A modular hybrid approach to employee timetabling

Drago Bokal
Department of Mathematics
and Computer Science
University of Maribor
Koroška 160
SI-2000 Maribor, Slovenia
drago.bokal@uni-mb.si

Gašper Fijavž
Faculty of Computer and Information Science
University of Ljubljana
Tržaška 25
SI-1000 Ljubljana, Slovenia
gasper.fijavz@fri.uni-lj.si

Bor Harej
Department of Mathematics
University of Ljubljana
Jadranska 19
SI-1000 Ljubljana, Slovenia
bor.harej@gmail.com

Andrej Taranenko
Department of Mathematics
and Computer Science
University of Maribor
Koroška 160
SI-2000 Maribor, Slovenia
andrej.taranenko@uni-mb.si

Klemen Žagar
Cosylab d.d.
Teslova 30
SI-1000 Ljubljana, Slovenia
klemen.zagar@cosylab.com

July 25, 2008

Abstract

We consider a classical employee timetabling problem: a set of employees with various skill levels, qualifications, workload and availability distributions has to be assigned to a set of shifts, each requiring a prescribed number of qualified employees and spanning a given time period.

The novelty of our approach is a hybrid combination of the methods proposed in bibliography, such that we leverage the advantages of known methods while minimizing their disadvantages. Thus, we first apply the generalized local hill-climbing with randomized neighborhoods to quickly reach the vicinity of local optima, and then use the tabu search to explore more of the search space around those solutions. Our experiments show that the resulting hybrid technique performs better than the comparable approaches presented in bibliography due to the hybrid nature of the technique.

In addition, we propose a modular design that utilizes dependency injection to compose the the search algorithm. Together with careful modeling, this approach allows for constant-time evaluation of each possible step in the neighborhood and for an easy evaluation of different hybrid combinations that can be combined and parameterized at runtime.

Keywords: employee timetabling, generalized local hill-climbing, tabu search, hybrid local optimization

1 Introduction

Employee timetabling problems are ubiquitous in large enterprise information systems. Bibliography reports about applications of automated timetabling algorithms in health care [Berghe, 2002; Burke *et al.*, 1999; Chiarandini *et al.*, 2001; Meisels and Schaerf, 1999] and bus crew scheduling [Voss and Daduna, 2001], but the range of applications is far from exhausted with these two. For instance, the commercial version of the system described in this paper has been applied in scheduling of airport and gaming facility crews, and is planned to be used for scheduling of bus fleet crews as well.

AMS Subject Classification (2000): 90B35, 90B70, 90C59

The employee timetabling problem in our model consists of a set of employees and a set of shifts, with each shift requiring a specific number of qualified employees. The solution to the problem is an assignment of employees to shifts, such that it satisfies the given constraints. Constraints mandate that the assigned work to each employee respects the policies of the organization (e.g., employees have required qualifications), that the legal regulations and employee's work contract are respected, that the schedule is historically fair for each employee and that the overall quality of the work process is maximized.

The underlying model we use is similar to the one described in [Chiarandini *et al.*, 2001; Meisels and Schaerf, 1999]. An important difference is that the tasks within a shift as well as shift's location are fixed, therefore assigning an employee to a shift determines both his/her task and location. We solved the problem using the generalized hill-climbing techniques with randomized neighborhoods from [Meisels and Schaerf, 1999] and the tabu search proposed in [Chiarandini *et al.*, 2001]. We replicated the results of [Meisels and Schaerf, 1999]: generalized hill-climbing with randomized neighborhoods does outperform the neighborhood-exhaustive generalized local hill-climbing if the methods are ran only for a comparatively short timed period. However, our special attention was devoted to efficiently computing the changes in the cost function between two solutions in the neighborhood. We managed to evaluate these changes in constant time, and thus were able to run both randomized and neighborhood-exhaustive generalized local hill-climbing to the convergence of the two methods. In this experiment, the exhaustive method needed substantially more time, but gave better results than the randomized method.

We used dependency injection [Fowler, 2004] to design a modular system in which the methods could be arbitrarily combined and adapted to each customer using a commercial application framework. Using this modular approach we designed a hybrid optimization technique that combines the better of both previous methods: generalized local hill-climbing is used to quickly produce a high-quality starting solution, and tabu search is used to finetune this solution. Furthermore, due to the special structure of the employees in the test problems, we discovered that for all methods (generalized hill-climbing, tabu search, and the hybrid method), a three-phase optimization performs best (both in quality and time-wise): the first phase distributes the work among regular employees, the second among the short-term contract employees, and the third optimizes for soft constraints.

