{E}$), medium ($\mathcal{M}$), hard ($\mathcal{H}$)}.

Of each instance class (σ,τ), ten instances were considered in the experiment. After pretests not further documented they were tackled by the algorithms $(\omega^0, \omega^1, \omega^2) \subset (\underline{\Omega}^0, \underline{\Omega}^1, \underline{\Omega}^2)$ with

$\underline{\Omega}^0$ = {MAX_BLK, MAX_RES_BTU},
$\underline{\Omega}^1$ = {MIN_COST_P1}, and
$\underline{\Omega}^2$ = {MIN_COST_P2, MAX_COST_P2, SPT_JOB, LPT_JOB, MIN_TRU_JOB, MAX_TRU_JOB, MIN_TRU_BTU, MAX_RRU, MAX_AVA, MIN_BTU, MAX_BTU, MAX_RES_JOB},

and the control parameter values $(a, b, c) \subset (\mathcal{A}, \mathcal{B}, \mathcal{C})$ were restricted to $\mathcal{A} = \mathcal{B} = \mathcal{C} = \{0, 1, 2, 3, 4\}$. Once more, after pretests not further documented the genetic parameters (λ, μ, ν) of the GA were restricted to $\lambda = \{1, 100\}$, $\mu = \{0.01, 1.0\}$, and $\nu = \{0.5, 1.0\}$. From this it follows that a specific combination of algorithm and parameters can be identified by ω^0, ω^1, ω^2, a, b, c, λ, μ, and ν. In addition, ϵ was set to 1.

4.4 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 $BF_{\omega^0\,\omega^1\,\omega^2\,abc\lambda\mu\nu\sigma\tau\iota}$ of the algorithm $(\omega^0, \omega^1, \omega^2)$ using the control parameter values (a, b, c) and the genetic parameter values (λ, μ, ν) for an instance class (σ, τ) after the individual ι in a run is computed, as *(i)* the average over all instances of that class attempted, and *(ii)* as the ratio of the objective function value of the best known solution for each instance attempted and $\mathcal{BF}_{\omega^0\,\omega^1\,\omega^2\,abc\lambda\mu\nu\sigma\tau\iota}$ (cp. the rule efficiency ratio proposed by Badiru 1988). Clearly, $BF_{\omega^0\,\omega^1\,\omega^2\,abc\lambda\mu\nu\sigma\tau\iota} \in [0,1]$; a ratio of one indicates that during ι individuals the best value currently known has been found, while a ratio of zero reflects the fact that no feasible ratio was generated at all. STD will be used to denote the STandard Deviation. $CPU_{\omega^0\,\omega^1\,\omega^2\,abc\lambda\mu\nu\sigma\tau\iota}$ denotes the average running time for producing one individual.

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 instance and the factor to be tested. Clearly, the average values can be interpreted as approximations of the expected values of the performance measures.

5. Computational Results

The methods presented in the preceeding sections have been coded in C and implemented on an IBM RISC 6000 model 550 workstation. The outline of this section is as follows: In Subsection 5.1 we report on the generation of benchmark solutions. Subsection 5.2 provides the results of the methods TPGRM and TPGRM/SPRT, while the results of the GA are reported in Subsection 5.3.

5.1 Generation of Benchmark Solutions

In order to evaluate the performance of the heuristics their results are compared with the best results available. Therefore, we first tried to solve extra small $(\mathcal{X})$ and tiny $(\mathcal{T})$ instances by

the use of the standard MIP-solver LINDO. Table 1 presents the results for the 30 instances (σ, τ) with $\sigma = \mathcal{X}$, $\tau = \{\mathcal{E}, \mathcal{M}, \mathcal{H}\}$, where we generated 10 instances for each combination of σ and τ.

Table 1: Computational Results of the MIP-Solver

τ	$\mathcal{E}$	$\mathcal{M}$	$\mathcal{H}$
SOL	8	5	2
CPU	69.34	193.21	590.39

SOL denotes the number of instances which could be solved optimally within a time limit imposed by 1,000,000 iterations. CPU provides the average computation times in minutes. The results indicate that the instances become significantly harder in terms of the required computational effort as a function of the tractability τ.

