Automated solution of a highly constrained school timetabling problem – preliminary results

Marc Bufé, Tim Fischer, Holger Gubbels, Claudius Häcker, Oliver Hasprich, Christian Scheibel, Karsten Weicker*, Nicole Weicker, Michael Wenig, Christian Wolfangel

University of Stuttgart, Faculty of Computer Science, Germany

Abstract. This work introduces a highly constrained school timetabling problem which was modeled from the requirements of a German high school. The concept for solving the problem uses a hybrid approach. On the one hand an evolutionary algorithm searches the space of all permutations of the events from which a timetable builder generates the school timetables. Those timetables are further optimized by local search using specific mutation operators. Thus, only valid (partial) timetables are generated which fulfill all hard constraints.

1 Introduction

Timetabling problems occur in many companies and educational institutions. Especially in education we can distinguish three different types of problems: the school timetabling problem, the university timetabling problem, and the exam timetabling problem. All three problems have a slightly different focus [9]. Therefore, different techniques are necessary for solving them. In this work we consider the school timetabling problem.

The school timetabling problem is a complex real-world problem with many interdependencies. Thus, it can be solved seldom by divide-and-conquer or greedy techniques. Practitioners report that feasible solutions may be found manually within a rather short period of time. However, manual solutions satisfying organizational or didactic requirements may need several person-days work. The timetabling problem has been shown to be NP-complete as soon as unavailabilities of teachers, classes, or rooms are involved [3]. Organizational requirements are of increasing importance since many teachers are only working part-time and inconvenient, flexible time arrangements are necessary for them. Moreover, schools put a lot of thought into didactic issues, e.g. the arrangement of the different subjects within the week. These considerations should be reflected in the timetables.

* correspondence address: Karsten Weicker, University of Stuttgart, Breitwiesenstr. 20–22, 70565 Stuttgart, Germany, Email: Karsten.Weicker@informatik.uni-stuttgart.de

2 The School Timetabling Problem

When talking about school timetabling problems, two different goals in timetabling must be distinguished. First, the simple form of the problem only requires that a feasible solution is found concerning certain hard constraints. However, this is not enough in practice where, second, additional soft constraints should be fulfilled.

The simple timetabling problem involves a set of teachers T, a set of classes C, a set of rooms R, a set of time periods P, and a set of events E. Each event requires a class $c(e) \in C$, a teacher $t(e) \in T$, and a number of weekly hours $hours(e) \in \mathbb{N}$. In order to solve the problem a mapping must be found which assigns each event to a room and a time period for each required hour, i.e.

$$TT : E \ \rightarrow \ 2^{P \times R}$$

where $|TT(e)| = hours(e)$. As an example, let e be chemistry for one class, then $TT(e) = \{(M-1, 101), (W-3, 102)\}$ assigns the chemistry lectures to the first hour on Mondays and the third hour on Wednesdays, the according rooms have the numbers 101 and 102.

Furthermore, a solution of the timetabling problem must suffice the following hard constraints. In order to simplify the notation in the constraints, the components of an assignment $a = (p, r) \in P \times R$ may be accessed by $p(a) = p$ for the time period and $r(a) = r$ for the room.

- Each teacher participates in at most one event per time slot.

$$\forall_{e \in E} \forall_{e' \in E \setminus \{e\}} \ \big(\ t(e) = t(e') \Rightarrow \big(\forall_{a \in TT(e)} \forall_{a' \in TT(e')} p(a) \neq p(a') \big) \big) \qquad (1)$$

- Each class participates in at most one event per time slot.

$$\forall_{e \in E} \forall_{e' \in E \setminus \{e\}} \ \big(c(e) = c(e') \Rightarrow \big(\forall_{a \in TT(e)} \forall_{a' \in TT(e')} p(a) \neq p(a') \big) \big) \qquad (2)$$

- Each room is assigned to at most one event per time slot.

$$\forall_{e \in E} \forall_{e' \in E \setminus \{e\}} \forall_{a \in TT(e)} \forall_{a' \in TT(e')} \ (r(a) = r(a') \Rightarrow p(a') \neq p(a')) \qquad (3)$$

This simple problem is not sufficient for modeling the requirements of a real schools timetable. In the remainder of this section the necessary extensions with additional hard and soft constraints are described.