The rest of the paper is organized as follows: Section 2 describes the model and its constraints in a greater detail. Section 3 reveals the details about the search techniques that we used, with special attention devoted to the multiphase hybrid technique in Section 4. Section 5 presents the modular design of the developed system and Section 6 presents the results of comparison of all the developed methods. Final remarks are given in Section 7.

2 Constraints and the cost function

The model is constructed following [Chiarandini *et al.*, 2001; Meisels and Schaerf, 1999; 2003]. Its main entities are employees, workdays and shifts. Each shift is performed on a specific workday. For each employee, his/her availability for each workday is known, as well as the overall minimum and maximum desired workload. For each employee, the list of feasible shifts is given, together with the suitability of the employee to work on a specific shift. Furthermore, for each employee and each pair of shifts, we also know whether the employee is allowed to work on both of those shifts, and if so, how suitable he/she is to work on the shifts.

A solution—assignment of employees to shifts—is feasible, if all of the hard constraints are met.

The quality of a feasible solution is measured according to the soft constraints.

Finding a single feasible solution is a difficult task (the corresponding decision problem is NP complete, as it contains list multicoloring of graph vertices as a subproblem [Marx, 2004; 2002]). Thus, we include nonfeasible solutions into the extended search space, heavily punishing any violation of the hard constrains as suggested in [Chiarandini *et al.*, 2001; Meisels and Schaerf, 1999].

The list of hard constraints includes:

(H1) Every shift is assigned a sufficient number of employees.

(H2) Employees can only be assigned to shifts they are qualified to work on.

(H3) No conflicting pair of shifts is assigned to the same employee.

(H4) The workload of every employee stays within prescribed employee-dependent lower and upper bounds.

(H5) The maximum number of consecutive working days is bounded; the bound may be employee-dependent.

(H6) The minimum number of consecutive working days is bounded, the bound may be employee-dependent (and not equal to 1).

(H7) The number of free days between two consecutive runs of working days may depend on the number of working days therein, i.e. an employee must be given two free days between two consecutive runs of, say, five working days, but can be assigned a single free day if both working-day-runs last, say, just four days.

The set of employees is partitioned into types. Employees of the same type are hired under comparable working contracts. Consequently, employees of the same type have the same forbidden shift pairs (H3).

The soft constraints are described as follows:

(S1) The price for employees to work on shifts, which is calculated independently for all feasible employee-shift pairs.

(S2) Suitability of employee types being assigned to work.

(S3) Suitability of pairs of shifts being assigned to the same employee (which for every pair of shifts depends on the employee type, similarly as in the case of (H3)).

The generalized cost is a sum of hard and soft cost functions. The hard cost function essentially counts the number of weighted violations of each of the hard constraints (H1–H7), multiplied with a large-enough coefficient. The soft cost function is computed as a linear combination of prices according to constraints (S1–S3). Choosing a large coefficient for the hard cost function makes our solution converge to a feasible one.

Additional auxiliary information is associated with each solution which enables us to test the relative quality of a neighboring solution (the one obtained from the current solution with either addition, removal, or exchange of employee(s) on a shift) in constant time with no restrictions on the number of employees, shifts or working days. This model can be generalized to include other constraint satisfaction problems related to assignment. The details and the generalization will be described in a forthcoming paper [Bokal and Fijavž, 2008].

3 Search techniques

The problem described in the previous section is solved using local search. The following techniques are recommended in the literature [Burke *et al.*, 1999]:

- Generalized local hill climbing GLHC [Meisels and Schaerf, 1999; 2003]: hill climbing is a family of techniques based on the idea of performing only operations that improve the value of the cost function f (or leave its value unchanged).
- Tabu search [Berghe, 2002; Burke *et al.*, 1999; Chiarandini *et al.*, 2001]: Tabu search is an extension of the local hill climbing technique that is capable of stepping out of local minima. To avoid retracing the steps used to get into a local minimum, the method records recent moves in one or more tabu lists. Avoiding reverse moves, tabu search chooses the best neighboring solution that may be worse than the original one. The main difference compared to the generalized local hill climbing is the prohibition mechanism implemented by the use of the tabu list.