Second, we tried to solve tiny ($\mathcal{T}$) instances by the use of LINDO. The results for the 45 instances (σ, τ) with $\sigma = \mathcal{T}$, $\tau = \{\mathcal{E}, \mathcal{M}, \mathcal{H}\}$, where we generated 15 instances for each combination of σ and τ, are as follows: For $\tau = \mathcal{E}$ still 7 of the 15 instances could be solved optimally within the imposed time limit, whereas for $\tau = \mathcal{M}$ ($\mathcal{H}$) only 1 (0) instance could be solved. Note that LINDO terminated prematurely for $\sigma = \mathcal{S}$ when reading the MPS-file.

5.2 Computational Results of TPGRM and TPGRM/SPRT

In the following first we present results of the basic method TPGRM. The experimental design is as follows ($\underline{L} = 1$, $\underline{M} = \underline{N} = 1.0$; $\underline{\Omega}^0$ and $\underline{A}$ are undefined):

$\underline{\Omega}^1$ = {MIN_COST_P1}

$\underline{\Omega}^2$ = {MIN_COST_P2, MAX_COST_P2, SPT_JOB, LPT_JOB, MIN_TRU_JOB, MAX_TRU_JOB, MIN_TRU_BTU, MAX_RRU, MAX_AVA, MIN_BTU, MAX_BTU}

$\underline{B}$ = {0, 1, 2, 3, 4}

$\underline{C}$ = {0, 1, 2, 3, 4}

$\underline{S}$ = $\{\mathcal{X}\}$

$\underline{T}$ = $\{\mathcal{E}, \mathcal{M}, \mathcal{H}\}$

$\underline{I}$ = {10,000}

Table 2 provides the performance measure BF and the standard deviation STD for one run of the experiment as a function of the control parameter b. The results indicate that the parameter values 1, 2, or 3 dominate the parameter values 0 and 4.

Table 2: Impact of b on BF

b	0	1	2	3	4
BF	0.690	0.855	0.839	0.857	0.556
STD	0.041	0.025	0.026	0.024	0.117

Table 3 presents the performance measure BF and the standard deviation STD as a function of the control parameter c. The results indicate that the parameter value c has some influence, but which value to take is not that evident. Probably there are interaction effects between the priority rules employed and the control parameter c; therefore, we will now have a closer look on the priority rules of phase 2.

Table 3: Impact of c on BF

c	0	1	2	3	4
BF	0.820	0.852	0.841	0.854	0.703
STD	0.030	0.020	0.027	0.025	0.093

Table 4 provides the performance measures BF and STD as well as the average CPU-times in seconds per iteration for the different priority rules employed in phase 2. The results show, that the rule MAX_RRU is, on the average and irrespective of the tractability of the instances, the winner w.r.t. solution quality, while it takes about twice as much CPU-time as the fastest rules.

Table 4: Comparison of Phase 2 Priority Rules

	BF	STD	CPU
MAX_RRU	0.835	0.039	0.068
LPT_JOB	0.833	0.043	0.032
MIN_COST_P2	0.831	0.044	0.032
MAX_TRU_JOB	0.826	0.041	0.032
MAX_AVA	0.825	0.046	0.441
MAX_COST_P2	0.818	0.047	0.032
MAX_BTU	0.813	0.046	0.035
SPT_JOB	0.799	0.049	0.033
MIN_BTU	0.795	0.046	0.034
MIN_TRU_JOB	0.790	0.053	0.033
MIN_TRU_BTU	0.772	0.068	0.033

Now, we analyze the performance of the TPGRM/SPRT-method. Recall that the TPGRM/SPRT eliminates parameter subspaces and thus spends the effort on priority rules and/or parameter subspaces, which seem to be more promising while the TPGRM does not. The experiment is as follows: First, we take only the three best phase 2 priority rules, i.e. $\underline{\Omega}^2$ = {MAX_RRU, LPT_JOB, MIN_COST_P2}. Second, $\underline{B} \subseteq \{\{0\}, \{1,2\}, \{3,4\}\}$ and

$\underline{C} \subseteq \{\{0\},\{1,2\},\{3,4\}\}$, as motivated in Subsection 2.4. In addition, we set $\underline{I} = \{6{,}000; 8{,}000; 10{,}000\}$ for the number of individuals.