To begin with, in German schools it is common practice to mix several classes for certain subjects, e.g. for physical education or religious knowledge lessons. As a consequence, more than one class $c(e) \subset C$ is involved in one event. However it is still required that each event must be scheduled to exactly one room. That means that religious knowledge lessons must be split for each age-class into two events. All protestant pupils participate in one event, all catholic pupils in another event. Although both events comprise pupils of the same classes, they are distinct and should be scheduled within the same time periods to guarantee a timetable of high quality. Thus, those events are grouped together in event

groups. All other events are called singular events. The singular events are denoted $E_S \subseteq E$, the grouped events $E_G = E \setminus E_S \subseteq E$. For each event $e \in E$, we denote by $[e] \subseteq E$ the events that belong together, i.e. for $e \in E_S$ it holds that $[e] = \{e\}$ and for $e \in E_G$ all events in the group of e are contained in $[e]$. The following hard constraint is affected slightly by this modification.

– Each class participates in at most one event per time slot.

$$\forall_{e \in E} \forall_{e' \in E \setminus [e]} \ \big((c(e) \cap c(e') \neq \emptyset) \Rightarrow \big(\forall_{a \in TT(e)} \forall_{a' \in TT(e')} p(a) \neq p(a')\big)\big) \tag{2'}$$

In favor of more flexibility in modeling timetabling problems and due to events requiring more than one teacher's attendance, we also allow more than one teacher $t(e) \subset T$ for each event.

– Each teacher participates in at most one event per time slot.

$$\forall_{e \in E} \forall_{e' \in E \setminus \{e\}} \ \big((t(e) \cap t(e') \neq \emptyset) \Rightarrow \big(\forall_{a \in TT(e)} \forall_{a' \in TT(e')} p(a) \neq p(a')\big)\big) \tag{1'}$$

Moreover, teachers, classes, and rooms are extended by further attributes in order to comprise all necessary constraints. First of all, all three basic elements of the problem may be provided with unavailabilities for certain periods, $\mathit{unavail}(t), \mathit{unavail}(c), \mathit{unavail}(r) \subset P$. The according hard constraints read as follows.

$$\forall_{e \in E} \forall_{a \in TT(e)} \ p(a) \notin \mathit{unavail}(t(e)) \tag{4}$$

$$\forall_{e \in E} \forall_{a \in TT(e)} \ p(a) \notin \mathit{unavail}(c(e)) \tag{5}$$

$$\forall_{e \in E} \forall_{a \in TT(e)} \ p(a) \notin \mathit{unavail}(r(e)) \tag{6}$$

In addition to the unavailabilities, most events have special requirements for the equipment of the room, e.g. a biology lab, speech lab, geography room. Those requirements are summarized in room features F. The features are assigned to the rooms $f(r) \subset F$ and the required features to the events $f(e) \subset F$. The required room features must be considered in the assignment process.

$$\forall_{e \in E} \forall_{a \in TT(e)} \ f(e) \subseteq f(r(a)) \tag{7}$$

All constraints (1'), (2'), (3), ... , (7) must be fulfilled by a solution to be feasible.

The succeeding conditions are soft constraints which are encouraged to be fulfilled. They form the didactic quality of the timetable and consider certain organizational issues a timetable must meet in order to be acceptable.

(S-1) Events should be placed more likely in the morning than in the afternoon, e.g. an average class has at most once or twice a week lessons in the afternoon.

(S-2) Many teachers have part-time jobs which means that they should have at least a minimum number of free days each week.

(S-3) Further requirements for certain events like a minimal and maximal number of double hours, fortnightly or marginal placement should be considered.
(S-4) A uniform distribution of the hours of each event over the week is desired (cf. [2]). In addition, certain event groups or singular events can be marked as associated which means that they should not take place at the same day.
(S-5) There should be no gaps in the morning and afternoon schedules of each class and teacher (cf. [4]).
(S-6) For each time period a teacher must be available to supervise a class in case of an unexpected absence of the regular teacher (cf. [2]).