These search methods use a neighborhood function N to obtain a list $N(s)$, the neighborhood of s, that contains the solutions among which the search methods choose the solution for the next iteration. The maximal neighborhood $N(s)$ contains all the solutions that are obtained from the solution s either by the operation *replace(S,E,E')* that replaces the employee E with the employee E' on the shift S, the operation *insert(S,E')* that inserts a new employee to a shift, or the operation *delete(S,E)* that deletes an employee from a shift. In this context, some parameters of the operations can be fixed to define a smaller neighborhood. In particular, [Meisels and Schaerf, 1999; 2003] suggest the following types of neighborhoods:

- RRB. We randomly select a shift S and its employee E. For employee E', we select an employee that is currently not assigned to S and for which the operation *replace(S,E,E')* yields maximal improvement of the cost function.
- RBB. We randomly select shift S and choose employees E and E' so that *replace(S,E,E')* maximally improves the value of the cost function.
- BBB. We choose the triple S, E and E' for which *replace(S,E,E')* maximally improves the value of f.

4 Fast hybrid local search for employee timetabling

As discussed in Section 6, several combinations of the above techniques with the listed neighborhood structures retain the appealing properties of the techniques. Generalized local hill climbing in combination with the RRB neighborhood structure returns a good suboptimal solutions quickly. On the other hand, tabu search combined with the BBB neighborhood structure is slower, but yields better solutions. The modularity of our approach (cf. Section 5) allowed us to test various combinations of the above methods and apply the advantages of both techniques into a hybrid local search. In the phases of the overall method discussed in Section 4, we interlace a run of generalized local hill climbing with the RRB neighborhood structure that quickly converges to a local optimum, and a run of tabu search with BBB neighborhood structure that explores a wider region of the search space around the local optimum.

The timetables, obtained by either generalized local hill-climbing or by tabu search, were unsatisfactory. Analysis of the constraint violations in the final solutions indicated that the problem lies in the structure of the employees: the regular employees, to whom the algorithm was supposed to distribute most of the workload, were highly constrained by the work regulations, but the short-term contract employees that were supposed to be auxiliary workforce were less constrained. As a result, the algorithm distributed too much workload to the short-term contract employees so that the regular employees did not get assigned their required monthly amount of work.

The problem was solved by splitting the optimization into three phases (for results consider Table 1):

- First phase: distributing the workload among the regular employees. The employees were weighted according to their type in such a way that assignment of a short-term contract employee to a shift was heavily discouraged.
- Second phase: obtaining a feasible solution. The weight of the short-term contract employees was decreased, and in this phase we tried to eliminate as many of the remaining violations of hard constraints as possible using both types of employees.
- Third phase: obtaining a near-optimal feasible solution. Only in the third phase we accounted for the soft constraints, keeping the relative weights of the hard constraints the same as in the second phase.

The phases are distinguished by their cost functions, which implies that the final timetable of one phase (its local optimum) is in most cases not the (locally) optimal timetable with respect to the cost function of the next phase.

The modularity of presented approach allowed the decomposition of the search into phases, making use of faster search methods in the initial phase, of slower but more exhaustive search methods in the later phases, and to completely rule out computation of several components of the cost function that make little or no sense in certain phases. For instance, the number of assigned employee-shift pairs which violate e.g. (H6) is of no information in the initial phases, and by avoiding its computation the performance of the search increased.

5 Modular design and dependency injection

The search techniques described in Section 3 all belong to commonly used local optimization techniques, and can thus be regarded as walks in the solution space of the employee timetabling problem. Using this abstraction, the techniques differ only in the following aspects of the walk:

- *Determining the neighborhood* $N(s)$ of the current solution s. The neighborhood depends on the operation, i.e., assignment, deletion, or exchange of an employee on a shift, and on whether one or more parameters are chosen randomly.
- *Evaluating the alternative solutions* in the neighborhood $N(s)$. The evaluation depends on the list of constraints used for the current timetabling problem.
- *Updating the current solution* s on the basis of the evaluation. This step depends on the algorithm used: generalized local hill climbing accepts only better solutions, whereas tabu search may accept an inferior one.