We expect that the TPGRM/SPRT improves over the TPGRM. Consequently, as performance measures BF' we take the ratio BF(TPGRM/SPRT)/BF(TPGRM), i.e. the best objective function value of the TPGRM/SPRT related to the best objective function value of the TPGRM. CPU' is defined analogously.

Table 5 presents the redefined performance measures BF' and CPU' as functions of the number of individuals $\underline{I}$. The results show considerable improvements, when the search effort is not equally distributed over all subspaces, but directed towards more promising rules and/or parameters. Obviously, these improvements have been gained w.r.t the learning which is built in the TPGRM/SPRT. Note that only a small additional computational effort is necessary in order to gain these improvements.

Table 5: Comparison 1 of TPGRM/SPRT and TPGRM

$\underline{I}$	6,000	8,000	10,000
BF'	1.101	1.129	1.149
CPU'	1.111	1.124	1.135

Table 6 provides the redefined performance measures BF' and CPU' as functions of the tractability τ for $\underline{I} = 10{,}000$. We see, that the improvements are quite large especially for the easy class of instances.

Table 6: Comparison 2 of TPGRM/SPRT and TPGRM

τ	$\mathscr{E}$	$\mathscr{M}$	$\mathscr{H}$
BF'	1.238	1.101	1.109
CPU'	1.131	1.142	1.132

5.3 Computational Results of the GA

In the following we present results of the GA. Based on preliminary computational results and in order to cut down the computational effort, the experimental design was restricted. The parameters a for BTU-selection and c for job-selection were set to $\underline{A} := \underline{C} := \{0, 2\}$, while the mode-selection parameter b was set to $\underline{B} := \{2\}$. Clearly, for the parameter value 0 we have random sampling, denoted as RANDOM. Hence, we get the following design:

$\underline{\Omega}^0$ = {MAX_BLK, MAX_RES_BTU, RANDOM}

$\underline{\Omega}^1$ = {MIN_COST_P1}

$\underline{\Omega}^2$ = {LPT_JOB, MAX_RES_JOB, RANDOM}

$\underline{A} = \underline{B} = \underline{C} = \{2\}$

$\underline{\mathcal{L}} = \{100\}$

$\underline{M} = \{0.01\}$

$\underline{N} = \{0.5\}$

$\underline{S} = \{\mathcal{X}\}$

$\underline{T} = \{\mathcal{E}, \mathcal{M}, \mathcal{H}\}$

$\underline{\mathcal{I}} = \{10{,}000\}$

Note that $\underline{\mathcal{L}} = \{100\}$ and $\underline{\mathcal{I}} = \{10{,}000\}$ imply that 100 generations have been produced.

In addition to the performance measures BF and STD already introduced in Subsection 5.1 we report in the following the average deviation AVE between the objective function value of an individual and the best known objective function value known for the instance under consideration.

Tables 7, 8, and 9 provide computational results for the easy, medium, and hard instances, respectively. In each case the six most promising combinations of priority rules are reported. The priority rule combinations are sorted in nonincreasing order of AVE. For the sake of shortness $\underline{\Omega}^1$ = MIN_COST_P1 is omitted in all tables.

Table 7: Results for the Easy Instances

$\underline{\Omega}^0$	$\underline{\Omega}^2$	BF	AVE	STD
MAX_RES_BTU	RANDOM	1.0	0.95	0.18
RANDOM	LPT_JOB	1.0	0.95	0.18
RANDOM	RANDOM	1.0	0.95	0.18
MAX_BLK	LPT_JOB	1.0	0.94	0.18
MAX_RES_BTU	LPT_JOB	1.0	0.94	0.18
MAX_BLK	RANDOM	1.0	0.93	0.18

Every procedure, i.e. combination of priority rules, has at least once computed the best known objective function value. Hence, a discrimination between "good" and "even better" procedures w.r.t. this performance measure is impossible. But the AVE performance measure provides more or less promising procedure candidates. For the easy instances the procedure ($\underline{\Omega}^0$, $\underline{\Omega}^1$, $\underline{\Omega}^2$) = (MAX_RES_BTU, MIN_COST_P1, RANDOM) performs best, while for the hard ones (RANDOM, MIN_COST_P1, LPT_JOB) is the winner.