3 Related Work

Automated timetabling has been an issue within the last 30 years. Schaerf [9] gives an overview of the techniques used for the different problems of school timetabling, university timetabling, and exam timetabling. Junginger [6] provides a list of the early approaches to the school timetabling problem in Germany. A survey on the usage of evolutionary techniques used for timetabling and scheduling can be found in [5]

The most available evolutionary approaches to the school timetabling problem use a direct representation of the timetable (e.g. [1, 2, 4]). Those algorithms have the huge disadvantage that the evolutionary operators may produce infeasible timetables which must be repaired by a genetic repair function. This has very often the effect that the correlation between the parent timetables and the offsprings is very low. Where the mutation operator can be chosen appropriately to minimize these effects, this is not possible for the crossover operator. Thus often the crossover acts more like a macromutation than a recombination. Fernandes et. al. [4] used standard operators where Colorni et. al. [1, 2] designed special operators which can only guarantee partial feasibility. As a consequence in [2] it is reported that tabu search outperforms the genetic algorithm in most test cases. Another high-quality tabu search algorithm was presented by Schaerf [8].

Instead of using a direct representation, there are also approaches in university timetabling where a timetable builder is used which generates a feasible timetable from an individual containing several parameters for the timetable builder (e.g. [7]).

4 Concept

In the previous section two different techniques have been pointed out to be very promising in evolutionary timetabling: phenotypic mutations and timetable builders using genotypic operators since in both approaches operators may be defined in such a way that a high correlation between parent and offspring timetables is guaranteed. Furthermore, they can always generate feasible (probably

partial) timetables fulfilling all hard constraints. As a consequence a hybrid approach was used to combine the positive aspects of both standard approaches and to minimize their negative drawbacks. The individuals use two different representations of the timetable: a parametric representation and a high level representation. The parametric representation consists of a permutation of the events which is understood as a queue for the timetable builder. The timetable builder is a deterministic algorithm, described in the next section, which creates a timetable from the queue. Its result is stored in the second half of the individual. Note, that this timetable builder might create only partial timetables where certain events remain unplaced. The different levels of this approach are sketched in Figure 1. The timetable builder is an essential technique since it

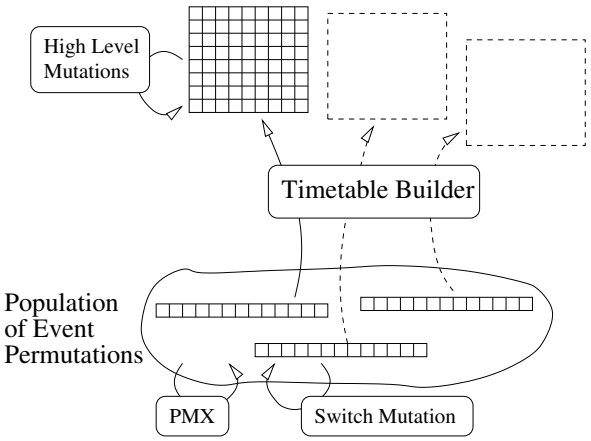


Fig. 1. Hybrid approach using a timetable builder and operators on both levels of representation.

may use many deterministic "intelligent" heuristics to create a timetable. Thus a high quality can already gained by the timetable builder. However, the use of those heuristics narrows the search space critically. Given the huge amount of constraints no timetable builder could be found which actually placed all events as we will discuss in Section 6. As a consequence the phenotypic mutations are necessary to optimize those timetables in order to get all events placed or to fulfill more violated soft constraints.

On the event queues, the evolutionary algorithm uses a mutation swapping two elements and the partial matching crossover (PMX) with the first crossing point fixed at the beginning of the queue to increase the correlation between parents and children. These operators provide the exploration of the search space. Since the timetable builder is deterministic the resulting timetable may be reproduced anytime and slightly changed permutations should also result in related timetables.

```
1: INPUT: list with all hours of one event
2: for daytime = morning, afternoon do
3:    for constraints = all, hardOnly do
4:       for day = M, W, Th, Tu, F do
5:          check for each slot ∈ daytime whether there is a room such that
6:                no unavailability (room, class, teacher) for slot    ∧
7:                room, class, and teacher are disposable    ∧
8:                constraints = all ⇒ soft constraints are fulfilled
9:          if such a room and slot exists then
10:            assign first hour in list to slot
11:         end if
12:      end for
13:   end for
14: end for
15: if not all hours in list are placed then
16:    undo all assignments of the event
17: end if
```

Fig. 2. Placing algorithm within the timetable builder.