The above tasks are independent of each other, and are for a particular technique combined together in a specific way. In order to capitalize on this implicit structure of the techniques, we decided to make it explicit by developing *data structures* for each of these *algorithm tasks*. We isolated and implemented the following abstract entities and their operations using the object-oriented programming approach:

- A *constraint* can perform the following operations:
  - determine the change of the value of a specific constraint when assigning or relieving an employee from a shift,
  - count and list the violations of the constraint appearing in a specific timetable.
- A *neighborhood structure* can perform the following operations:
  - maintain a list of constraints used to search for a timetable,
  - using the list of constraints, determine the change of the value of the cost function when assigning or relieving an employee from a shift,
  - using the list of constraints, count and list all the violations appearing in a specific timetable,
  - choose the most suitable parameters (shift, employee), for a given operation according to the specific neighborhood and the list of applicable constraints.
- An *algorithm* can perform the following operations:
  - maintain the neighborhood structure used for walking through the solution space,
  - choose the next operation to be performed,
  - choose the parameters of the next operation, using the specifics of the algorithm and the given neighborhood structure,
  - perform the operation using the specifics of the implemented algorithm,
  - perform the optimization until the stopping condition is met.

To make the above structures reusable for all the contexts in which they are required, we have made use of the *dependency injection* software design pattern, where a *module assembler* [Fowler, 2004] instantiates and links the structures as required for a given task.

All the structures had access to the problem data in order to manipulate and maintain it; the dependency diagram is presented in Figure 1.

The dependency injection approach allowed us to configure the parameters of the search technique at runtime, as well as combine different techniques in several ways, among which the hybrid local optimization technique described in Section 3 turned out to give the best results (see Table 3).

The presented modularity proved valuable when the application was extended to a slightly different setting, where the focus was on determining the changes in the planned timetable due to changes in the availability of employees (e.g. sick leaves). The only change we added to the algorithm was a new soft constraint that penalized changing of the assigned shift whenever this was not necessary due to the change of availabilities. The change may still be necessary in order to produce a feasible timetable, but the presence of this new soft constraint allowed the same algorithm as before to minimize not just the number of violated constraints, but also the number of adjustments needed to produce a feasible schedule. In a similar manner we added a constraint penalizing employees according to their employment type when we discovered that this is a bottleneck when distributing the workload among regular and short-term contract employees (cf. Section 4).

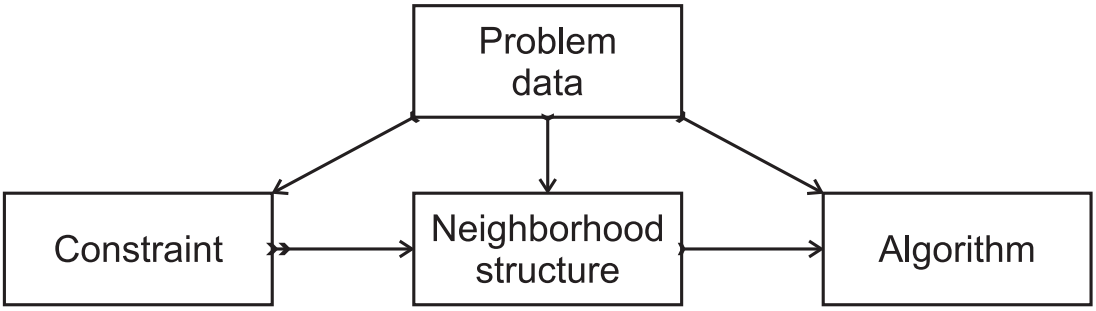

Figure 1: Dependencies among the entities involved in search techniques.

The modularity also allowed for a systematic treatment of the constraints described in Section 2. We developed auxiliary data structures so that evaluating the change of the partial cost function with regard to a specific constraint for a proposed operation took constant time. The benefits are discussed in Sections 3 and 6.

6 Results

In this section we present the performance evaluation of the algorithms on three different problems in monthly employee timetabling.

- T_1: The basic problem on 121 employees.
- T_2: A variant of T_1; a couple of employees in T_1 were consistently producing violations of having too many consecutive working days in the beginning of the month — together with working days of the previous month, which are hard coded into the input data. We have relaxed the previous month's conditions on exactly two employees that were causing problems.
- T_3: 151 employees with relaxed conditions on minimal and maximal workload.