Table 8: Results for the Medium Instances

$\underline{\Omega}^0$	$\underline{\Omega}^2$	BF	AVE	STD
RANDOM	RANDOM	1.0	0.95	0.26
MAX_BLK	LPT_JOB	1.0	0.86	0.26
RANDOM	LPT_JOB	1.0	0.86	0.26
MAX_RES_BTU	LPT_JOB	1.0	0.85	0.26
MAX_RES_BTU	RANDOM	1.0	0.85	0.26
MAX_BLK	RANDOM	1.0	0.85	0.27

Table 9: Results for the Hard Instances

$\underline{\Omega}^0$	$\underline{\Omega}^2$	BF	AVE	STD
RANDOM	LPT_JOB	1.0	0.98	0.04
RANDOM	RANDOM	1.0	0.95	0.13
MAX_RES_BTU	RANDOM	1.0	0.94	0.13
MAX_RES_BTU	LPT_JOB	1.0	0.94	0.14
MAX_BLK	RANDOM	1.0	0.93	0.14
MAX_BLK	LPT_JOB	1.0	0.92	0.16

Table 10 provides the CPU-times in seconds required per individual as a function of the problem size $\underline{S}$. # BTUs, # Jobs, # Res, and CPU denotes the average number of BTUs, jobs, resources, and CPU-seconds required. As expected the computational effort increases drastically with increasing problem size. Clearly, for the large instances one has to spend hours of computation for the generation and evaluation of hundreds of individuals.

Table 10: CPU-times as a Function of the Problem Size

$\underline{S}$	# BTUs	# Jobs	# Res	CPU
extra small ($\mathscr{X}$)	11	35	26	0.04
tiny ($\mathscr{T}$)	25	88	40	0.05
small ($\mathscr{S}$)	69	239	105	0.43
medium ($\mathscr{M}$)	180	624	402	3.98
large ($\mathscr{L}$)	623	2124	1149	42.07

6. Summary and Conclusions

In this paper we propose a new model for timetabling which addresses most of the items which are relevant for applications: Lectures of different length; precedence relations; availability of rooms of different size and equipment; changeover times between rooms; compactness and distribution requirements. In addition, we present greedy randomized and genetic algorithms. We provide an instance generator for the generation of a representative set of instances. The generator along with a statistical model is used for a thorough experimental evaluation of the

methods. Computational results show that the methods solve the instances investigated close to optimality.

In the future first specialized exact methods for solving larger problem instances to optimality should be developed. Second, exchange (cp. Ferland and Lavoie 1992) or local search methods have to be improved, whereas among the most promising meta-heuristics tabu search (cf. Glover 1989, 1990, Hertz 1992, Alvarez-Valdes, Martin, and Tamarit 1994, Costa 1994) has to be considerd. Third, the methods have to be incorporated into a decision support system with database access (cp. Johnson 1993) and easy-to-use dialog capabilities.

Appendix: Fundamentals of Sequential Analysis

Let Y be a Bernoulli distributed (discrete) random variable, i.e.

$$f_Y(y; p) := p^y (1-p)^{1-y} \quad (0 \le p \le 1)$$

with p being a probability. In addition let $Y_1, Y_2, \ldots, Y_n$ represent n identical independent Bernoulli distributed random variables. Denoting with y_i a realization of Y_i, the random variable

$$S_n := \sum_{i=1}^{n} y_i$$

is binomially distributed, denoted as $f_{S_n}(s_n; n, p)$, with distribution function:

$$f_{\mathfrak{X}}(\mathfrak{x}; n, p) := \begin{cases} \left[\begin{matrix} n \\ \mathfrak{x} \end{matrix} \right] \cdot p^{\mathfrak{x}} \cdot (1-p)^{n-\mathfrak{x}}, & \text{for } \mathfrak{x} = 0,1,\ldots,n \\ 0, & \text{otherwise} \end{cases}$$

Moreover, let $f_Z(z)$ denote the distribution function of a continuous random variable Z, ξ_p the p%-quantile of $f_Z(z)$ and $z_1, z_2, \ldots, z_n$ a sample of $f_Z(z)$. Then the transformation

$$y_i := \begin{cases} 1, & \text{if } z_i \le \xi_p \\ 0, & \text{otherwise} \end{cases} \quad \text{for } i = 1, 2, \ldots, n$$

yields $S_n = \sum_{i=1}^{n} y_i$ with distribution function $f_{S_n}(s_n; n, p)$.