Furthermore, various mutation operators are defined on the complex timetables. They respect the feasibility of the timetable and apply little changes to the timetable. The result of this mutation has no effect on the permutation used by the timetable builder. This local optimization of the timetables should encourage a high exploitation of interesting solutions in the search space. Currently three different types of mutation operators are used. The first operator unplaces an event or a part of an event. The second operator tries to place an unplaced event or a partially unplaced event. A the third operator combines both operations by moving an event from one time slot to a different time slot. Also all three operators try to place the unplaced events after their primary operation.

Since the algorithm uses uniform parental selection and only replaces the worst 40 percent of the population in each generation those operators are able to perform an extensive parallel hill-climbing search on the better individuals.

5 Details

Different kinds of constraints need different constraint handling within the evolutionary algorithm. The timetable builder considers the hard constraints (1'), (2'), (3), ... , (7), i.e. all solutions generated from the permutation of events fulfill those requirements – if this is not possible a partial timetable is created and certain events remain unplaced. Also the timetable builder takes care of the soft constraints (S-2) and (S-3) directly. Constraints (S-1) and (S-4) are tried to be fulfilled indirectly by the timetable builder. The high-level mutation operators can only consider the hard constraints (1'), (2'), (3), ... , (7) where soft constraints may be hurt. All soft constraints are considered within the fitness function.

	level of constraint	averaged value	standard deviation
(S-1)	classes C	$E_{\text{S-1}}[TT] = \frac{1}{\lvert C\rvert}\sum_{c\in C}$ violation for c	$\sqrt{V_{\text{S-1}}[TT]}$
(S-2)	teachers T	$E_{\text{S-3}}[TT] = \frac{1}{\lvert T\rvert}\sum_{t\in T}$ violation for t	$\sqrt{V_{\text{S-3}}[TT]}$
(S-3)	events E	$E_{\text{S-4}}[TT] = \frac{1}{\lvert E\rvert}\sum_{e\in E}$ violation for e	$\sqrt{V_{\text{S-4}}[TT]}$
(S-4)	events E	$E_{\text{S-5}}[TT] = \frac{1}{\lvert E\rvert}\sum_{e\in E}$ violation for e	$\sqrt{V_{\text{S-5}}[TT]}$
(S-5)	T and C	$E_{\text{S-6}}[TT] = \frac{1}{\lvert T\cup C\rvert}\sum_{x\in T\cup C}$ violation for x	$\sqrt{V_{\text{S-6}}[TT]}$
(S-6)	periods P	$E_{\text{S-7}}[TT] = \frac{1}{\lvert P\rvert}\sum_{p\in P}$ violation for p	$\sqrt{V_{\text{S-7}}[TT]}$
(P)	timetable	$E_P[TT] = \frac{\lvert unplacedevents\rvert}{\lvert allevents\rvert}$	—
(Q)	timetable	$E_Q[TT] = \frac{\lvert partiallyplacedevents\rvert}{\lvert allevents\rvert}$	—

Table 1. Considered constraints in the fitness function.

The timetable builder consists of three phases. In the first phase, free days are assigned to the part time teachers (cf. S-2) using a round-robin method. This method guarantees that the free days are distributed equally over the week. The strategy which days are assigned to which teacher depends on the event queue in the individual. The second phase, splits up the events into single or paired lecture hours based on the preferences contained in the data (cf. S-3). Then, in the third phase, the placing algorithm in Figure 2 is applied to all events. This algorithm guarantees that first all hours are tried to be placed in the morning (in the first instance under consideration of all constraints, then only meeting the hard constraints). If this fails they are placed in the afternoon. This ensures that constraint (S-1) is fulfilled as good as possible. Also the uniform distribution of the hours of each event over the week (S-4) is encouraged by the order of the days for trying to find a valid time slot.

To define a fitness function the degree of constraint violation has to be measured. Since the different constraints are defined on different levels within the timetable we define first all those measures in Table 1. Then, the fitness function is composed of three components, the number of placed events f, the soft constraints g, and the standard deviation h concerning the soft constraints:

$$f(TT) = \kappa_1 E_P[TT] + \kappa_2 E_Q[TT]$$
$$g(TT) = \sum_{i=1}^{6} \gamma_i E_{\text{S-}i}[TT]$$
$$h(TT) = \sum_{i=1}^{6} \rho_i \sqrt{V_{\text{S-}i}[TT]}$$

where κ_i, γ_i, and ρ_i are adjustable weights. The standard deviation is considered to guarantee a fair distribution of the constraint violations. The fitness function