Each of the problems contains almost 600 different shifts distributed over 30 days, cumulatively having around 1350 employee-day requests. The problems all originate in the gaming business. The shifts were distributed around the clock with no suitable condition to canonically separate the problem into working days of the month.

The tests present performance of both Tabu and GLHC search techniques with both RRB and BBB neighborhood types and comparison of the results with the hybrid technique (see Tables 1, 2, and 3).

6.1 Phases and soft constraints

The sets of employees in our problems could be partitioned into two blocks: *regular employees* with strong restrictions on minimal and maximal workload (hired on a full time basis) and *short-term contract employees* (part time, student jobs).

A single phase of the optimization consists of a cost function φ, a search technique employed (Tabu or GLHC), and a neighborhood type (RRB or BBB). As noted in Section 4 one phase approaches are not suitable for our problems. A 3-phase approach uses the same search technique (Tabu or GLHC)

	1-phase		2-phase		3-phase	
	Avg	Stdev	Avg	Stdev	Avg	Stdev
GLHC RRB	80.3	6.34	65.53	5.55	**2.6**	1.71
GLHC BBB	76.8	4.47	67.6	2.35	**1.06**	1.08
Tabu RRB	283.5	32.7	209.8	15.0	287.3	18.9
Tabu BBB	75.7	3.14	67.5	1.92	**1.03**	1.21

Table 1: Comparing 1, 2 and 3 phase approaches: number of violations of hard constraints, 30 runs, problem T_1.

	1-phase	2-phase	3-phase
GLHC RRB	9.9	15.2	10.8
GLHC BBB	599.6	499.4	829.7
Tabu RRB	2.1	2.6	2.9
Tabu BBB	723.8	539.0	615.4

Table 2: Average running times in seconds, problem T_1.

and the same neighborhood type (RRB or BBB) from beginning to the end of the optimization. It employs a sequence of cost functions $\varphi_1, \varphi_2, \varphi_3$, with each phase terminating after a number of consecutive steps have yielded no progress (typically less than 200 steps). The best solution found in a phase is then used as the starting solution of the next phase.

We have compared 1-, 2-, and 3-phase approaches to our problem (see Table 1). All the tests have used the same cost function φ_{final} in their respective last phases and the same halting criterion: 200 consecutive steps with no progress.

The poor performance of Tabu RRB relative to GLHC RRB is because Tabu is required to make a move, whereas GLHC on the other hand may pass on a suggested move. Limiting to a small subsets of neighbors of the current solution (using RRB) may force the algorithm to actually take many moves which reduce the quality of the solution before finally finding one that improves on the quality.

The results in Table 1 indicate that the 3-phase BBB approaches (Tabu and GLHC) produce solutions of better quality than any other technique. Also note that 3-phase GLHC RRB comes in third regarding output quality.

Local search using RRB neighborhood type significantly reduces the running time compared to BBB techniques. The average running times are presented in Table 2 (in seconds using a standard office PC - Pentium 4 3GHz, 1GB RAM).

6.2 Hybrid approach

The ability of Tabu BBB to produce solutions of superior quality, and fast convergence with reasonable quality of GLHC BBB have inspired us to combine both approaches into a hybrid technique: we split each phase of the three phase method into a pair of phases utilizing the same cost function. The first solution is obtained using the faster GLHC RRB method, which is subsequently used as a starting solution for Tabu BBB. Hence, the hybrid approach we have investigated is a six phase approach, with vague convergence stopping criteria in intermediate phases. We emphasize that the very same Tabu and GLHC implementations that were described in the previous section were used in the hybrid

Problem	Algorithm	Avg	Stdev	Time
T_1	GLHC RRB	2.6	1.7	**10.8**
	GLHC BBB	1.07	1.08	829.7
	Tabu BBB	1.03	1.21	615.4
	Hybrid	**0.63**	**0.85**	235.9
T_2	GLHC RRB	1.9	1.9	**8.8**
	GLHC BBB	1.13	1.5	697.2
	Tabu BBB	0.57	0.9	510.0
	Hybrid	**0.53**	**0.86**	174.4
T_3	GLHC RRB	0.03	0.18	**12.2**
	GLHC BBB	**0**	**0**	894.0
	Tabu BBB	**0**	**0**	457.1
	Hybrid	**0**	**0**	199.2

Table 3: Comparison over three data sets; number of violations, running time; 30 runs.

approach; this was possible due to the aforementioned modular design of the algorithms.