Suppose that it is desired to test whether ξ_{p*} is less than or equal a specified value, say ξ. This leads to

$$\mathfrak{H}_0: \xi_{p*} \le \xi \text{ versus } \mathfrak{H}_1: \xi_{p*} > \xi$$

where $\mathfrak{H}_0$ is called the null-hypothesis, being tested, and $\mathfrak{H}_1$ the alternative hypothesis. This test, concerning the quantile $f_Z(z)$, may be transformed into the following *hypothesis testing problem*

$\mathfrak{H}_0$: $p \geq p^*$ versus $\mathfrak{H}_1$: $p < p^*$,

concerning the probability $f_Y(y; p)$ with $0 \leq p^* \leq 1$. We

accept $\mathfrak{H}_0$, if S_n/n is "large" or
reject $\mathfrak{H}_0$, if S_n/n is "small"

where "large" and "small" obviously depend on p^*.

From sequential analysis several tests are known with which hypotheses testing may be performed; cf. e.g. Siegmund 1985. Among them are curtailed tests, repeated significance tests as well as sequential probability ratio tests. In Haase 1994 these tests have been compared via simulation with respect to their capability of approximating the power function (probability of rejecting $\mathfrak{H}_0$, expected number of repetitions of the test; both as a function of p^*). As a result the *sequential probability ratio test (SPRT)* seems to be most suited.

The SPRT, which is based on the likelihood ratio, originally has been designed for simple hypotheses only. Thus for our composite hypothesis we have to make the following transformations:

$\mathfrak{H}_0$: $p \geq p^* \Rightarrow \mathfrak{H}_0$: $p = p^0$
$\mathfrak{H}_1$: $p < p^* \Rightarrow \mathfrak{H}_1$: $p = p^1$ with $p^1 < p^* \leq p^0$

Using the conventional symbols, i.e. α denotes the type I error (significance level) and β the type II error, we get the following stopping and decison rules for the SPRT:

Stopping rule: Stop at $\mathfrak{J} = \min \{ i \mid b_i \notin (A, B)\}$, where

$$b_i := \left[\frac{p^1}{p^0}\right]^{S_i} \left[\frac{1-p^0}{1-p^1}\right]^{S_i - i},$$

$$A := \frac{1-\alpha}{\beta} \text{ and } B := \frac{\alpha}{1-\beta}.$$

Decision rule: Reject $\mathfrak{H}_0$, if $b_{\mathfrak{J}} \geq B$; accept $\mathfrak{H}_0$, if $b_{\mathfrak{J}} \leq A$.

Acknowledgements: The authors are indepted to Stefan Carstens and Harald Hoffmann for the coding of the algorithms. Special thanks go to Andreas Schirmer for his advice concerning Section 4 and for his help to improve the phrasing.

References

Alvarez-Valdes, R., Martin, G., Tamarit, J.M., "Constructing good solutions for a school timetabling problem", Working Paper, University of Valencia, Valencia/Spain 1994.

Badiru, A.B., "Towards the standardization of performance measures for project scheduling heuristics", *IEEE Transactions on Engineering Management*, Vol. 35 (1988), pp. 82-89.

Bean, J.C., "Genetic algorithms and random keys for sequencing and optimization", *ORSA J. on Computing*, Vol. 6 (1994), pp. 154-160.

Colorni, A., Dorigo, M., Maniezzo, V., "Genetic algorithms: a new approach to the timetable problem", in: *Combinatorial Optimization*, Akgül, M. et al. (eds.), Springer, Berlin 1992, pp. 235-239.

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

Davis, E.W., Patterson, J.H., "A comparison of heuristic and optimum solutions in resource-constrained project scheduling", *Management Science*, Vol. 21 (1975), pp. 944-955.

Dorndorf, U., Pesch, E., "Evolution based learning in a job shop scheduling environment", *Computers and Operations Research*, Vol. 22 (1995), pp. 25-40.

Drexl, A., "Scheduling of project networks by job assignment", *Management Science*, Vol. 37 (1991), pp. 1590-1602.

Drexl, A., Haase, K., "Sequential-analysis-based randomized-regret-methods for lotsizing and scheduling", to appear in *J. of the Operational Research Society* (1996).