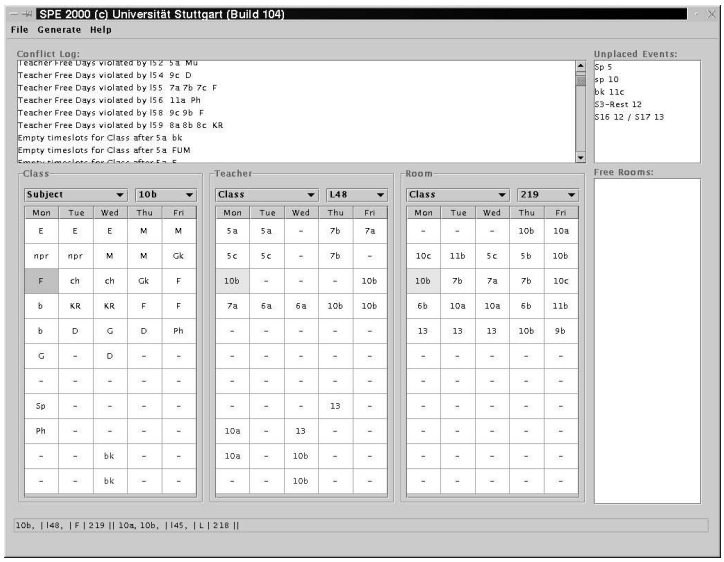


Fig. 3. Graphical user interface displaying the timetables of a class, a teacher, and a room from left to right. Conflicts are displayed in the top section and at the right unplaced events and possible free rooms for an event are shown.

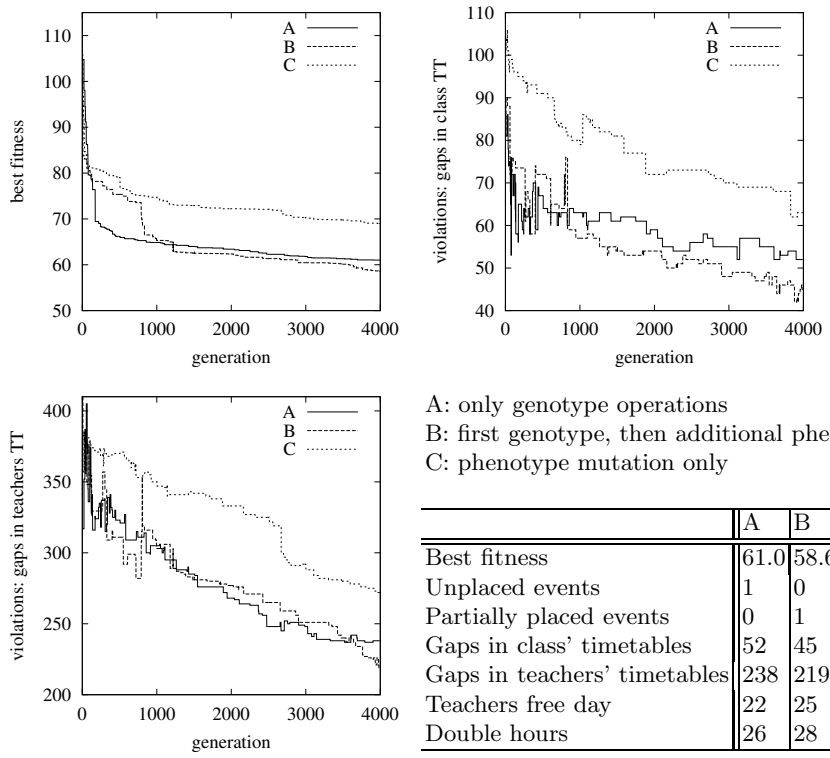
is defined as the length of the vector $(f(TT), g(TT), h(TT))$ which has to be minimized.

6 Results

The presented concept is implemented in Java (cf. Figure 3). It allows not only the generation of a timetable using the evolutionary algorithm, also manual optimizations and further local search are possible.

The timetabling data is provided by a German high school and consists of 61 teachers, 23 classes, 49 rooms, and 351 events. The high school comprises 9 age-classes – the first 7 age-classes are split into three parallel classes, the last 2 age-classes are taught using individual selections of courses. Thus each of the latter age-classes is considered as one class where only fractions take part in various parallel courses. The rooms contain many special rooms for the different subjects.