The partial usage of RRB neighborhood structure in the hybrid technique caused a significant speed-up over 3-phase GLHC and Tabu BBB approaches. Also, the quality of the output compares favorably, as can be seen in Table 3.

6.3 Quality spectra

We have used cost functions that combine both requirements: meeting all of the hard constraints and sufficient quality with respect to the soft constraints. The coefficients of the hard constraints were tuned so that the search techniques produced as few violations of these as possible. On the other hand, we may regard the relative weights of soft constraints as fixed.

Whereas maximizing the total sum of the suitabilities of employees does help improve the quality of the timetable, it could still happen that only minority of the assignments would be highly suitable, whereas the majority would be of low quality. It is not clear whether such an assignment with few exceptionally well-assigned shifts is better than a timetable in which there are no exceptionally well-assigned shifts, but the shifts in general get a more balanced assignment. For this reason we also evaluated the distribution of the suitabilities and not just their sum in the designed timetables. In this section, we formalize this concept and present the results of the evaluations.

Assume we have constructed a feasible timetable T in which an employee e is assigned to work on a shift s_k on a working day d. Let $s_1, s_2, \ldots, s_n$ be the sequence of shifts of day d so that:

(i) e is allowed to work on shifts s_i, $i = 1, \ldots, k$, ie. assigning e to work on any of these shifts does not cause a violation of (H2).

(ii) The shifts s_i are sorted according to the cost of e working on s_i i.e. the constraint (S1) suggests that assigning e to work on s_i is preferred to assigning him/her to work on s_j, if $i < j$.

The *grade* of the employee-shift pair e, s_k on his/her working day d is defined as $\lceil \frac{5k}{n} \rceil$. The choice of assigning e to work on s_k is *first grade* if the shift s_k is among top 20% of shifts on which e can be assigned to work on (among shifts of day d).

	Shift-employee					Employee-shift				
Grade	1st	2nd	3rd	4th	5th	1st	2nd	3rd	4th	5th
GLHC RRB	23.8	47.3	8.0	1.1	17.1	70.4	12.2	11.6	3.9	0
GLHC BBB	23.7	46.5	8.2	1.5	17.4	69.4	12.4	12.3	3.8	0
Tabu BBB	23.5	46.4	8.5	1.4	17.7	69.5	12.0	12.5	4.0	0
Hybrid	23.8	47.3	8.2	1.1	17.0	70.2	12.1	12.0	3.8	0

Table 4: Comparison of quality spectra for the constraint (S1) in the example T_1.

The *employee-shift quality spectrum* is the set of relative grade frequencies averaged over all employees and their working days in T.

We define *grades* of shift-employee pairs in a similar way: given a shift s and an employee e_k assigned to work on s in T, we sort the employees $e_1, e_2, \ldots, e_m$ according to the constraint (S1) taking into account only the employees which can be assigned to work on this shift and which were indeed assigned to work on the day of shift s in timetable T. The *shift-employee quality spectrum* is the set of relative grade frequencies averaged over all shifts.

The quality spectra shown in Table 4 indicate that for both quality spectra considered, the produced solutions accounted for the preferences of the employees rather well, as most of the assignments were either first or second grade. Furthermore, the hybrid method marginally outperforms Tabu BBB as well as GLHC BBB with regard to both spectra.

To conclude this section, we point out that GLHC with RRB neighborhood does perform best when only comparably short time is allowed for the execution of the optimization (in accordance with the results of [Meisels and Schaerf, 1999]). On the other hand, if the methods are allowed a sufficiently long time to converge, then those utilizing the exhausting neighborhood search produce timetables of higher quality. The results indicate that the proposed new hybrid method is faster than GLHC BBB and Tabu BBB for a considerable factor, is more likely to produce a feasible solution, and gives solutions of slightly better distribution of the suitability of assigned shifts than these two methods individually. As the hybrid approach utilizes precisely the same Tabu Search and GLHC implementations as it was compared to, we conclude that the better performance is a consequence of the hybrid nature of the proposed method.