Drexl, A., Salewski, F., "Timetabling under partially renewable resource constraints", Paper presented at the Joint DGOR/GMÖOR International Conference, Berlin, September 1994.

Feo, T.A., Resende, M.G.C., Smith, S.H., "A greedy randomized adaptive search procedure for the maximum independent set", *Operations Research*, Vol. 42 (1994), pp. 860-878.

Ferland, J.A., Lavoie, A., "Exchanges procedures for timetabling problems", *Discrete Applied Mathematics*, Vol. 35 (1992), pp. 237-253.

Glover, F., "Tabu search - Part I", *ORSA J. on Computing*, Vol. 1 (1989), pp. 190-206.

Glover, F., "Tabu search - Part II", *ORSA J. on Computing*, Vol. 2 (1990), pp. 4-32.

Goldberg, D.E., *Genetic algorithms in search, optimization, and machine learning*, Reading/Mass. 1989.

Grefenstette, J.J., "Incorporating problem specific knowledge into genetic algorithms", in: *Genetic algorithms and simulated annealing*, Davis, L. (ed.): Pitman, London 1987, pp. 42-60.

Haase, K., *Lotsizing and scheduling for production planning*, Springer, Berlin 1994.

Hart, J.P., Shogan, A.W., "Semi-greedy heuristics: an empirical study", *Operations Research Letters*, Vol. 6 (1987), pp. 107-114.

Haupt, R., "A survey of priority rule-based scheduling", *OR Spektrum*, Vol. 11 (1989), pp. 3-16.

Hertz, A., "Finding a feasible course schedule using Tabu search", *Discrete Applied Mathematics*, Vol. 35 (1992), pp. 255-270.

Holland, J.H., *Adaptation in natural and artificial systems*, The University of Michigan Press, Ann Arbor 1975.

Johnson, D., "A database approach to course timetabling", *J. of the Operational Research Society*, Vol. 44 (1993), pp. 425-433.

Johnson, D.S., "Local optimization and the traveling salesman problem", in: *Proceedings of the 17th Colloquium on Automata, Languages, and Programming*, Springer, Berlin 1990, pp. 446-461.

Kelley, J.E., "The critical-path method: resources planning and scheduling", in: *Industrial scheduling*, Muth, J.F., Thompson, G.L. (eds.), Englewood Cliffs, New Jersey 1963, pp. 347-365.

Kolisch, R., *Project scheduling under resource constraints – Efficient heuristics for several problem classes,* Physica, Heidelberg 1995.

Kolisch, R., "Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation", to appear in *European J. of Operational Research* (1996).

Kolisch, R., Sprecher, A., Drexl, A., "Characterization and generation of a general class of resource-constrained project scheduling problems", *Management Science,* Vol. 41 (1995).

Laguna, M., Feo, T.A., Elrod, H.C., "A greedy randomized adaptive search procedure for the two-partition problem", *Operations Research,* Vol. 42 (1994), pp. 677-687.

Liepins, G.E., Hilliard, M.R., "Genetic algorithms: Foundations and applications", *Annals of Operations Research,* Vol. 21 (1989), pp. 31-57.

Michalewicz, Z., *Genetic algorithms + data structures = evolution programs,* Springer, Berlin 1992.

Mühlenbein, H., Gorges-Schleuter, M., Krämer, O., "Evolution algorithms in combinatorial optimization", *Parallel Computing,* Vol. 7 (1988), pp. 65-85.

Panwalkar, S.S., Iskander, W., "A survey of scheduling rules", *Operations Research,* Vol. 25 (1977), pp. 45-61.

Pesch, E., *Learning in automated manufacturing,* Physica, Heidelberg 1994.

Rechenberg, I., *Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution,* Frommann-Holzboog, Stuttgart 1973.

Salewski, F., Bartsch, Th., "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 (1994).

Schwefel, H.-P., *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie,* Birkhäuser, Basel 1977.

Siegmund, D., *Sequential analysis, tests and confidence intervals,* Wiley, New York 1985.

Storer, R.H., Wu, S.D., Vaccari, R., "New search spaces for sequencing problems with application to job shop scheduling", *Management Science,* Vol. 38 (1992), pp. 1495-1509.