Several optimization runs have been executed where always 4000 generations are computed using a population size of 20. Because of restrictions caused by the Java programming language the rather small population size is necessary. Each optimization run needs approximately 12 hours computation time. We have carried out experiments using genotype operations only (A), experiments starting with genotype only operations where after 1200 generations the genotype mutation is replaced by the phenotype mutations (B), and experiments using phenotypic mutation only (C).



A: only genotype operations
B: first genotype, then additional phenotype
C: phenotype mutation only

	A	B	C
Best fitness	61.0	58.6	69.0
Unplaced events	1	0	0
Partially placed events	0	1	1
Gaps in class' timetables	52	45	63
Gaps in teachers' timetables	238	219	272
Teachers free day	22	25	28
Double hours	26	28	33

Fig. 4. Experimental results

Due to the expensive fitness function tuning the algorithm is a tedious (and ongoing) task. As a consequence, we could not produce enough results to get statistical confidence for any of the above experimental setups. Thus, we provide the best result for each of the setups.

The experiments show that the phenotype mutations starting after 1200 generations help to reach a better fitness compared to the experiments using genotype operators only. Primary reason for this effect is the improved ability of the phenotype operators to move single hours and thus create partially placed events. This also leads to better scores concerning the gaps in individual timetables. The algorithm using only phenotype mutation is also able to place almost all events but without the timetables of high quality as starting points created by the genotype operators it cannot reach an overall quality comparable to the hybrid approach. Other experiments with slightly changed setups tend to similar results.

These results indicate that the algorithm is an encouraging approach. However, the results are still not good enough to be used in daily school practice. To improve the algorithm in the future, the phenotypic operators are planned to

get more intelligence in choosing time slots for placing events and in resolving constraint violations in the timetable.

7 Conclusion

This work presents an evolutionary timetabling algorithm for highly constrained school timetabling problems. Where most previous work uses an algorithm either on the high-level representation of a timetable or on a low-level encoding (together with a timetable builder) this work chooses a hybrid approach where both techniques are combined. The algorithm respects all hard constraints, i.e. no infeasible solutions are generated, – additional soft constraints may be weighted and are considered within the fitness function. In particular the algorithm is designed in such a way that there is a high correlation between the individuals before and after the operators' application. This enables an effective search where on the one hand the recombination preserves characteristics of parent candidate solutions and on the other hand the mutation on the high-level representation allows the use of very sophisticated heuristics for the fine-tuning of timetables. These heuristics will be included in the near future. The consideration of standard deviations within the fitness function guarantees a fair distribution of soft constraint violations over all events, classes, and teachers.

References

1. Alberto Colorni, Marco Dorigo, and Vittorio Maniezzo. Genetic algorithms: A new approach to the time-table problem. In M. Akgül, editor, *Combinatorial Optimization*, pages 235–239. Springer, Berlin, 1990.
2. Alberto Colorni, Marco Dorigo, and Vittorio Maniezzo. Metaheuristics for high-school timetabling. *Computational Optimization and Applications*, 9(3):277–298, 1998.
3. S. Even, A. Itai, and A. Shamir. On the complexity of timetabling and multicommodity flow problems. *SIAM Journal of Computation*, 5(4):691–703, 1976.
4. Carlos Fernandes, João Paulo Caldeira, Fernando Melicio, and Agostinho Rosa. High school weekly timetabling by evolutionary algorithms. In *ACM SAC 99*, pages 344–350, New York, 1999. ACM.
5. Emma Hart and David Corne. The state of the art in evolutionary approaches to timetabling and scheduling. EvoStim – The EVONET Working Group on Evolutionary Scheduling and timetabling, 1998.
6. Werner Junginger. Timetabling in germany – a survey. *Interfaces*, 16(4):66–74, 1986.
7. Ben Paechter, R. C. Rankin, Andrew Cumming, and Terence C. Fogarty. Timetabling the classes of an entire university with an evolutionary algorithm. In Agoston E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature – PPSN V*, pages 865–874, Berlin, 1998. Springer. Lecture Notes in Computer Science 1498.
8. Andrea Schaerf. Tabu search techniques for large high-school timetabling problems. Technical Report CS-R9611, CWI, Amsterdam, NL, 1996.
9. Andrea Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127, 1999.