7 Final remarks

In this contribution, we describe a novel approach to employee timetabling. Following the previous bibliography that favored generalized local hill climbing using various neighborhood types or tabu search, we combine the good aspects of these approaches and demonstrate that the resulting hybrid technique outperforms the previous approaches. We also propose a modular design that uses the dependency injection software design approach to combine pieces of the search algorithms. This allows for an easy evaluation of different hybrid approaches that can be combined and parametrized at runtime. Furthermore, we propose quality spectra as a measure of the distribution of the suitability of the shift assignments in the produced timetables. This measure can be easily adopted to measure quality of timetables regarding other weak constraints.

The proposed techniques were applied in an employee timetabling module of the SchedulerExpert software [ICIT, 2008] and used to produce employee timetables in airport and gaming facilities.

In our future research work, we plan to develop an abstract relational model for constraint satisfaction problems that will generalize the model used in this paper and will fully capitalize on the constant-time criterion function change evaluation, [Bokal and Fijavž, 2008]. In addition, we intend to use the modularity of the developed system to study adaptive methods of self-configuration of the hybrid local search for specific customers, whose monthly schedules have similar structure and to which the parameters of the search can be fine-tuned.

Note added in proof. After submitting the paper to the PATAT 2008 conference, we learned that an approach similar to ours was proposed at a somewhat higher level in [Di Gaspero and Schaerf, 2006]. Their approach also uses a variety of neighborhoods and algorithms, and combines them in a manner similar to ours, but automated. We believe that the constant-time neighboring solution evaluation method applied in our study can be combined with their approach, and propose this as another direction for further research.

Acknowledgement

The research is supported by ICIT d. o. o. and HIT d. d. through the SchedulerExpert project, `www.icit.si`. We are also greateful to anonymous referees for useful comments that helped improve the paper.

References

[Berghe, 2002] G. V. Berghe. *An Advanced Model and Novel Meta-heuristic Solution Methods to Personnel Scheduling in Healthcare*. PhD thesis, University of Gent, 2002.

[Bokal and Fijavž, 2008] D. Bokal and G. Fijavž. Relational model for CSP that supports constant time difference evaluation. in preparation.

[Burke *et al.*, 1999] E. Burke, P. De Causmaecker, and G. V. Berghe. A hybrid tabu search algorithm for the nurse rostering problem. In B. McKay, X. Yao, C. S. Newton, J. H. Kim, and T. Furuhashi, editors, *Simulated Evolution and Learning*, number 1585 in Lecture Notes in Computer Science, pages 187–194. Springer, Berlin, 1999.

[Chiarandini *et al.*, 2001] M. Chiarandini, A. Schaerf, and F. Tiozzo. Solving employee timetabling problems with flexible workload using tabu search. In E. Burke and W. Erben, editors, *Practice and theory of automated timetabling III*, number 2079 in Lecture Notes in Computer Science, pages 298–302. Springer, Berlin, 2001.

[Fowler, 2004] M. Fowler. Module assembly. *IEEE Software*, 21:65–67, 2004.

[Di Gaspero and Schaerf, 2006] L. Di Gaspero, A. Schaerf. Neighborhood portfolio approach for local search applied to timetabling problems. *Journal of Mathematical Modelling and Algorithms*, 5:65–89, 2006.

[ICIT, 2008] ICIT d.o.o. Scheduler expert, 2008. `http://www.icit.si/`.

[Marx, 2002] D. Marx. The complexity of tree multicolorings. In *Mathematical Foundations of Computer Science 2002 (Warsaw-Otwock)*, pages 532–542. Springer, Berlin.

[Marx, 2004] D. Marx. Graph coloring problems and their applications in scheduling. *Periodica Polytechnica Ser. El. Eng.*, 48:5–10, 2004.

[Meisels and Schaerf, 1999] A. Meisels and A. Schaerf. Solving employee timetabling problems by generalized local search. In *Proc. Ital. AI-1 (Bologna, Italy, May 1999)*, pages 281–331.

[Meisels and Schaerf, 2003] A. Meisels and A. Schaerf. Modelling and solving employee timetabling problems. *Ann. Math. Artif. Intell.*, 39(1-2):41–59, 2003. Foundations of artificial intelligence, VI.

[Voss and Daduna, 2001] S. Voss and J. R. Daduna, editors. *Computer-aided scheduling of Public Transport*. Lecture Notes in Economics and Mathematical Systems. Springer, Berlin, 2001.